

**Ph.D. Thesis**

submitted in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy in Computer Science

**Enabling Collaboration  
on Semiformal Mathematical Knowledge  
by Semantic Web Integration**

Christoph Lange

Submitted: 2011-01-31

Defended: 2011-03-11

Approved: 2011-03-11

Dissertation Committee:

Prof. Dr. Michael Kohlhase, Jacobs University Bremen (supervisor)

Prof. Dr. Peter Baumann, Jacobs University Bremen

Prof. Dr. Stefan Decker, National University of Ireland, Galway

Reproduced with permission from AKA and IOS Press\*

\*Original publication:

Christoph Lange: *Enabling Collaboration on Semiformal Mathematical Knowledge by Semantic Web Integration*. Volume 011 in *Studies on the Semantic Web* (series editor: Pascal Hitzler). AKA Verlag Heidelberg and IOS Press, 2011. ISBN 978-3-89838-657-9 (AKA) and 978-1-60750-840-3 (IOS Press).



Den frühen Förderern meiner Liebe zur Wissenschaft:  
meinen Eltern und meinen Lehrern



---

I hereby declare that this thesis has been written independently, except where sources and collaborations are acknowledged, and has not been submitted at another university for the conferral of a degree.

As the chapters are organized by topic, whose sections cover diverse subtopics, reviews of the state of the art and related work of others closely precede the descriptions of my own research on that, but are marked accordingly. I will report of work that I have done myself in the first person singular. Work done in collaboration with others will be reported in the first person plural; the exact contributions of collaborators will be acknowledged at the end of each chapter.

Christoph Lange



# Contents

<b>I</b>	<b>Introduction</b>	<b>1</b>
<b>1</b>	<b>Web Collaboration on Mathematical Knowledge</b>	<b>3</b>
1.1	Current Practices of “Doing Mathematics” . . . . .	4
1.2	Enabling Management, Understanding, and Application of Mathematical Knowledge . . . . .	6
1.3	Web 2.0 and Semantic Web in Science . . . . .	7
1.3.1	The Original World Wide Web . . . . .	7
1.3.2	Web 2.0 – Social Networks and User-Generated Content . . . . .	7
1.3.3	Semantic Web and Linked Data . . . . .	8
1.3.4	Combining Web 2.0 and Semantic Web – Benefits and Challenges . . . . .	9
1.4	Mathematics on the Web – State of the Art and Challenges . . . . .	10
1.4.1	Web 1.0 – Publication Databases, Formalized Libraries, Educational Content . . . . .	11
1.4.2	Web 2.0 – Open Collaboration in Blogs and Wikis . . . . .	12
1.4.3	Semantic Web for MKM – Digital Libraries and Web Services . . . . .	16
1.5	Collaborative Mathematics on the Web – Why Retry Now? . . . . .	19
1.5.1	Combining Semantic Web and Web 2.0 for MKM . . . . .	19
1.5.2	What MKM can Contribute to the Semantic Web . . . . .	19
1.6	Challenges to be Addressed by a New MKM Infrastructure . . . . .	21
1.6.1	The User’s Perspective: Providing Incentives to Contributors . . . . .	21
1.6.2	Research Questions . . . . .	23
1.7	Structure and Contribution of this Thesis . . . . .	25
<b>II</b>	<b>Knowledge Representation</b>	<b>29</b>
<b>2</b>	<b>Representing Mathematical Knowledge</b>	<b>33</b>
2.1	Structures of Mathematical Knowledge . . . . .	34
2.1.1	General Concepts and Terminology . . . . .	34
2.1.2	Logical and Functional Structures . . . . .	37

2.1.3	Rhetorical Structures . . . . .	39
2.1.4	Document Structures . . . . .	40
2.1.5	Presentation and Notation . . . . .	40
2.1.6	Dependencies . . . . .	41
2.1.7	Metadata . . . . .	43
2.1.8	Discussions in Mathematical Collaboration . . . . .	49
2.2	Requirements for Reusably Representing and Exchanging Mathematical Knowledge . . . . .	51
2.3	Knowledge Representation on the [Semantic] Web (State of the Art) . . . . .	51
2.3.1	URIs, IRIs, and the Linked Data Principles . . . . .	51
2.3.2	XML . . . . .	53
2.3.3	RDF . . . . .	55
2.3.4	Ontologies . . . . .	60
2.4	Representing Semiinformal Mathematical Knowledge (State of the Art) . . . . .	62
2.4.1	Semantic vs. Content Markup . . . . .	62
2.4.2	MathML . . . . .	63
2.4.3	OpenMath . . . . .	65
2.4.4	OMDoc . . . . .	67
2.4.5	Defining Notation in MathML, OpenMath, and OMDoc . . . . .	72
2.4.6	MathLang . . . . .	75
2.4.7	TeX, LaTeX, and sTeX . . . . .	75
2.4.8	Languages for Books and Manuals . . . . .	77
2.4.9	Languages for Formalized Mathematics . . . . .	81
2.4.10	RDF with Structural Ontologies . . . . .	82
2.5	Designing an Improved Representation and Exchange Language . . . . .	89
2.5.1	Assessment of the State-of-The-art Languages . . . . .	89
2.5.2	Towards an RDF-extended OMDoc as an Exchange Language . . . . .	94
<b>3</b>	<b>Ontologies for Structures of Mathematical Knowledge</b>	<b>99</b>
3.1	Overview of the Ontologies by Structural Dimension . . . . .	99
3.2	Logical and Functional Structures, and Notation . . . . .	100
3.2.1	Methodology . . . . .	100
3.2.2	The OMDoc Ontology . . . . .	103
3.2.3	The OpenMath Content Dictionary Ontology . . . . .	113
3.3	Rhetorical and Document Structures . . . . .	118
3.3.1	SALT and Related Rhetorical and Document Ontologies . . . . .	118
3.3.2	Mapping OMDoc's Rhetorical Markup to SALT . . . . .	120
3.3.3	Discussion and Future Work: Alignment with OMDoc . . . . .	122
3.4	Metadata . . . . .	122
3.4.1	Directly Reusable Ontologies . . . . .	124
3.4.2	Propagation of Metadata Along Other Structural Dimensions . . . . .	124
3.4.3	Formalizing and Implementing Classification Schemes as Ontologies . . . . .	125
3.4.4	Mapping the OMDoc 1.2 Metadata to RDF . . . . .	125
3.4.5	A New Ontology for OpenMath CD Metadata . . . . .	128



3.5	The Application Environment . . . . .	128
3.5.1	Users and their Interaction with a System . . . . .	128
3.5.2	Application Domains of Mathematics . . . . .	130
3.6	Discussions about Knowledge Items . . . . .	130
3.6.1	A Generic Argumentation Ontology . . . . .	131
3.6.2	Mathematics-specific Extensions: Problems with Mathematical Knowl- edge and their Solution . . . . .	133
3.6.3	Related Work . . . . .	135
3.6.4	Conclusion and Future Work . . . . .	136
3.7	Requirements for Extracting Structures from Semantic Markup to RDF . . . . .	137
3.8	Related Work . . . . .	140
3.8.1	Markup Language Semantics . . . . .	140
3.8.2	Mathematical Domain Ontologies . . . . .	141
3.9	Conclusion and Future Work . . . . .	142
<b>4</b>	<b>Using Mathematical Markup for Implementing and Documenting Expressive Ontologies</b>	<b>145</b>
4.1	Problem and Requirements Statement . . . . .	146
4.2	State of the Art . . . . .	147
4.2.1	Expressivity . . . . .	147
4.2.2	Modularity . . . . .	148
4.2.3	Documentation . . . . .	148
4.3	Implementing and Documenting Heterogeneous Ontologies in OMDoc . . . . .	149
4.3.1	Correspondences between OMDoc and Semantic Web Ontology Languages	150
4.3.2	Knowledge Representation . . . . .	151
4.3.3	Connecting OMDoc and Semantic Web URIs . . . . .	151
4.3.4	Documentation and Presentation . . . . .	154
4.4	Implementation of the OMDoc Ontology . . . . .	154
4.5	Case Study: Reimplementing FOAF in OMDoc . . . . .	155
4.6	Related Work . . . . .	158
4.6.1	Implementing OWL Ontologies in OMDoc . . . . .	158
4.6.2	Heterogeneous Formalization . . . . .	158
4.6.3	Integrated Ontology Documentation . . . . .	158
4.7	Conclusion and Future Work . . . . .	158
<b>5</b>	<b>Multi-Dimensional Metadata Markup</b>	<b>161</b>
5.1	The Metadata Syntax of OMDoc 1.2 (State of the Art) . . . . .	161
5.2	The new OMDoc+RDFa Metadata Framework . . . . .	163
5.2.1	Integrating RDFa into OMDoc . . . . .	163
5.2.2	Recommended RDFa Syntax for Metadata Records . . . . .	164
5.2.3	Semantic Interaction of RDFa and OMDoc Markup . . . . .	164
5.2.4	Rewriting OMDoc 1.2 Metadata in RDFa . . . . .	166
5.3	Related Work . . . . .	168
5.4	Conclusion . . . . .	168

<b>III</b>	<b>Services and their Integration</b>	<b>171</b>
<b>6</b>	<b>Primitive Services for Managing Mathematical Knowledge</b>	<b>175</b>
6.1	Tasks, Scenarios, and Required Primitive Services . . . . .	175
6.1.1	Tasks in Managing, Understanding, and Applying Mathematical Knowledge . . . . .	176
6.1.2	Concrete Workflows and Usage Scenarios . . . . .	177
6.1.3	Primitive Services and the Tasks they Accomplish . . . . .	182
6.2	Editing . . . . .	184
6.2.1	State of the Art, by Type of Interface . . . . .	184
6.2.2	sTeX as a Frontend Input Syntax for OMDoc+OpenMath+RDFa . . . . .	187
6.2.3	Visually Annotating Logical/Functional, Rhetorical, Document, and Pre- sentational Structures in Documents . . . . .	189
6.2.4	Integrating a Formula Editor into a Visual Document Editor . . . . .	191
6.2.5	Accessing and Editing Notation Definitions . . . . .	192
6.2.6	Alternative Visual Editing Interfaces for Metadata: Documents vs. Forms . . . . .	193
6.2.7	Related Work . . . . .	194
6.2.8	Conclusion and Future Work . . . . .	196
6.3	Validating . . . . .	197
6.3.1	Validation and Division of Labor in Collaborative Workflows . . . . .	197
6.3.2	A Stack of Validation Services . . . . .	198
6.3.3	Validating Metadata and Links . . . . .	201
6.3.4	Related Work . . . . .	204
6.3.5	Conclusion . . . . .	205
6.4	Human- and Machine-Comprehensible Publishing . . . . .	205
6.4.1	Publishing Linked Data . . . . .	205
6.4.2	Publishing Documents Comprehensible for Humans and Assistive Services . . . . .	212
6.4.3	Conclusion . . . . .	223
6.5	Information Retrieval . . . . .	223
6.5.1	Searching MathML/OpenMath Objects (State of the Art) . . . . .	224
6.5.2	Querying Structures above the Object Level . . . . .	225
6.5.3	Related Work . . . . .	229
6.5.4	Conclusion . . . . .	230
6.6	Arguing about Problems and their Solutions . . . . .	230
6.6.1	Issue Tracking Systems (State of the Art) . . . . .	231
6.6.2	Recommendations for Using the SIOC Argumentation Module . . . . .	231
6.6.3	Automated Problem-Solving Assistance . . . . .	232
6.6.4	Related Work . . . . .	233
6.6.5	Conclusion and Future Work . . . . .	233
6.7	Conclusion . . . . .	234
<b>7</b>	<b>Integrating Assistive Services into Interactive Documents</b>	<b>239</b>
7.1	State of the Art and Related Work . . . . .	240
7.2	Requirements for Integrating Services into Documents . . . . .	241

7.3	The JOBAD Architecture . . . . .	242
7.3.1	Types of Client Services and their Initialization . . . . .	243
7.3.2	A Generic Proxy for Accessing Remote Web Services . . . . .	244
7.3.3	User Interface Elements . . . . .	244
7.4	In-Document Client Services . . . . .	245
7.4.1	Folding Subterms and Undoing Interactions . . . . .	245
7.4.2	Flexible Elision and Display of Reading Aids . . . . .	246
7.5	Symbol-based Client Services . . . . .	249
7.5.1	Definition and Type Declaration Lookup . . . . .	249
7.5.2	Interactive Notation Switching . . . . .	253
7.5.3	Linked Data Navigation . . . . .	254
7.5.4	Visualizing Rhetorical Structures . . . . .	254
7.6	Expression-based Client Services . . . . .	255
7.6.1	Rendering as a Web Service . . . . .	255
7.6.2	Looking up General Computational Information with Wolfram Alpha . . . . .	255
7.6.3	Unit Conversion – a Case of Domain-specific Computation . . . . .	256
7.7	Conclusion and Future Work . . . . .	257
<b>8</b>	<b>Transparent Translations in Knowledge Bases</b>	<b>261</b>
8.1	Extracting Structures from Semantic Markup . . . . .	262
8.1.1	The Kxextor XML→RDF Framework and its Translation Process . . . . .	263
8.1.2	Extraction Modules for Mathematical Markup . . . . .	266
8.1.3	Reasoning with OWL Ontologies Represented in OMDoc . . . . .	267
8.1.4	Related Work and Discussion . . . . .	268
8.1.5	Conclusion and Future Work . . . . .	271
8.2	Migration to More Expressive Languages . . . . .	272
8.2.1	Translating Less Expressive Languages to OMDoc (State of the Art) . . . . .	273
8.2.2	Translating OWL and Other Ontologies to OMDoc . . . . .	274
8.3	Coping with Different Representation Granularities on Import and Export . . . . .	274
8.3.1	Challenges that Complex Files Pose to Semantic Services . . . . .	275
8.3.2	Splitting and Reassembling Files on Import and Export . . . . .	275
8.3.3	Making Metadata Accessible for Different Services and Editors . . . . .	277
8.3.4	Related Work . . . . .	277
8.4	Recommendations for Running Translations Transparently . . . . .	278
8.5	Conclusion . . . . .	279
<b>9</b>	<b>The Semantic Wiki SWiM – An Integrated Collaboration Environment</b>	<b>281</b>
9.1	Wikis and Semantic Wikis (State of the Art) . . . . .	281
9.1.1	Elementary Wiki Characteristics . . . . .	281
9.1.2	Semantic Wikis . . . . .	282
9.1.3	Mathematical Services in [Semantic] Wikis . . . . .	283
9.1.4	Argumentative Discussions . . . . .	285
9.2	Requirements Analysis and Design Decisions . . . . .	287
9.2.1	Original Reasons for Choosing IkeWiki . . . . .	287

9.2.2	Requirements for Project Management . . . . .	288
9.2.3	Requirements for Maintaining OpenMath CDs . . . . .	288
9.3	Architecture . . . . .	289
9.3.1	The IkeWiki Base and its Extension into SWiM . . . . .	289
9.3.2	Storage Backend . . . . .	292
9.3.3	User Interface . . . . .	295
9.4	How SWiM Supports OpenMath CD Maintenance Workflows . . . . .	296
9.4.1	Quickly Fixing Minor Errors . . . . .	296
9.4.2	Fixing and Verifying Notations . . . . .	299
9.4.3	Peer Review and Preparing Major Revisions by Discussion . . . . .	299
9.5	Related Work . . . . .	301
9.5.1	PlatΩ, Lurch, jEditOQMath, and WIRIS: Editors with Integrated Validation . . . . .	301
9.5.2	PlanetMath/Noösphere: Automatic Linking and other Services . . . . .	301
9.5.3	OpenMath CD Manager: Maintaining Fixed Structures . . . . .	302
9.5.4	Connexions/Rhaptos: Structured Semantic Markup Editing . . . . .	302
9.5.5	ProofWiki, Mizar Wiki, and Logiweb: Formalized Mathematics . . . . .	302
9.5.6	ASciencePad and Mathematica-users.org: Computation and Graphing . . . . .	303
9.5.7	Lekapidia, Cicero, and SharedHCONE: Wikis with Argumentative Discussions . . . . .	303
9.6	Conclusion and Future Work . . . . .	304
9.6.1	Mathematical Services Integrated into a Collaboration Environment . . . . .	304
9.6.2	Improvements over the State of the Art of [Semantic] Wikis . . . . .	306
9.6.3	Incubator for New Services and Integration Approaches . . . . .	306
9.6.4	Future Work . . . . .	307
<b>10</b>	<b>Usability Evaluation of an Integrated Environment for Maintaining Semiformal Collections</b> . . . . .	<b>313</b>
10.1	Preparation and Setup . . . . .	314
10.2	Evaluation Hypotheses and Method . . . . .	314
10.2.1	The Interaction Triptych Model and a Definition of Usability . . . . .	314
10.2.2	Usability Evaluations of Related Systems . . . . .	315
10.2.3	Hypotheses about the Usability of the OpenMath Wiki . . . . .	317
10.2.4	Techniques Applied for Testing the Hypotheses . . . . .	318
10.3	Quantitative Content Analysis of Argumentative Discussions . . . . .	319
10.4	Community Survey . . . . .	320
10.4.1	OpenMath CD Experience and Practices . . . . .	320
10.4.2	Utility of CD Maintenance Workflow Supports . . . . .	321
10.4.3	Feedback on Other Wiki Features, and Wishes . . . . .	322
10.4.4	Conclusion on Utility . . . . .	323
10.5	Supervised Usability Experiments with Test Users . . . . .	323
10.5.1	Setting and Evaluation Method . . . . .	323
10.5.2	Overall Figures . . . . .	325
10.5.3	Quickly Fixing Minor Errors . . . . .	327
10.5.4	Fixing and Verifying Notations . . . . .	329

10.5.5	Peer Review and Preparing Major Revisions by Discussion . . . . .	332
10.5.6	Feedback on Incoherent Integration . . . . .	335
10.5.7	Discussion of the Evaluation Setup and Method . . . . .	336
10.6	Evaluation Results and their Interpretation . . . . .	337
10.6.1	Usability of the OpenMath Wiki . . . . .	337
10.6.2	The Challenge of Integrating Heterogeneous Services . . . . .	339
10.6.3	Semantically Transparent User Interfaces for Integrated Environments . . . . .	339
10.7	Conclusion and Future Work . . . . .	341
10.7.1	Coverage of the Evaluation . . . . .	341
10.7.2	Evaluating Collaboration and Other Axes and Components . . . . .	342
10.7.3	Customizable and Documented System Ontologies as a Means of Appropriating Environments and Achieving Semantic Transparency . . . . .	343
<b>IV</b>	<b>Conclusion and Future Work</b>	<b>345</b>
<b>11</b>	<b>Conclusion and Future Work</b>	<b>347</b>
11.1	Retrospective Summary . . . . .	347
11.2	Evaluation Against the Original Research Questions . . . . .	349
11.3	Future Directions for e-Science . . . . .	351
11.3.1	The Planetary “eMath 3.0” Environment . . . . .	352
11.3.2	Better Support for Peer Review and Integrated Publishing . . . . .	353
11.3.3	Interactive Workbenches for Scientific Collaboration . . . . .	353
11.3.4	Integrating Mathematics into the Web of Data . . . . .	354
11.4	Conclusion . . . . .	356
<b>V</b>	<b>Appendix</b>	<b>357</b>
<b>A</b>	<b>Namespace Prefixes</b>	<b>359</b>
<b>B</b>	<b>Ontologies</b>	<b>363</b>
B.1	OMDoc . . . . .	363
B.2	OpenMath CDs . . . . .	372
B.3	Mathematics-specific Issue and Solution Types . . . . .	376
<b>C</b>	<b>Algorithm and Implementation Details</b>	<b>379</b>
C.1	Primitive Services . . . . .	379
C.1.1	TinyMCE+Sentido, a Visual Editor for Semantic Markup, Content Markup Formulæ, and Metadata . . . . .	379
C.1.2	JOMDoc, a Semantics-Preserving Renderer . . . . .	380
C.1.3	Querying Multiple Structural Dimensions with SPARQL . . . . .	383
C.1.4	Automated Problem-Solving Assistance . . . . .	384
C.2	JOBAD, a Library of Assistive Services for Interactive Documents . . . . .	386
C.2.1	A Generic Proxy for Accessing Remote Web Services . . . . .	386

C.2.2	In-Document Services . . . . .	387
C.2.3	Symbol-based Services . . . . .	388
C.2.4	Expression-based Services . . . . .	389
C.2.5	Unit Conversion . . . . .	390
C.3	Transparent Translations in Knowledge Bases . . . . .	390
C.3.1	The Krextor XML→RDF Translation Framework . . . . .	390
C.3.2	A Translator of OWL Ontologies from RDF to OMDoc . . . . .	394
C.3.3	Translations Between Different Representation Granularities . . . . .	395
C.4	The Semantic Wiki SWiM . . . . .	396
C.4.1	IkeWiki’s Ontology and Reasoning Support . . . . .	396
C.4.2	Storage Backend . . . . .	397
C.4.3	User Interface . . . . .	398
<b>D</b>	<b>Survey Results</b>	<b>401</b>
D.1	Reporting and Solving Issues with Mathematical Knowledge Items . . . . .	401
D.2	OpenMath Wiki Evaluation . . . . .	407
D.2.1	General OpenMath Questions . . . . .	407
D.2.2	Minor Edits . . . . .	411
D.2.3	Discussing Major Revisions . . . . .	412
D.2.4	Editing and Verifying Notations . . . . .	414
D.2.5	Other Wiki Features . . . . .	416
D.3	OpenMath Wiki Usability Experiments . . . . .	418
	<b>Bibliography</b>	<b>451</b>

## Abstract

This thesis presents a collection of methods and technologies that enable building a collaboration infrastructure for managing mathematical knowledge in a way that makes it comprehensible, reusable, and applicable. Working mathematicians have already embraced Social Web applications such as blogs for communication and collaboration, but these neither make knowledge accessible to automated agents for, e.g., verification or computation, nor to specific audiences such as students having less background knowledge than the original authors. The key challenge addressed in this thesis is effectively supporting collaborative mathematical knowledge management (MKM) workflows by making the knowledge comprehensible to a wide range of services, while aiming at an entry barrier that, for a domain expert, is not disproportionately higher than that of successful Social Web sites.

As the building blocks for the envisaged collaboration environment were not available in a way that would merely have required putting them together, the main focus of this thesis is “under the hood”, i.e. in preparing these building blocks. To get an idea of the building blocks, consider the workflow of writing a research paper: That involves formalizing the original idea from one’s mind into a structured document, searching existing knowledge to build on, validating the formal structure, presenting the content in a comprehensible way, and submitting it for review. Reviewers would look up background information in cited publications, and point out problems with the paper and the formal concepts it introduces. Previous research on MKM has produced services that effectively support the primitive tasks that the overall workflow is composed of. However, these services take different perspectives on mathematical knowledge and speak different languages, which restricts their integration.

Our integration approach starts with opening up a wider audience for existing expressive mathematical knowledge representation languages – in the first step an “audience” of machines, which then make the mathematical knowledge accessible to their human end users. We improve the interoperability of different mathematical knowledge representations with each other, and with sources of non-mathematical knowledge about applications, projects, and people, by putting them on a common Semantic Web foundation that combines the document-oriented view of mathematical authoring and publishing with the network-oriented view of the growing Web of Data and Web-based information retrieval.

We address service integration from two perspectives: enriching published documents by embedding assistive services, and integrating translations between different knowledge representations transparently into a knowledge base. Ultimately, we combine both perspectives into a semantic wiki environment for collaboratively producing and consuming mathematical knowledge. This serves as a prototype for evaluating the effectivity of supporting realistic workflows following our integration approach. An evaluation of the wiki’s usability in the setting of maintaining a widely used collection of semiformal mathematical knowledge helps to understand the remaining challenges in making environments that integrate heterogeneous services for different knowledge representations learnable, effective, useful, and satisfying to use.

Finally, we discuss future directions in combining the building blocks obtained in this work towards e-science on the Web: supporting scientists in collaboratively gaining new knowledge, and steps towards contributing existing collections of mathematical knowledge to the Web of Data.

## Abstract (deutsch)

Diese Arbeit präsentiert eine Sammlung von Methoden und Technologien, die den Aufbau einer Kollaborations-Infrastruktur zur Verwaltung mathematischen Wissens dahingehend ermöglichen, dass dieses verständlich, wiederverwendbar und anwendbar wird. Mathematiker kommunizieren und kollaborieren bereits mithilfe von sozialen Web-Anwendungen wie Blogs; diese machen allerdings Wissen weder Agenten zugänglich – z. B. für Verifikation oder Berechnungen – noch speziellen Zielgruppen, wie z. B. Studenten, denen das Hintergrundwissen der ursprünglichen Autoren fehlt. Diese Arbeit geht die Herausforderung an, kollaborative MKM-Arbeitsabläufe (MKM = Mathematical Knowledge Management) effektiv zu unterstützen, indem das Wissen einer großen Zahl von Diensten verständlich gemacht wird. Dabei soll die Einstiegsschwelle (für Fachleute) nicht unverhältnismäßig höher liegen als bei erfolgreichen sozialen Websites.

Da die Bausteine für die angestrebte Kollaborationsumgebung nicht in einer Form verfügbar waren, dass man sie lediglich hätte zusammensetzen müssen, liegt der Schwerpunkt dieser Arbeit „unter der Haube“, und zwar auf der Bereitstellung dieser Bausteine. Zur Veranschaulichung sei der Arbeitsablauf betrachtet, einen wissenschaftlichen Artikel zu schreiben: Dies erfordert, die ursprüngliche Idee aus dem Kopf in ein strukturiertes Dokument zu formalisieren, vorhandenes Wissen zu suchen, um darauf aufzubauen, die formale Struktur zu validieren, den Inhalt verständlich zu präsentieren, und ihn zur Begutachtung einzureichen. Gutachter schlagen Hintergrundwissen in zitierten Publikationen nach und weisen auf Probleme des Artikels und der darin eingeführten formalen Konzepte hin. Die bisherige MKM -Forschung hat Dienste hervorgebracht, die effektiv die elementaren Aufgaben unterstützen, aus denen der Arbeitsablauf insgesamt besteht. Diese Dienste betrachten mathematisches Wissen allerdings aus unterschiedlichen Blickwinkeln und sprechen unterschiedliche Sprachen, was ihre Integration behindert.

Unser Integrationsansatz beginnt damit, vorhandenen ausdrucksstarken Repräsentationssprachen für mathematisches Wissen eine breitere Zielgruppe zu erschließen – im ersten Schritt eine „Zielgruppe“ von Maschinen, die mathematisches Wissen dann ihren Endanwendern zugänglich machen. Wir verbessern die Interoperabilität unterschiedlicher mathematischer Wissensrepräsentationen untereinander sowie mit nicht-mathematischen Wissensquellen über Anwendungen, Projekte und Menschen, indem wir sie auf ein gemeinsames Semantic-Web-Fundament stellen, das die dokumentenorientierte Sicht des mathematischen Schreibens und Publizierens mit der netzorientierten Sicht des wachsenden Web of Data und webbasierten Information Retrievals kombiniert.

Wir betrachten Dienstintegration von zwei Seiten: Bereicherung publizierter Dokumente durch Einbettung assistiver Dienste, und transparente Integration von Übersetzungen zwischen unterschiedlichen Wissensrepräsentationen in Wissensdatenbanken. Schließlich kombinieren wir beide Richtungen in einer semantischen Wiki-Umgebung zum kollaborativen „Produzieren und Konsumieren“ mathematischen Wissens. Dieser Prototyp dient zur Evaluation der Frage, ob realistische Arbeitsabläufe durch unseren Integrationsansatz effektiv unterstützt werden können. Eine Evaluation der Usability des Wikis zur Pflege einer weithin verwendeten Sammlung semiformalen mathematischen Wissens hilft zu verstehen, welche Herausforderungen es mit sich bringt, integrierte Umgebungen mit heterogenen Diensten für unterschiedliche Wissensrepräsentationen erlernbar, effektiv, nützlich und zufriedenstellend für Anwender zu machen.



Abschließend diskutieren wir, wie zukünftige Forschung die in dieser Arbeit gewonnenen Bausteine in Richtung e-Science im Web kombinieren kann – um Wissenschaftler beim kollaborativen Wissensgewinn zu unterstützen und vorhandene Sammlungen mathematischen Wissens zum Web of Data beizusteuern.

## Acknowledgments

*It takes a village to raise a child.*

—African Proverb

While formal acknowledgments to collaborators are placed at the end of each chapter, this is the place to thank the “village” of people that helped me to raise this very “child” of thesis, and helped the “father” to gain much deeper insights into the craft of raising such children.

I would like to thank my supervisors – my “doctor father” MICHAEL KOHLHASE (a pity that there is no suitable English translation of this term that fits him so well) as well as PETER BAUMANN and STEFAN DECKER – for teaching me to view my research topic from quite different perspectives, comprising representations, operations, applications, improving over shortcomings of the past as well as creating a new future.

I am furthermore grateful to the following persons – roughly given in alphabetical order, as any attempt at a total order by impact would wrong someone – for broadening my mind by giving feedback on my ideas or pointing out new directions: ANDREA ASPERTI, SERGE AUTEXIER, JOHN BATEMAN, ULDIS BOJĀRS, JOHN BRESLIN, CLAIRE BROWNE, TILMAN DE BRUIN, DAVID CARLISLE, XIAOYU CHEN, STÉPHANE CORLOSQUET, JOE CORNELI, JOHN CUNLIFFE, RICHARD CYGANIAK, JAMES DAVENPORT, ALEXANDER GARCÍA CASTRO, ALBERTO GONZÁLEZ PALOMO, TUDOR GROZA, SIEGFRIED HANDSCHUH, TUUKKA HASTRUP, MANFRED HAUSWIRTH, SAJJAD HUSSAIN, JAN WILLEM KNOPPER, my current and former colleagues in the KWARC research group at Jacobs University Bremen – including ȘTEFAN ANCA, MATTHIAS BRÖCHELER, CĂTĂLIN DAVID, ȘTEFANIA DUMBRĂVĂ, ANCA DUMITRACHE, JANA GIČEVA, DEYAN GINEV, SÖNKE HOLSTEN, FULYA HOROZAL, CONSTANTIN JUCOVSCI, ANDREA KOHLHASE, CHRISTINE MÜLLER, NORMEN MÜLLER, IMMANUEL NORMANN, FLORIAN RABE, GORDAN RISTOVSKI, HEINRICH STAMERJOHANN, JAKOB ÜCKER, and VYACHESLAV ZHOLUDEV –, TIMOTHY LEBE, PAUL LIBBRECHT DIMITAR MIŠEV, KNUD MÖLLER, KRYSTIAN SAMP, SEBASTIAN SCHAFFERT and the member of the KiWi project – particularly including SZABY GRÜNWALD, JAKUB KOTOWSKI, THOMAS KURZ, MIHAI RADULESCU, MAREK SCHMIDT, STEPHANIE STROKA, ROLF SINT, and KLARA WEIAND – THOMAS SCHANDL, MARVIN SCHILLER, ALAN SEXTON, CHRISTOPH TEMPICH, JOSEF URBAN, MAX VÖLKEL, DENNY VRANDEČIĆ, MARC WAGNER, and VERA ZEGERS and the participants of her seminar „Großprojekt Doktorarbeit“ (“large-scale project Ph.D. thesis”).

I have received (not only!) financial support from the German National Academic Foundation (Studienstiftung des deutschen Volkes), Jacobs University Bremen, as well as partly from the German Research Foundation (DFG) and the German Academic Exchange Service (DAAD), whom I would like to thank as well.

After this “scientific list”, very special thanks to those who faithfully kept reminding me that there is not only a life *after* the Ph.D. thesis (as MICHAEL rightly uses to say), but also a life *during*

the Ph.D. thesis – these include (alphabetically) J-CAPPELLA, SARAH-ELISA NEES, CHRISTOPH REITNAUER, SONJA WOHLAIB, and, above all (intentionally not listed alphabetically), KERSTIN BEVER.

# **Part I**

## **Introduction**



# Web Collaboration on Mathematical Knowledge

*In the relation between mathematics and computing science,  
the latter has been for many years at the receiving end,  
and I have often asked myself  
if, when, and how computing would ever be able to repay its debt.*

—EDSGER W. DIJKSTRA [Dij86]

The goal of my work is to support collaboration on mathematical knowledge in a way that makes it comprehensible, reusable, and applicable – for mathematicians, scientists, engineers, and students (cf. [section 1.1](#)). Nowadays, the Web<sup>1</sup> is a preferred medium for scientific communication. Web 2.0 and semantic web technologies, two complementary approaches to improve collaboration and knowledge reuse on the Web, are already being used in scientific applications ([section 1.3](#)). Working mathematicians have started to adopt web 2.0 applications, whereas research on mathematical knowledge management (MKM) has so far largely failed to successfully embrace the Semantic Web ([section 1.4](#)). I argue that there is now a good opportunity to try applying semantic web technologies to mathematical collaboration once more, and, conversely, outline how other semantic web applications can benefit from the availability of mathematical knowledge ([section 1.5](#)). While mainly motivated from a mathematical perspective, the envisaged infrastructure is not restricted to mathematics, but addresses all STEM fields (science, technology, engineering, mathematics). This thesis generally speaks of mathematics but points out interrelations with other STEM fields wherever appropriate. Previous MKM and semantic web research has neither sufficiently addressed the challenge of exchanging STEM knowledge – in all of its different degrees of formality – across knowledge bases, nor the flexible composition of intelligent services to the end of increasing the comprehensibility of STEM documents – be it articles, textbooks, manuals, or drafts ([section 1.6](#)). These challenges, and the challenge of collaboratively maintaining such reusable and comprehensible representations of STEM knowledge, are addressed in this thesis.

---

<sup>1</sup>In order to reduce eye strain, I only capitalize this term, as well as the “brand names” “Web 2.0” and “Semantic Web”, when they denote the Web as a whole, but not when they are in an adjective position, as in “semantic web services”.

## 1.1 Current Practices of “Doing Mathematics”

*Outsiders see mathematics as a cold, formal, logical, mechanical, monolithic process of sheer intellection;  
we argue that insofar as it is successful,  
mathematics is a social, informal, intuitive, organic, human process, a community project.*

—RICHARD A. DE MILLO, RICHARD J. LIPTON, ALAN J. PERLIS [DMLP79]

Science requires communication and collaboration; mathematics is no exception. There is still the widespread perception of a mathematician sitting alone at his<sup>2</sup> desk and developing ideas with pen and paper. While this working method certainly constitutes a *part* of mathematical research, communication and collaboration have always been important. On a small scale, mathematicians often hold informal face to face meetings to exchange ideas and discuss problems they are working on, as BETTINA HEINTZ describes in her sociological study of mathematics [Heioo]. The long collaboration between G. H. HARDY and JOHN E. LITTLEWOOD in the early 20<sup>th</sup> century is another historically well documented case: Even when they could have met face to face, they preferred to communicate in writing, following four collaboration “axioms” they had established [BD78]. Large-scale collaboration is still less common in mathematics than in the natural sciences; however, there have been notable examples in recent history, such as the classification of finite simple groups, which was pursued by around a hundred international mathematicians over a period of several decades and led to the publication of about 500 articles of 15,000 pages altogether [Heioo, 186–187]. Overall, an “industrialization” of mathematical research has been observed, exhibiting patterns such as big teams of authors, instant communication, more fluid collaboration, decentral modes of publication and knowledge authentication, and the usage of big computer systems [BS05; Cario]; ANDREA ASPERTI et al. similarly argued that “*mathematics is destined to assimilate some practices of software development*” [AGNo9].

The final result of any research effort in mathematics is a *proof*, whose obvious role is the verification of a newly established piece of mathematical knowledge. Besides establishing truth, a proof serves as the preferred medium of *communicating* this truth to the mathematical community (cf. [Heioo, chapter 6.2] and [DMLP79]): By social convention, the community does not accept a new finding without a proof. Moreover, “*a new proof of a theorem [already known to be true] can provide crucial insights*” [GNo9]. A proof is expected to argue rigorously; however, it is disputed what exactly constitutes a rigorous proof [Ker10]. The language for communicating proofs is highly stylized, using symbolic notation embedded into fixed natural language phrases. A mathematician who wants the reader to understand a proof has to ensure common background knowledge. In writing down a concrete proof, this can be achieved by providing explicit back-references to established results that have been applied in carrying out the proof, and by using an instructive symbolic notation that appeals to the reader’s intuition.<sup>3</sup> THOMAS HALES’s initial “proof” of the Kepler conjecture<sup>4</sup> is a notable (counter-)example. It was not accepted by the mathematical community, as computer code for solving thousands of linear programs, together

<sup>2</sup>Whenever the gender of a person is not determined, I generally use the male form. In the context of this thesis, this is not intended as a political statement.

<sup>3</sup>On intuitive notation, compare [Heioo, 163–168], [DH81, 122–125], and [P6173].

<sup>4</sup>This conjecture, posed in 1611, states that the density of a packing of unit spheres in 3 dimensions is at most  $\pi/(3\sqrt{2})$ . This reflects the intuitive observation that the way, in which, e.g., oranges in a market booth are stacked, is optimal. However, it turned out exceedingly complex to prove.

with the respective input and output data, constituted a core part of it, and the reviewers found that unusually hard to validate. Therefore, this proof is now being completely revised in a collaborative effort of several mathematicians and computer scientists in the Flyspeck project [HM+; HHM+10].

However, the highly conventionalized and streamlined language of mathematical proofs and its ways of referring to required background knowledge is not necessarily sufficient for fostering intuitive comprehension and mutual understanding; it only becomes an efficient means of communication once a certain level of “*mental infrastructure*” (WILLIAM P. THURSTON in [Thu94]) not only of factual background knowledge but also of methodological skills (“*ways of thinking*” [Thu94]) has been established. Sufficient shared mental infrastructure exists within the highly specialized subfields of mathematics, but much less so outside of these subfields or even outside of mathematics [Thu94]; THURSTON considers informal events, such as talks or seminars, useful for building it (as cited in [Heioo, pp. 224-226]). In particular, the deductive style, in which mathematical findings are commonly presented (from axioms defining structures to propositions stating properties of these structures to the proofs of these propositions), does not reveal *how* these insights were gained, but the latter is of particular importance when teaching mathematics (cf. [DH81, chapter 6]). The actual discovery of mathematical knowledge, the actual methodology of mathematical research, is much more of an experimental process driven by intuition and heuristics, as described by GEORGE PÓLYA [Pól73] and IMRE LAKATOS [Lak76]. PÓLYA describes the solution of a given [mathematical] problem as a four-step procedure of (i) understanding the problem, (ii) devising a plan, (iii) carrying out the plan, and (iv) looking back. Particularly addressing *step (ii)*, he tries to give a systematic account of common heuristics, such as induction from examples, analogy, and generalization. The final *step (iv)* is concerned with checking the result, coming up with a different – possibly easier – derivation, and reusing the result for other problems. PÓLYA’s method primarily addresses a single mathematician – and partly a teacher–student dialog –, whereas LAKATOS studies dialogs with multiple participants [Lak76] – not necessarily real dialogs in a face-to-face meeting, but he also uses the dialog as a device to present “*the transformations that several famous theorems underwent from initial conception to general acceptance*” [DMLP79]. His technique starts with an initial conjecture, for which an informal<sup>5</sup> “proof” (rather: a draft of a justification) is provided. This informal proof usually turns out to be wrong or incomplete, provoking counterexamples that refute it and thus require reworking either a step of the proof, or even restating the initial conjecture, i.e. adapting it to the proof.<sup>6</sup>

Nowadays, there is computer assistance for several steps of the workflows described by PÓLYA and LAKATOS. For example, the above-mentioned Flyspeck project not only aims at verifying the computer code involved, but also formalizing the large “traditional” parts of the proof in a way that can be checked by a computer; this is estimated to take 20 person-years [HHM+10]. Note, however, that a proof that has been formalized to such an extent that a computer can check

<sup>5</sup>In this chapter, “informal”, “formalized”, and related terms are mostly used intuitively. Their meaning in the context of this thesis will be fixed in [section 2.1.1.2](#).

<sup>6</sup>RICHARD A. DE MILLO et al. and ANDREA ASPERTI et al. summarize more recent cases of LAKATOSIAN dialogs in the large, i.e. refutations of wrong proofs that had already been published years ago [AGN09; DMLP79]; ASPERTI et al. emphasize how that has contributed to mathematical progress [AGN09].

each step is usually no longer comprehensible to a human reader<sup>7</sup>; the machine has to make it human-comprehensible again by laborious proof explanation techniques (see, e.g., [Fie01]). Besides the complete development and validation of a proof using a proof assistant, HEINTZ mentions the utility of the computer as a tool for gaining intuition by experiment and generating counterexamples [Heio0, p. 154].

## 1.2 Enabling Management, Understanding, and Application of Mathematical Knowledge

Considering the observations about mathematical practices summarized above, the goal of this thesis is to provide methods and technologies that enable the acquisition of new mathematical knowledge, the formalization and organization of existing knowledge, and that supports mathematicians in expanding the “mental infrastructure” required for understanding, reusing, and applying this knowledge. This process will be driven by human users but aided by a computer system, which provides a collaboration interface on top of a knowledge repository. The ultimate work environment, not entirely to be built in the course of this thesis, but not too utopian by composing the building blocks to contributed by this thesis will support working mathematicians in elaborating an initial sketch into a final version that is computer-verifiable and both comprehensible to automated agents and to human readers. It will achieve comprehensibility of the knowledge by supporting users in choosing an intuitive notation for formal concepts, and documenting conversations about problems and solutions, as intended by PÓLYA and LAKATOS, so that the complete process of discovery is retraceable. Thus, the envisaged environment will not only aid research, but also problem solving, education, and application.

That is, my research towards building this collaboration environment broadly addresses *mathematical knowledge management* (MKM). The interdisciplinary MKM community comprises computer scientists, computer-savvy mathematicians, and digital library researchers, whose objective is “to develop new and better ways of managing mathematical knowledge using sophisticated software tools” [Far04]<sup>8</sup>, or, more specifically, “to serve (i) mathematicians, scientists, and engineers who produce and use mathematical knowledge; (ii) educators and students who teach and learn mathematics; (iii) publishers who offer mathematical textbooks and disseminate new mathematical results; and (iv) librarians and mathematicians who catalog and organize mathematical knowledge” [Far04]<sup>9</sup>.

Beyond pure mathematics, applications of mathematics are within the focus. Science, technology, and engineering share mathematics as a common foundation and consequently use the same rigorous style of argumentation – albeit establishing evidence by empirical observations instead of formal proofs – and the same symbolic formula language. The process of understanding results

---

<sup>7</sup> ALFRED NORTH WHITEHEAD and BERTRAND RUSSELL needed over 300 pages to derive  $1 + 1 = 2$  from a well-defined set of axioms and inference rules in symbolic logic [WR10]. “Going back to a logic level proof is typically like being dragged on a level on which we do not see the wood for the trees.” [Ker10]

<sup>8</sup> This notion of “knowledge management” is wider than its traditional definition as “a range of practices used in an organisation to identify, create, represent, distribute and enable adoption of insights and experiences. Such insights and experiences comprise knowledge, either embodied in individuals or embedded in organisational processes or practice.” [Wik09b]

<sup>9</sup> enumeration added by the author



is similar, too. For example, a software engineer can hardly understand a piece of software from its source code and the brief embedded documentation alone – i.e. the counterpart to rigorous mathematical notation –, but will usually have to consult external manuals – compare mathematical textbooks – and records of developers’ communication about the code, such as e-mail discussions and bug reports – here, think of transcribed dialogs in the manner of LAKATOS.

The following sections review how much contemporary web applications have already contributed to this goal and establish an agenda of what still needs to be done, and a plan of how this will be achieved.

## 1.3 Web 2.0 and Semantic Web in Science

*The internet offers us the first major opportunity  
to improve this collective long-term memory[i.e. the “scientific journal system”],  
and to create a collective short-term working memory,  
a conversational commons for the rapid collaborative development of ideas.  
The process of scientific discovery – how we do science –  
will change more over the next 20 years than in the past 300 years.*

—MICHAEL NIELSEN [Nieo8]

Nowadays, the World Wide Web (WWW) is a preferred medium for publishing documents and communicating, also in science. This section reviews its state of the art. [Section 1.4](#) reviews web applications that already support the mathematical practices described in [section 1.1](#).

### 1.3.1 The Original World Wide Web

In 1990, the WWW was created as a hypertext architecture to support scientists at CERN in their day to day work by offering an infrastructure for sharing documents and information about experiments, facilities, and systems [BL90]. The Web was originally envisaged as a *read-write information space* with *explicit information about the type of interrelation that two linked documents have*, such as “the manual *M* describes in detail how the system *S* works”. These two aspects were not initially taken up in practice. When the Web went commercial in the mid-1990s, input forms (e.g. for ordering a product from a shop) offered a limited degree of interaction with sites that were otherwise read-only for their visitors. Information about the content of a web page and its links to other pages was usually given in a way that the brain of a (sighted) human could understand, but not in a well-structured way suitable for automated agents. These two problems were addressed by two complementary innovations called *Web 2.0* and *Semantic Web*.

### 1.3.2 Web 2.0 – Social Networks and User-Generated Content

The Web 2.0, addressing the lack of interactivity of the Web 1.0<sup>10</sup>, has given birth to social websites for collaboratively creating documents, for sharing documents and multimedia artifacts, and for commenting on such artifacts or on products and recommending them to friends; the common

<sup>10</sup>In retrospect, this term is used for the read-only Web with limited interaction that only addressed humans. Conversely, the term “Web 3.0” is sometimes used for the emerging combination of Web 2.0 and Semantic Web.

term for all that is *user-generated content* [O’Ro5; AKT+o8]. Users are often encouraged to share the content they generated under permissive licenses such as Creative Commons [Cre], which allow other users to legally share, remix, and reuse it. Besides manual remixing, there are *mashups* that combine, aggregate and transform data and web services into new lightweight interactive applications [AKT+o8].

Documents on the Web 2.0 are often created in *wikis* (cf. section 9.1) and *blogs*, two kinds of lightweight content management systems (CMS). The main difference is that a wiki article is usually authored and evolved collaboratively and covers one topic, linked to related topics, whereas a blog post is written by a single author on a single date, and others can comment on it. While a sharp distinction is hard to make, wikis and blogs differ from traditional CMS in the easier creation of new content<sup>11</sup> and their focus on content rather than on a high-end layout. Wikis, in particular, have a flat hierarchy of user permissions – in many wikis, everybody is allowed to edit articles – and make it easier to link articles, as their URLs usually correspond to their titles.

### 1.3.3 Semantic Web and Linked Data

The objective of the Semantic Web effort, addressing the lack of machine-comprehensibility of the Web 1.0, is to enrich the Web with machine-readable data enabling intelligent retrieval and inference services [BLHL01]. Informal web content, such as HTML documents, is annotated with terms whose meaning has been defined in machine-comprehensible vocabularies. (The latter are also called *ontologies* when they are more formal, e.g. using a description logic foundation.) Documents – called “information resources” in the WWW terminology [JWo4] –, as well as real-world objects – called “non-information resources” – are addressed globally and uniquely by URIs/IRIs (uniform/internationalized resource identifiers, cf. section 2.3.1). Semantic web services and automated agents access knowledge bases and utilize web services from various places on the Web, combining knowledge from different sources, drawing their own inferences, and ultimately delivering added value to users. Regarding this issue, there are many technical approaches, ranging from heavyweight architectures for finding self-describing web services that accomplish parts of the job to be done and orchestrating them, to lightweight mashup-like solutions drawing on *linked data*.

The term “linked data” denotes a set of best practices for publishing data on the Semantic Web, then also called “Web of Data”. Section 2.3.1 lists the principles in detail; in summary, they state that, whenever something is identified by a URI, machine-comprehensible information and links to further information should be provided right under that URI. These principles are widely considered to have made the Semantic Web vision work practically: They make basic information retrievable without complicated lookup mechanisms, and they respect the decentral nature of the Web. Moreover, linked data are usually published using vocabularies with a lightweight semantics, which enables scalable reasoning across datasets. A lot of providers have already published their data according to these principles and interlinked them with other datasets (cf. figure 1.1). The hub in this big picture is DBpedia [Dbp], a huge collection of general-purpose data extracted from the

<sup>11</sup>WARD CUNNINGHAM, who invented the first wiki in 1994, characterized it as “the simplest online database that could possibly work” [Cun+o2]. Long before WYSIWYG HTML editors became widespread, wikis and blogs featured simple text input syntaxes corresponding to subsets of HTML (cf. section 6.2.1 on editors).

<sup>12</sup>More recent versions without colors are available from the cited URL.

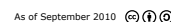


Figure 1.1: Linked Open Datasets as of September 2010<sup>12</sup> [CJ09]

web 2.0 encyclopedia Wikipedia (cf. [section 1.4.2](#)) and made available as RDF. Data from specific domains, such as scientific publications (green), biomedicine (pink), social networks (orange), multimedia (dark blue), geodata (yellow) and government statistics (cyan) have also been published as linked open data. Note that linked data do not *have* to be open<sup>13</sup>, but making datasets open of course helps to interlink and reuse knowledge; therefore, the open datasets have so far been the most visible and most widely used instances of linked data. Applications include browsers, which allow users to traverse the Web of Data and discover unknown connections (see, e.g., [[HLS10](#)]), semantic search engines and indexes, which enable a more accurate information retrieval than keyword-based engines, as well as mashups that aggregate linked data from distributed sources and expose them via a coherent user interface with less development effort than traditional web 2.0 mashups (see, e.g., [[HMF09](#)] for an interactive map of database researchers and their publications, filterable by research topics, or eZaragoza [[TAFB+10](#)], which presents information about the city of Zaragoza to tourists).

### 1.3.4 Combining Web 2.0 and Semantic Web – Benefits and Challenges

Initially, the Web 2.0 and the Semantic Web were developed independently from each other, but, recently, they are more and more being combined (cf. [AKT+08]). Semantic web technology has the potential to provide web 2.0 applications with better information retrieval and more intelligent

<sup>13</sup>An example for using them in an enterprise intranet is given in [section 1.5.2](#).

services. Conversely, the hard job of representing knowledge in such an explicit way that semantic web services can utilize it can be facilitated by massive bottom-up collaboration, applying the lightweight tools that the Web 2.0 has brought forth. This combination, outlined, e.g., by ANUPRIYA ANKOLEKAR et al. [AKT+08], has partly made it from academia into real life by now. For example, Semantic MediaWiki (cf. [section 9.1.2](#)), which started as an academic prototype in 2005, has been bundled with a number of extensions into SMW+ by the Ontoprise company, which markets it as an enterprise intranet system, e.g. for project management [Smw]. Freebase [Fre], an open content repository of community-contributed structured data, which is also a major constituent of the linked open data cloud shown in [figure 1.1](#), was acquired by Google in July 2010. Freebase data have been used in a number of semantic mashups; other examples for semantic web mashups have been mentioned in the previous section.

On the other hand, ANKOLEKAR et al. pointed out a number of challenges that still exist today: (i) the need for more expressive ontologies for adequately representing complex knowledge, (ii) the ongoing challenge to balance expressivity and scalability when working with large amounts of data, (iii) the shortage of intuitive user interfaces for semantically rich applications, (iv) establishing trustworthiness of data and their creators, (v) and knowledge mapping and integration. This thesis primarily addresses challenges for managing *mathematical knowledge* with web 2.0 and semantic web technologies, which will be specified in [section 1.6](#), but part of its results are sufficiently general to contribute to the solution of challenges (i), (iii) and (v).

## 1.4 Mathematics on the Web – State of the Art and Challenges

*Without a need for laboratories or expensive apparatus,  
mathematics would seem particularly suited among the sciences to open online collaboration*

—MICHAEL J. BARANY [[Bar10](#)]

A lot of mathematical knowledge has been created and published on the Web, both by practitioners doing mathematical research, education, or applications, and in research projects that investigated the applicability of web technologies to mathematics. This section reviews the state of the art of mathematics on the Web, focusing on how well the mathematical practices described in [section 1.1](#) are already covered, and to what extent the technologies reviewed in [section 1.3](#) are applied. As many mathematical applications on the Web are not yet using the more modern web 2.0 and semantic web technologies, I first review traditional web 1.0 applications, which are still widely in use. Web 2.0 applications, enabling better communication and collaboration, are becoming more and more commonplace also among mathematicians, whereas semantic web technologies have not yet achieved a breakthrough in the mathematical domain, despite their potential to considerably improve retrieval and exchange of information. [Section 1.5](#) points out the still existing challenges for mathematics on the Web and argues why the opportunity is now to make another attempt to address them.

### 1.4.1 Web 1.0 – Publication Databases, Formalized Libraries, Educational Content

Even the solitary pen-and-paper mathematician nowadays uses digital libraries in order to look up literature and to keep informed about recent research. Zentralblatt MATH [Zbl] and MathSciNet [Ame], the online version of the Mathematical Reviews, are the largest service that provides reviews and abstracts for publications in pure and applied mathematics. Nowadays, the knowledge base is searchable online, by full text as well as metadata, such as author, title, and the Mathematics Subject Classification (MSC [Msc]; see also [section 2.1.7.4](#) on classification schemes). Most research results are again published on the Web, either by commercial publishers, or by the researchers themselves, using freely accessible pre-print servers, such as arXiv [Arx].

Computer-based mathematics tools, such as computer algebra systems (CAS), proof assistants, and program verification systems, draw on large libraries, in which the required foundations of mathematics and previous research results have been formalized. While these libraries are primarily shipped with the particular tool they have been made for, many of them have also been published on the Web for a long time – while still being edited and maintained off the Web. For example, the Journal of Formalized Mathematics, publishing proofs of the Mizar Mathematical Library (MML [Miza]), which have been checked using the Mizar proof checker [Mizb], has existed since 1990 – on paper, and, for most of that time, also on the Web [For].

Finally, there are educational content and reference works. Not only do many mathematics educators put their lecture notes online, but there are also more structured, searchable and browsable knowledge collections. Two examples of general-purpose reference works – not particularly pedagogically optimized – are the Digital Library of Mathematical Functions (DLMF) and Wolfram MathWorld. The DLMF is a centrally edited reference of special functions and their application [Natio]. MathWorld is a collection of about 13,000 hyperlinked and categorized entries on mathematical topics, which has been maintained since 1999 by ERIC S. WEISSTEIN, with contributions from a larger community [Weib]. For about a quarter<sup>14</sup> of all MathWorld entries, related files (“notebooks”) for the Mathematica CAS [Urla] are available for download. However, they have been hand-crafted and do not directly correspond to the informal content of the encyclopedia entries.

While educational content has usually been optimized for comprehensibility by a human target audience, this is not necessarily the case with formalized libraries. In order to make the latter more comprehensible, researchers in the field of mathematical libraries have applied results from software engineering, where the importance of documentation for understanding is undoubted and confirmed by research on program understanding<sup>15</sup> [VMS99; Stoo6]. While mainstream source code documentation usually extends down to the level of functions/methods, the structure of a *literate program* is, more radically, governed by the flow of a natural language explanation of the program logic, interspersed with fragments of source code [Knu92]. Both pure, compilable source code and human-readable documentation can be generated from a literate program. Some proof assistants support integrated documentation that can yield L<sup>A</sup>T<sub>E</sub>X output; PAUL CAIRNS and JEREMY Gow provide an overview and have themselves developed maze, a “literate proving” prototype for Mizar [CGo6]. As a complementary approach towards making formalized libraries more

<sup>14</sup>estimation based on 200 random entries downloaded on December 2, 2009

<sup>15</sup>also known as “program comprehension”



comprehensible, they have given a proof of concept for opening up the MML to a wider audience, particularly professional mathematicians and mathematics students, i.e. “*people interested in proof-centred mathematical content [but having] little or no understanding or interest in formal mathematics per se*” [GCo7]. They exposed parts of the MML via a traditional digital library web interface and observed the following issues that a user-friendly interface would have to address: reducing the verbosity of a formalized representation, interleaving informal explanations (which do not exist in the original MML sources) with the formalized content, hiding formalized content irrelevant to understanding, splitting long formalizations into more digestible units, providing a search facility adequate to the structures of the formalized content but still using a comprehensible query language [GCo7].

### 1.4.1.1 Critique – Easy Access, but Poor Collaboration and Retrieval

Summarizing, Web 1.0 sites facilitate the *access* to mathematical knowledge, be it publications or formalized libraries. However, (i) they do not yet facilitate collaboration, and (ii) the means of automatically retrieving, using, and adaptively presenting knowledge are restricted.<sup>16</sup> [Problem \(i\)](#) is partly addressed by web 2.0 applications, and [problem \(ii\)](#) has been partly addressed by early attempts at applying semantic web technologies to mathematics. Both are reviewed in the following sections.

### 1.4.2 Web 2.0 – Open Collaboration in Blogs and Wikis

Mathematicians are using the Web 2.0 for collaboratively developing and discussing new ideas and results, but also as a new publication channel for established knowledge.

Several mathematics researchers and research groups share thoughts and preliminary findings on blogs. They are eager to collect feedback, far before traditional peer review mechanisms take effect, yet with a broader reach than informal face-to-face or mailing list discussions. JOHN BAEZ, an active blogger himself (see below), mentions successful collaborations among mathematicians not knowing each other before, which had started by comments on blog posts, and finally converged into the publication of conventional articles [Bae10].

However, the authors of the n-Category Café blog [Nca], one of them being BAEZ, found their blog to be neither the most suitable medium for collaboratively evolving an idea that had emerged from a blog discussion, nor for creating permanent, short, interlinked descriptions of topics (cf. [section 1.3](#) on the general benefits and drawbacks of blogs). Therefore, they created the nLab wiki [Nla] as a companion site for archiving discussions from the blog by topic, but also as an open group lab notebook for taking notes and collaboratively developing new ideas. The nLab wiki is an example for the emerging practice of *Open Notebook Science*, i.e. “*making the entire primary*

---

<sup>16</sup>This has been observed independently by JÜRGEN RENN: “*The structured representation of mathematical formulae using MathML in the internet of today still plays a subordinate role, particularly when considering the potential of subsequent processing, multimedial presentation, and the cross-linking of formal expressions. We wonderfully represent common speech in the internet, we work with hypertext, but not with ‘hyperformulae’.*” [SGRo9]; original German source: „[...] die strukturierte Darstellung mathematischer Formeln im heutigen Netz mit Hilfe von MathML spielt immer noch eine untergeordnete Rolle, insbesondere wenn man an das Potential der Weiterverarbeitung, an multimediale Darstellung und die Vernetzung formaler Ausdrücke denkt. Wir bilden die Umgangssprache wunderbar im Netz ab, wir arbeiten mit Hypertext, aber nicht mit ‚Hyperformeln.‘“ [SGRo9]

record of a research project public[, including] failed, less significant, and otherwise unpublished experiments” [Wik10c]. Similarly, TIMOTHY GOWERS, who had actively been blogging before, initiated Polymath in 2009 – a massive collaborative effort to prove a theorem<sup>17</sup> using a blog as the exclusive communication medium [Polc; GN09; Bar10]<sup>18</sup>. Within 37 days, 27 voluntary participants, from students to professors, contributed approximately 800 comments [GN09]. More recently, another Polymath project, the collaborative review of a claimed proof of the  $P \neq NP$  statement of computational complexity [Pola], gained considerable public attention. The Polymath maintainers have also set up a companion wiki [Polb] for “collect[ing] pertinent background information which was no longer part of the active ‘foreground’ of exchanges on the [...] blog entries” [Bar10]; however, it “does not appear to have been as actively used in support of the ongoing research discussions, themselves, as might have been possible” [Bar10]. GOWERS established a set of collaboration rules, which included [Gow09b]: Work top down (from general comments towards more technical elaborations; rule 1), write comments that are easy to understand (2), don’t hesitate to express preliminary and incomplete ideas (3), argue constructively (10), do not go offline to solve a problem on your own (5, 6), announce your further steps by explicit comments (7), roll out subdiscussions into new threads to keep the main discussion focused (12), and acknowledge all contributors if the experiment should result in a publication (15). As advantages of doing research on public blogs and wikis, he emphasized that they archive the complete history of comments and changes, thus transparently exposing the ownership of contributions, and making all previous solution attempts available to new members [Gow09a]. Similarly, BAEZ points out the value of such sites in supporting newcomers to the fields in “get[ting] a sense of what research is like” [Bae10]. More specifically, one can argue that such sites, thanks to their archiving of comments and changes, help to promote a better understanding of how mathematical findings have been made (cf. section 1.1).

The audience of a research blog is relatively small, and thus a researcher blogging about a problem he got stuck with might not receive instant help. On the MathOverflow forum (cf. [Mate] and section 6.6.4), started in 2009, users can post their own problems and solutions to others’ problems. In an agile “simulation” of the traditional mechanisms of scientific publication and peer review, users automatically gain reputation by posting answers that receive a positive rating from the community.<sup>19</sup> While MathOverflow focuses on concrete problems and solution, the Tricki [Tri] – also initiated by GOWERS, in 2008, – is a wiki repository of general mathematical techniques, comparable to a web 2.0 remake of PÓLYA’s “How to Solve It” [Pól73].

The Polymath wiki and the Tricki have been set up from scratch, not reusing content from existing knowledge bases, but the maintainers of established knowledge bases are also starting to use web 2.0 frontends to support collaboration. Particularly for the extremely cost-intensive formalization of textbook mathematics into machine-checkable code, such technical support is welcomed. ASPERTI et al. cite cost figures from one week per textbook page to 1.5 hours per line of code, and observed that the top-down workflow in wikis, where new, more specific content is typically created by pointing a (dangling) link to it from an existing article and then creating

<sup>17</sup> Actually, he chose a theorem that was already known to be true. The goal of Polymath was to find an elementary proof, from which new insights were expected [GN09].

<sup>18</sup> Where project homepages are available in addition to scientific publications, they are generally cited in the first position.

<sup>19</sup> This way of collaborative problem solution was pioneered by Stack Overflow [Sta], a site about programming problems started in 2008. In fact, MathOverflow runs the same software.

the new article, matches the top-down process of formalizing mathematics well [AGN09]. A concrete example is the recent wiki frontend for the Mizar Mathematical Library (MML) [UAR+10]. The wiki intends to support common workflows in enhancing and maintaining the MML and thus to disburden the human MML Library Committee. At the time of this writing, the wiki is in a prototype stage and thus not yet the main interface to the MML. Development has so far concentrated on the underlying distributed version control system, for which the wiki is a browsing frontend; it is currently assumed that the contributors mainly edit offline and have the Mizar proof checker installed locally. However, potentially reusable prototypes of wiki frontends for proof assistants exist (see, e.g., [CK07], and [section 9.5](#) for further examples).

Several more widely known wikis have so far been used for collecting existing mathematical knowledge and editing it for educational and general purposes. PlanetMath [Plab] is a community-run mathematics encyclopedia, counting more than 8,000 entries<sup>20</sup>. Wikipedia, a community-run general-purpose encyclopedia with 15 million articles in over 250 languages, also covers mathematics [Wik09c]. Out of the 825,000 articles that the German Wikipedia had in 2008, about 7,000 covered mathematical topics [Biro8]. Wikipedia targets a general audience, including non-mathematicians. Therefore, it focuses less on formal aspects and a rigorous presentation (e.g. by omitting most proofs), but it embeds the pure mathematical knowledge into a wider context, including, e.g., the history of mathematics, biographies of mathematicians, and information about application areas. The lack of proofs is sometimes compensated by linking to the technically similar ProofWiki [Prob], containing over 2,500 proofs, or to PlanetMath.

While PlanetMath and Wikipedia do not exclusively focus on education, albeit being frequently used by students as a source of information, Connexions [Cnxa] is an open, web repository for courseware. Connexions promotes the contribution of small, reusable course modules to its “content commons”, so that the original author, but also other users can flexibly combine them into collections, such as the notes for a particular course. Currently, there are more than 17,000 modules in over 1,000 collections – about 4,000 modules in 100 collections from mathematics and statistics, and about 6,000 modules in 400 collections from science and technology. Connexions has been realized on top of a traditional CMS; it differs from a wiki in its more rigid management and publication workflow. Modules are created as drafts before publication, and collaborators have to be invited by the original authors.<sup>21</sup> Compared to the above-mentioned MathWorld, PlanetMath, Wikipedia, and Connexions have the following features in common: (i) Their content can directly be edited on-site, (ii) it is controlled by the community rather than a central authority, and (iii) it is available under a Creative Commons license permitting free reuse, redistribution, and creation of derivative works<sup>22</sup>.

#### 1.4.2.1 Critique – Little Reuse, Lack of Service Integration

Web 2.0 applications have facilitated collaboration but still *require* a massive investment of manpower for compiling a knowledge collection. This is usually done from scratch (as in research blogs

---

<sup>20</sup>all figures as of July 2010, unless stated otherwise

<sup>21</sup>In a brief comparison to Wikipedia, the maintainers argue that Connexions’ notion of ownership is more attractive for academic authors, as it is consistent with established academic conventions [Cnxb].

<sup>22</sup>PlanetMath was actually created in response to a temporary shutdown of MathWorld in the course of a copyright lawsuit.



and wikis, open encyclopedias and courseware repositories), or alternatively by equipping a previously existing knowledge collection with a web 2.0 interface in order to facilitate its maintenance (as done for the MML).

However, machine-supported intelligent knowledge reuse, e.g. from other knowledge collections on the Web, does not take place. Different knowledge bases are technically separated from each other by using document formats that are merely suitable for knowledge presentation but not for representation, such as XHTML with L<sup>A</sup>T<sub>E</sub>X formulæ. In these formats, the only way of referring to other knowledge bases is an untyped hyperlink that a human reader can click but that a machine does not understand. The proof techniques collected in the Tricky cannot be automatically applied to a problem developed in a research blog, as neither the proof techniques nor the problems are sufficiently formalized.

Intelligent information retrieval, a prerequisite for finding knowledge to reuse and to apply, is poorly supported on web 2.0 sites. For example, Wikipedia states the Pythagorean theorem as  $a^2 + b^2 = c^2$  and files it into the categories “Articles containing proofs” and “Mathematical theorems” [Wik09d]. The L<sup>A</sup>T<sub>E</sub>X representation of the formulæ does not allow them to be searched by their functional structure. Putting the fact aside that Wikipedia cannot search formulæ at all, a search for the equivalent expression  $x^2 + y^2 = z^2$  would not yield the theorem, and certain more complex rewritings, such as  $c = \sqrt{a^2 + b^2}$ , would probably only be retrievable because they explicitly occur in the article as well. From the categorization it is neither clear for a machine (albeit very likely for a human) whether the article contains a proof of the theorem – or just any other, unrelated proof – nor whether the proof is correct.

Repositories of formalized mathematics, such as the MML, use specialized search engines (cf. [Bano6]). They do support internal knowledge reuse by formalizing new mathematical concepts of existing ones and proving new theorems by applying ones that have already been proven, but they do not support links to external repositories<sup>23</sup>. Thus, the maintainers of each knowledge collection, informal or formalized, hope to receive a critical mass of contributions that makes it sufficiently self-contained for the desired application.

Finally, the integration of mathematical web 2.0 sites with automated reasoning and computation services is scarce. As stated in section 1.1, automated reasoning is increasingly used to support the development of new mathematical theorems and proofs. Moreover, studying concrete examples is a key to testing mathematical hypotheses and understanding established results, and *computing* the value of a function for concrete given values, and possibly visualizing the result, is a task that, by their very nature, *computers* excel at. As pointed out above for information retrieval, the representation of mathematical knowledge in web 2.0 repositories is often too presentation-oriented to be amenable to automated reasoning and computation. Interactive computation is available in mathematical e-learning systems, such as ActiveMath [Act] or MathDox [Matb] – where document authors have sufficiently formalized the underlying mathematics in separate editing tools before publishing –, but less so in general-purpose digital libraries and collaboration

<sup>23</sup>Translating entries of one formalized library for reuse in another one is, however, hard, due to differences not only in syntax (i.e. different languages for representing axioms, theorems, and proofs), but also, more importantly, in semantics (i.e. different logical foundations).

environments. Mashups, which have otherwise been a driving force of web 2.0 development, scarcely exist for mathematical tasks.<sup>24</sup>

### 1.4.3 Semantic Web for MKM – Digital Libraries and Web Services

In the early 2000s, XML-based markup languages were increasingly used for representing mathematics, particularly formulæ. MathML mainly focused on representing their layout, enabling browsers to render them when embedded into HTML (cf. [section 2.4.2](#)). The complementary OpenMath language focused exclusively on the functional structure of mathematical expressions, targeting information exchange between symbolic computation software (cf. [section 2.4.3](#)). Around the same time, the first building blocks of the Semantic Web vision, such as the RDF vocabulary description language RDFS, were on their way towards standardization, and first prototypical implementations, e.g. of RDF-aware databases and query engines, were coming up (cf. [section 2.3](#)).

These developments sparked interest in the emerging MKM community. They hoped that Semantic Web technologies would help to address their challenges. This seemed technically feasible, particularly as the aforementioned markup languages and RDF shared a common foundation of XML and URIs [[Mar03](#)]. The two main lines of applying semantic web technologies to MKM focused on *digital libraries* – improving information retrieval and giving readers access to automated reasoning and computation services –, and *web services* – providing self-describing interfaces to automated reasoning and computation on the Web, so that they could solve problems sent to them by humans or other automated agents.

#### 1.4.3.1 Digital Libraries – MathNet, HELM, and their Spin-Offs

MathNet [[Int](#)], which had started as a German research project from 1997 to 1999 and was then internationalized by the International Mathematical Union, was an effort to build “*a distributed, efficient and user-driven information and communication system for mathematics*” [[DSN01](#)]. Mathematical institutes were advised to put up uniformly structured homepages and publishing preprints and annotate both with machine-comprehensible RDF. Recommended vocabularies included Dublin Core (cf. [section 2.1.7.3](#)) for general bibliographical metadata, MSC (cf. [section 2.1.7.4](#)) for describing the subject of a publication, and a MathNet-specific vocabulary for describing the structure of an institute homepage. Some of the 180 MathNet homepages that existed in 2002 [[Spe03](#)] are still online; however, the central services, including a preprint search engine<sup>25</sup> and a browser for MathNet pages, have either been defunct or no longer supplied with recent data since 2007.

HELM, the Hypertextual Electronic Library of Mathematics [[Hel](#); [APSC+03](#)], was developed from 1999 on, independently from MathNet and partly supported by the MoWGLI (Mathematics on the Web – Get it by Logic and Interfaces [[Mow](#)]) and MKM-NET (Mathematical Knowledge

---

<sup>24</sup>ProgrammableWeb [[Proa](#)], a directory of mashups, lists 3 mashups tagged with “math”, out of nearly 5,000 mashups overall. This may, however, change soon. Wolfram, who had already released the Wolfram Alpha “computational knowledge engine” (cf. [section 7.6.2](#) and [[Wola](#)]), recently released a number of “widgets” that perform simple computations backed by Wolfram Alpha and can be embedded into web pages, as well as a development environment for creating new widgets or derivatives of existing ones [[Wolb](#)]. However, these widgets are limited to acting as frontends to Wolfram Alpha.

<sup>25</sup>... which actually featured the first working implementation of Dublin Core [[Plü04](#)]!

Management Network [Mona]) European projects. HELM aimed at “*integrat[ing] the current tools for the automation of formal reasoning and the mechanization of mathematics [...] with the most recent technologies for the development of web applications and electronic publishing*” [Hel]. In contrast to MathNet and other traditional digital libraries, where one document is the atomic unit of information, HELM intended to explicitly represent the fine-grained structures of mathematical expressions to expose them to, e.g., automated reasoners, but also to enrich their publication on the Web. For example, mathematical formulæ were rendered in MathML in such a way that actions could be invoked on them, such as simplifying a selected (sub)expression using an automated reasoning backend attached to the library. HELM completely relied on XML and RDF not only for publishing, but also for its internal knowledge representation. Formalizations of mathematical statements and proofs were encoded in one XML dialect per underlying logical system, which was obtained by translation from the native formalized language. This was actually carried out for the library of the Coq proof assistant [Coq], for which an XML export was developed. Relevant structural properties (e.g. the top-level operator of a mathematical statement; cf. section 2.4.10.2 for details), interrelations, and metadata were represented as RDF.

The HELM developers had to make a lot of foundational research and development, as suitable reusable implementations were not available for many of the planned features. Two concrete examples are query answering and rendering of interactive formulæ.

**Query answering:** None of the prototypical RDF query engines that were available in 2003 satisfied the HELM requirements<sup>26</sup>; therefore, a new one, called MathQL<sup>27</sup>, was developed [GSo3; Guio3]. Later on, the more efficient Whelp search engine featured a completely reimplemented query engine with a TeX-like query syntax [AGC+06]. While still following the same paradigm of indexing structural metadata, the technical dependency on RDF was eliminated.

**Rendering interactive formulæ:** Browsers did not sufficiently support MathML in the early 2000s. Therefore, GtkMathView, a MathML rendering widget suitable for embedding into desktop applications, was developed [Pad].

### 1.4.3.2 Web Services – MONET and Related Architectures

The MONET European project pioneered an architecture for mathematical web services built on semantic web technologies [Monb; CDTo4a]. MONET services give access to numeric and symbolic computation systems; access to proof assistants or digital libraries was envisaged but not pursued. MONET services come with a machine-comprehensible description of their capabilities and can be registered with a central broker. Mathematical expressions in queries or computation requests to the broker were represented by their functional structure using OpenMath (cf. section 2.4.3). The broker would match the problem received against the registered web services (cf. section 2.4.10.2 for an example) and then, through the web service interface that matched best, invoke the actual underlying mathematical service. As with HELM, MONET also required some foundational work

<sup>26</sup>independence of a concrete RDF syntax (such as RDF/XML), disjunction, data source identification, and a well-defined formal semantics [Guio3]

<sup>27</sup>The first version focused on generic RDF queries. Further mathematics-specific extensions were planned, but not realized for the RDF-based MathQL.

to be done. The standard semantic web ontology language OWL and an OWL-based reasoner were already found suitable for the internal description of services and problems and computing matches. However, the XML-based frontend languages for service descriptions and queries (which the broker then translated to OWL) had to be designed from scratch. Furthermore, the OWL reasoners of that time could not efficiently deal with a large number of instances of classes (here: concrete problems instantiating problem descriptions), which required a specific database/reasoner hybrid to be developed, the Instance Store, but then, again, the separate treatment of classes and instances constrained the design of the MONET ontologies in that they had to model every object as a class [CDT04b]. The MONET project ended in 2004. Parts of its query language are still used in the MathDox e-learning system [Matb; CCV10]. More importantly, MONET and the competing MathBroker architecture for symbolic computation web services [Baro6] influenced each other. The latter was continued until 2007 but made less use of semantic web service technologies; instead it evolved some of MONET's proprietary languages and introduced new ones. The MathServe architecture, influenced by both of the former but focusing on automated reasoning services, made extensive use of more recent semantic web service technologies, such as OWL-S service profiles [Zimo8].

#### 1.4.3.3 Critique – Early Efforts Discontinued

Semantic web approaches to MKM have so far failed to fulfill the hopes set in them. The aftermath of the early research efforts HELM and MONET is an instructive example.<sup>28</sup> In both projects, the researchers were initially enthusiastic about the possibilities of the emerging Semantic Web, but then it turned out that, apart from specifications of languages, few stable and reusable implementations existed, and hence a considerable amount of resources had to be invested into developing fundamental building blocks (MathQL and GtkMathView in the case of HELM, the Instance Store in the case of MONET).<sup>29</sup> Whelp, GtkMathView, and other parts of HELM have survived in the desktop-based interactive proof assistant Matita [ASCT+07], whereas the web frontend and the RDF-based components have been discontinued. Semantic web technologies are not yet a well-established basis for mathematical web services either. While large parts of the influential OpenMath community had been involved into MONET, which heavily relied on Semantic Web technologies, the current driving force of research symbolic computation web services, the SCIENCE project (Symbolic Computation Infrastructure for Europe [Sci]), does not use “standard” Semantic Web service technologies at all: SCSCP (Symbolic Computation Software Composability Protocol [HHK+10]) is a lightweight XML protocol using TCP sockets, or alternatively SOAP, whose communication semantics heavily relies on a custom OpenMath vocabulary.

<sup>28</sup>The reasons for discontinuing MathNet have not been documented in publications and are not known to me by other means.

<sup>29</sup>The HELM developers made no secret out of their frustration: “It is a pity that [...] most of the expectations about XML technologies [including RDF] have not been fulfilled due to intrinsic deficiencies in their design and implementation. MathML failed to be adopted by major browsers; XSLT is just too prolix for simple operations and too weak for more complex ‘content sensitive’ operations; XQuery is too slow for large, highly structured data bases; and RDF never really went beyond the project phase.” [AGN09] Personal communication with ANDREA ASPERTI on 2010-07-09 confirmed that that statement referred to the immaturity of these technologies at the time of developing HELM.

## 1.5 Collaborative Mathematics on the Web – Why Retry Now?

Web 1.0 applications for mathematics are ubiquitous nowadays, but they merely facilitate access to diverse kinds of mathematical knowledge – from scientific publications to formalized libraries to educational content. Web 2.0 applications, particularly blogs and wikis, have succeeded in attracting an increasing number of working mathematicians, who comment on new ideas, collaboratively write publications, and collect and remix educational knowledge. The usage of semantic web technology to improve information retrieval and the integration of automated reasoning and computation services with knowledge bases and with each other has been investigated, albeit without becoming mainstream yet. Here, I argue why a new combination of web 2.0 and semantic web technologies is needed to address them, and why such a solution is now feasible.

### 1.5.1 Combining Semantic Web and Web 2.0 for MKM

The combination of web 2.0 and semantic web technology has already proven successful in some fields, as mentioned in [section 1.3.4](#); however, it has hardly been applied to MKM yet. In a “postface” to his Ph.D. thesis in the context of Matita, STEFANO ZACCHIROLI gave two reasons why a hypothetical retry of HELM (cf. [section 1.4.3.1](#)) would benefit from web 2.0 technology [[Zac07](#)]: Web 2.0 applications enable direct editing of mathematical content on the Web, and projects like PlanetMath have proven that there is “*a community of people interested in collaboratively authoring rigorous mathematics on the web*” [[Zac07](#)].<sup>30</sup> Furthermore, a retry of HELM, MONET, and other early attempts to do MKM with semantic web technology would now benefit from a much wider availability of stable libraries and tools. For example, there is now a standardized and widely supported query language for RDF (SPARQL; cf. [section 2.3.3.5](#)).

### 1.5.2 What MKM can Contribute to the Semantic Web

*How does it happen that mathematics has remained  
as if it were a blind spot in our culture  
— alien territory, in which only the elite, the initiate few  
have managed to entrench themselves?*

—HANS MAGNUS ENZENSBERGER [[Enz99](#)]

Conversely, there are now also opportunities for MKM to give back to the Semantic Web. Mathematics is a ubiquitous foundation of science, technology, and engineering. Some of these application areas are already well represented on the Web of Data, but their mathematical foundations are not. Having them as well represented would enable a whole range of new applications:

**General-purpose Mathematical Knowledge:** The inadequate representation of mathematical knowledge in Wikipedia has been criticized in [section 1.4.2.1](#). DBpedia, the linked dataset obtained from Wikipedia (cf. [section 1.3.3](#)) inherits these limitations. Such limitations – in DBpedia and elsewhere – forced the Polymath collaborators mentioned in [section 1.4.2](#) to search for previous publications of refutations of  $P \neq NP$  “proofs” by keyword.

<sup>30</sup> Similarly, BAEZ suggests that the release of a T<sub>E</sub>X formula editor plugin for the popular WordPress blog engine was a major incentive for mathematicians to start blogging [[Bae10](#)].

**Statistics:** Public sector information, increasingly being published as linked data by the US, UK, and other governments [ST10; DDG+10], has been used to provide, e.g., localized information retrieval about political representatives, crime statistics, and hospital waiting list statistics [OKP+10]. Statistical datasets contain values derived from ground values, or from other derived values using mathematical functions. Planning data collection from statistical datasets and interpreting collected data requires a notion of mathematical provenance of their data points [VLH+10; Lanio].

**Publication Databases:** The RKB Explorer ACM linked dataset [Adv] classifies the scientific publications of the ACM according to their Computing Classification System (cf. [section 2.1.7.4](#)). Still, it is impossible for a linked data agent to understand that a publication merely classified as “F.1.3 Complexity Measures and Classes” actually deals with the  $P$  and  $NP$  complexity classes, and how they are defined.

**Enterprise Applications:** Linked data do not have to be open; the architecture also works in enterprise intranets. Renault has used them for retrieving information about spare car parts [Sero8]. Now consider decisions to be made when designing whole cars: They ultimately require mathematical understanding. An engineer looking for an efficient engine for a projected city car might feed inputs such as the weight of the car, the average length and duration of a trip, the most widely available type of fuel and the average environment temperature when starting the engine into a mathematical model of the engine in order to predict its fuel consumption under these constraints.

**e-Science:** The above use case is actually about reproducing an experiment – one of the key principles of e-science [BAB+10]. Publishing descriptions of scientific experiments as linked data not only makes the provenance of their result data explicit [MSZ+10] but also makes whole experiments more easily accessible and thus reproducible. Fine-grained reproducibility once more demands a representation of the mathematical models. Some e-science datasets include them, e.g. the SysMO SEEK “*assets catalogue’ describing data, models, [...], workflows and experiment[s]*” [BAB+10] from systems biology of microorganisms [Sys], whose publication as linked data is in progress (cf. [BAB+10]). Currently, the mathematical models are given as Content MathML formulæ (cf. [section 2.4.2](#)) deeply nested into XML files and thus not directly accessible via URIs.

Thus, in order to enhance current applications of linked data towards mathematics, dataset publishers need a mathematical vocabulary. The quality of linked data vocabularies – often designed in an ad hoc mapping of existing database structures to RDF – and hence of the linked datasets is often low (see, e.g., [JHY+10]). ANTOINE ZIMMERMANN has observed the following reasons for vocabularies being of a low quality [Zim10]:

1. ontologies defining the domain of interest do not exist;
2. they exist but are difficult to find because developed by small groups for experimentation, lacking advertisement;
3. they exist and can be found but they are of poor quality, not complying with standards or best practices;



4. they exist and can be found but there are too many, of mixed quality, and it is difficult to assess which ones are appropriate for a specific use case.

High-quality machine-readable vocabularies for mathematics do exist: The official OpenMath 2 Content Dictionaries (CDs), for example, defining 260 mathematical symbols – operators, functions, sets, constants –, have undergone a strict review process (cf. [section 2.4.3](#)). Large parts of the MKM community accept them as standard vocabularies for representing mathematical expressions. However, for the rest of the world – including the publishers and ultimately the consumers of linked data – ZIMMERMANN’s [criterion 2](#) applies to the OpenMath CDs. Besides a technical mismatch – they are not available as RDF<sup>31</sup> – I argue that there is a *cultural* mismatch. Mathematics, due to its practice of rigorously reasoning about abstract concepts in a self-contained way using a symbolic notation, is generally perceived as hard and inaccessible (see, e.g., [Enz99]), and mathematicians, at least pure mathematicians, are perceived as sitting in the Ivory Tower. The average computer scientist, whose work builds on a very restricted area of applied discrete mathematics, is not immune to such stereotypes. By expanding both the “mental infrastructure” required for understanding mathematical knowledge (cf. [section 1.2](#)) and meeting the technical requirements of linked data publishers, we can, by way of linked open data becoming more and more widely applied, take mathematics out of the Ivory Tower.

## 1.6 Challenges to be Addressed by a New MKM Infrastructure

Before establishing an agenda towards a new MKM infrastructure based on web 2.0 and semantic web foundations, we need to understand the challenges of such an enterprise. These lead to a set of key research questions, which previous MKM and semantic web research have not yet addressed sufficiently, but which this thesis has to address.

### 1.6.1 The User’s Perspective: Providing Incentives to Contributors

When expecting users to do the hard work of contributing expressive, well-structured mathematical knowledge to a shared collection, the mere perspective that this content may be more comprehensible, reusable, and applicable for *other*, future users and that it is *potentially* accessible to a wide range of semantic services is not necessarily enough of an incentive for them, as ANDREA KOHLHASE observed for the special case of authoring semantic documents, including semiformal mathematical knowledge [Koho8a]. She proposed a tight integration of relevant “*author-tailored services*” into the setting of authoring as an incentive – to entice the author into performing the next step of formalization – or as a gratification – to point out to the author the benefits of the formalization step he has just made [Koho8a]. Incentives and gratification are frequently re-occurring themes in Semantic Web research: The idea of “*instant gratification*” [MEG+03] has already motivated the design of early semantic web user interfaces and led to the idea of using wikis as an interface for authoring and browsing semantic content [AA05].<sup>32</sup> ASPERTI et al. requested similar functionality for future authoring tools for libraries of formalized mathematics:

<sup>31</sup>That can be accommodated for, as explained in [section 6.4.1.1](#).

<sup>32</sup>KATHARINA SIORPAES and ELENA SIMPERL provide a more recent summary of incentives and incentive-based semantic web tools [SS10].

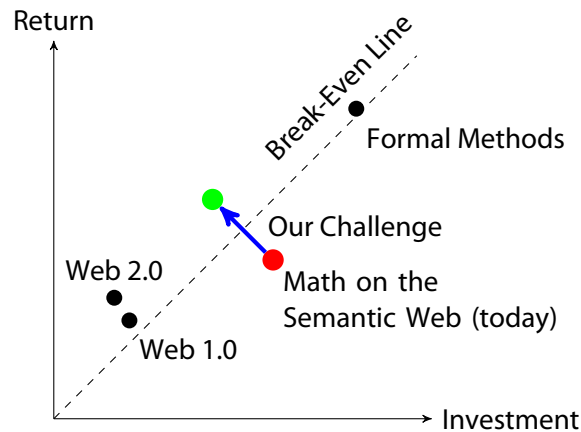


Figure 1.2: The cost/benefit challenge for editing semiformal mathematical knowledge on the Web

The point is not only to reduce [the] cost [of formalization], but also to improve the benefits coming from the representation of the information in a “machine comprehensible” richly structured format, suitable to be elaborated by a machine. This means developing innovative, content-based functionalities, eventually overcoming the reductive operational perspective of *verification*. (ASPERTI et al. [AGN09])

The developers of the MathLang language and toolkit argue in the same vein and once more stress the influence of the choice of representation language on effective services:

The degree of formality in representing the mathematical semantics should be flexible, and at least one choice of degree of formality should be both inexpensive *and* useful. (FAIROUZ KAMAREDDINE et al. [KWZo8])

Thus, from a user’s perspective, the most important research questions for a semantic web collaboration environment for MKM is:

**Value:** How can the investment required to create a human- and machine-comprehensible representation of mathematical knowledge be lowered, while at the same time utilizing that knowledge to provide useful knowledge management services and applications?

This question is hard to answer, and will not be answered fully in this thesis, but it served as the central motivation for conducting this research. In the following, I break it down into more concrete questions, which this thesis will answer.

Figure 1.2 compares the cost/benefit challenge for MKM on the Semantic Web to existing web 1.0 and 2.0 sites and formal methods (e.g. for software verification), which are already sufficiently successful to survive. Maintaining a knowledge collection is never an end in itself; the knowledge is always utilized for services such as providing helpful and comprehensible information to users or supporting applications. Once these basic needs have been met, additional services can be provided to facilitate the collaborative maintenance of the knowledge. However, on the Web 2.0, where



the same user can both be a consumer and a producer<sup>33</sup> (see, e.g., [Koho8a]), this second step is short: A sustainable web 2.0 site should try to entice the consumer into producing user-generated content by giving him quick, local access to an editing interface, but also ensure the producer's loyalty by continuously perceiving him as a consumer and providing him with incentives and gratifications. The ultimate hope is that this turns into a productive feedback loop yielding more and more users, who contribute more and more high-quality content.

### 1.6.2 Research Questions

The above analysis of possible ways of turning consumers of mathematical knowledge into producers and offering them high return on investment leads to another two fundamental research questions on supporting knowledge management workflows and knowledge reuse, which previous research on mathematics on the Web has not yet addressed sufficiently. From the user's perspective, effective workflow support is most crucial:

#### **Workflows: How can workflows be transferred from one knowledge base to another one?**

Each mathematical knowledge base currently supports a relatively small set of knowledge management workflows by employing a set of services specialized to one particular knowledge representation language. The practical consequences are:

- Effectively acquiring and organizing mathematical knowledge and making it comprehensible and reusable (recall [section 1.2](#)) comprises multiple complex workflows. When a knowledge base only supports few of them, it is likely that a user will not get adequate support with the next workflow he wants to perform.
- Users fall victim to “vendor lock-in”: They are forced to provide any additional knowledge, for which they need support with performing a workflow, in the native language of a knowledge base that supports that workflow – if the workflow is supported at all.
- Contributors to a knowledge base can neither rely on a return on investment nor on the sustainability of their contribution. They may have put effort into authoring or formalizing a piece of knowledge, but after some years the specific representation language or the services might no longer be maintained.<sup>34</sup>

I will answer the “Workflows” question by methods and techniques for integrating heterogeneous services (see below). An effective integration of heterogeneous services has to consider the knowledge representations that they understand:

<sup>33</sup>The choice of the word “can” is deliberate. A survey among US adult online consumers in late 2006 confirmed that most Web 2.0 users do not actively contribute. The users were grouped into six overlapping categories, resulting in the following distribution (based on participating in an activity at least monthly): creators (publishing original content): 13%, critics (commenting, rating, reviewing original content): 19%, collectors (tagging original content): 15%, joiners (using social networking sites): 19%, spectators (consuming user-generated content): 33%, inactives: 52% [LBo8]. Specifically for Wikipedia, “a very skewed distribution, with less than 10% of the total number of authors performing more than 90% of the total number of contributions” [Orto9] has been observed from database dumps of late 2007.

<sup>34</sup>Indeed users have been found shy of formalization due to the danger of prematurely committing to the wrong formalization and not being able to convert to a different one, and the perception that incorrectly or inconsistently formalized information might even be less useful than information not formalized [SM93].

**Knowledge:** How can knowledge be made reusable and comprehensible across knowledge bases? Contemporary mathematical knowledge bases, both informal and formalized ones, do not even expose part of their knowledge in a mutually understood language, which hampers knowledge reuse – in the sense of retrievability by automated agents, comprehensibility for human end users, and the connection of mathematical knowledge with knowledge from its non-mathematical areas of application.

To the “Knowledge” question, I will contribute an improved exchange language, which bridges the native languages of different knowledge bases. The state of the art suggests two approaches, each of which addresses half of the question: Expressive mathematical representation languages, such as the OpenMath extension OMDoc (cf. [section 2.4.4](#)), cover the structures of mathematical knowledge well but hardly interact with knowledge represented in other languages and with non-mathematical knowledge that occurs in practical applications. RDF has been used to share structural outlines of knowledge and metadata across knowledge bases but lacks vocabularies for mathematics and is not entirely adequate for representing complex mathematical structures (cf. [section 2.4.10](#)). Both approaches are actually complementary, and thus I will combine them. Note that a knowledge representation intended to facilitate reuse must not only target automated agents and information retrieval, but also human users trying to understand what the machine has inferred or retrieved for them, and whether and how they can apply it. It has been observed before (e.g. by Gow and CAIRNS, cited in [section 1.4.1](#)) that *both* the knowledge representation language and the services running on top of it contribute to understanding; that has to be taken into account when designing an exchange language.

Having committed to an exchange language enables us to answer the “Workflow” question. Exchange languages have already proven their utility for connecting services in the *same* application area, whose internal knowledge representations have a similar expressivity – consider OpenMath/SCSCP connecting different CAS (cf. [section 1.4.3.3](#)). Exchange languages as those mentioned above can span a wide range of applications and thus have the potential to connect services for formalized mathematics to services for scientific publications or educational repositories. Understanding, a particular focus of this thesis, is already fostered by individual systems; for example, the MathDox e-learning system computes examples via a CAS backend [[Matb](#); [CCK+o8](#)]. However, my goal is to achieve recombination with the ease of web 2.0 mashups.

The strong dependency of the envisaged infrastructure on an expressive exchange language entails a fourth research question:

**Authoring:** How can expressive knowledge representations be authored collaboratively? Web 2.0 interfaces have so far been successful for informal mathematics texts with presentation-oriented formulæ. Semantic markup languages, such as OMDoc, or languages for formalized mathematics, such as Mizar, have so far been edited using plugins for advanced desktop text editors, such as Emacs or jEdit (see, e.g., [[Jano6](#); [Lib10b](#)]). These usually assume expert users and do not offer particular collaboration support beyond a basic integration with revision control systems. Our envisaged environment needs to support the formalization of informal texts, as well as the annotation of formalized content with human-comprehensible documentation, taking into account the collaborative setting.

My short answer to the “Authoring” question is to reduce complexity. This question can partly be answered by extending the authoring components of web 2.0 collaboration environments towards

handling more expressive representations of mathematical knowledge. Due to their web nature, such environments may, however, attract a larger community of users contributing highly diverse content than, e.g., a traditional repository of formalized mathematics, whose expert users have specialized tools installed on their own computers and contribute content of similar structure and expressivity. Therefore, the editor has to remain accessible to non-experts and has to support the full range of expressivity of the underlying exchange language, at least by gracefully degrading to generic text/XML editing on structures not specially supported. User-friendly editors for expressive languages have been developed before, but mainly for restricted use cases in a desktop setting. For example, the  $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$  editor has been extended for editing OMDoc, which is interactively verified by the  $\Omega$ mega proof assistant and checked for notational consistency [WABo6; AFN+07; DSWo8]<sup>35</sup>. However, this editor is not technically suitable for integration into a web interface, and it makes certain assumptions about the formal and linguistic structure of mathematical theories and proofs, which would not make it a suitable choice for, e.g., annotating the formal structure of a Wikipedia article while leaving its presentation unchanged.

This breakdown of the initial, broadly phrased “Value” question finally enables us to reduce it to a more concrete formulation, which this thesis will directly address:

**Usability:** How can environments that integrate heterogeneous services for producing and consuming mathematical knowledge be made usable?

## 1.7 Structure and Contribution of this Thesis

In [section 1.2](#), I have set the high-level goal of developing the building blocks for an environment that supports users in collaboratively creating new mathematical and other STEM knowledge and expanding the “mental infrastructure” required for understanding and applying this knowledge. First steps towards this goal have already been made using web 2.0 and semantic web technologies, as reported in [section 1.4](#). Web 2.0 interfaces support collaboration and have already been adopted by a number of mathematicians. Semantic web technologies have the potential to improve information retrieval, knowledge exchange, and service integration. Their adoption for MKM has not been a complete success so far, but, in [section 1.5](#), I have argued that now, with the availability of collaborative web 2.0 interfaces and better semantic web libraries and tools, there is a good opportunity to retry the effort.

The following questions have not yet sufficiently been answered by MKM, web 2.0 and semantic web research (cf. [section 1.6](#)), and will be answered in the following chapters.<sup>36</sup>

**Knowledge:** How can knowledge be made reusable and comprehensible across knowledge bases? Chapters 2 to 5 present an improved language for representing and exchanging mathematical knowledge. In order to be widely applicable, this language, based on OMDoc, OpenMath, MathML, and RDF, bridges informal and formalized knowledge, and integrates

<sup>35</sup>In an unpublished talk given on 2008-12-08, MARC WAGNER compared this to the way a spell-checker in a word processor works.

<sup>36</sup>Adequately to the diversity of languages and services integrated with each other, these chapters also review specific state of the art and related work, complementing the high-level overview of mathematics on the Web provided in this chapter.

mathematical knowledge with its areas of application and the Web of Data. I point out the general applicability of this approach to settings where knowledge occurs in different structural dimensions and is represented in different degrees of formality.

**Workflows: How can workflows be transferred from one knowledge base to another one?**

[Chapter 6](#) takes a closer look at the process of collaborative MKM, describes concrete workflows – fixing minor errors in content and its presentation and discussing revisions – and usage scenarios – giving a lecture and managing a project – and reviews primitive services that support individual steps of these workflows. Chapters [7](#) and [8](#) introduce an architecture for integrating such services into a coherent environment – into the publishing frontend as well as into the knowledge base backend. SWiM, a semantic wiki prototype of the complete environment, is introduced in [chapter 9](#). While the concrete services focus on mathematics, the integration architecture only depends on the heterogeneity of these services and of the representation languages they natively understand.

**Authoring: How can expressive knowledge representations be authored collaboratively?** Authoring mathematical knowledge and other knowledge of similar structural complexity, as well as the related tasks of discussing problems with knowledge items and validating representations, are of particular interest in [chapter 6](#). Giving users quick, local access to an editor and propagating changes made in the editor to other components that use the same content are crucial requirements for effectively supporting management workflows that [chapter 9](#) addresses.

Up to that point, the focus is on providing effective solutions that answer these questions. But does the target audience benefit from these theoretical solutions and their technical realizations?

**Usability: How can environments that integrate heterogeneous services for producing and consuming mathematical knowledge be made usable?** This question is answered in [chapter 10](#), which reports on a usability evaluation of the SWiM wiki and its support for the above-mentioned knowledge management workflows and draws conclusions for the sparsely explored area of usability of environments that integrate heterogeneous knowledge-oriented services.

[Chapter 11](#) concludes the thesis by critically reviewing to what extent these questions have been answered – from a STEM point of view as well as compared to the state of the art of the Web 2.0 and Semantic Web –, and by laying out a roadmap towards disseminating the findings gained so far: I discuss concrete next steps towards the envisioned infrastructure for managing, understanding, and applying mathematical and other STEM knowledge, but also towards supporting scientists in collaboratively gaining new knowledge.

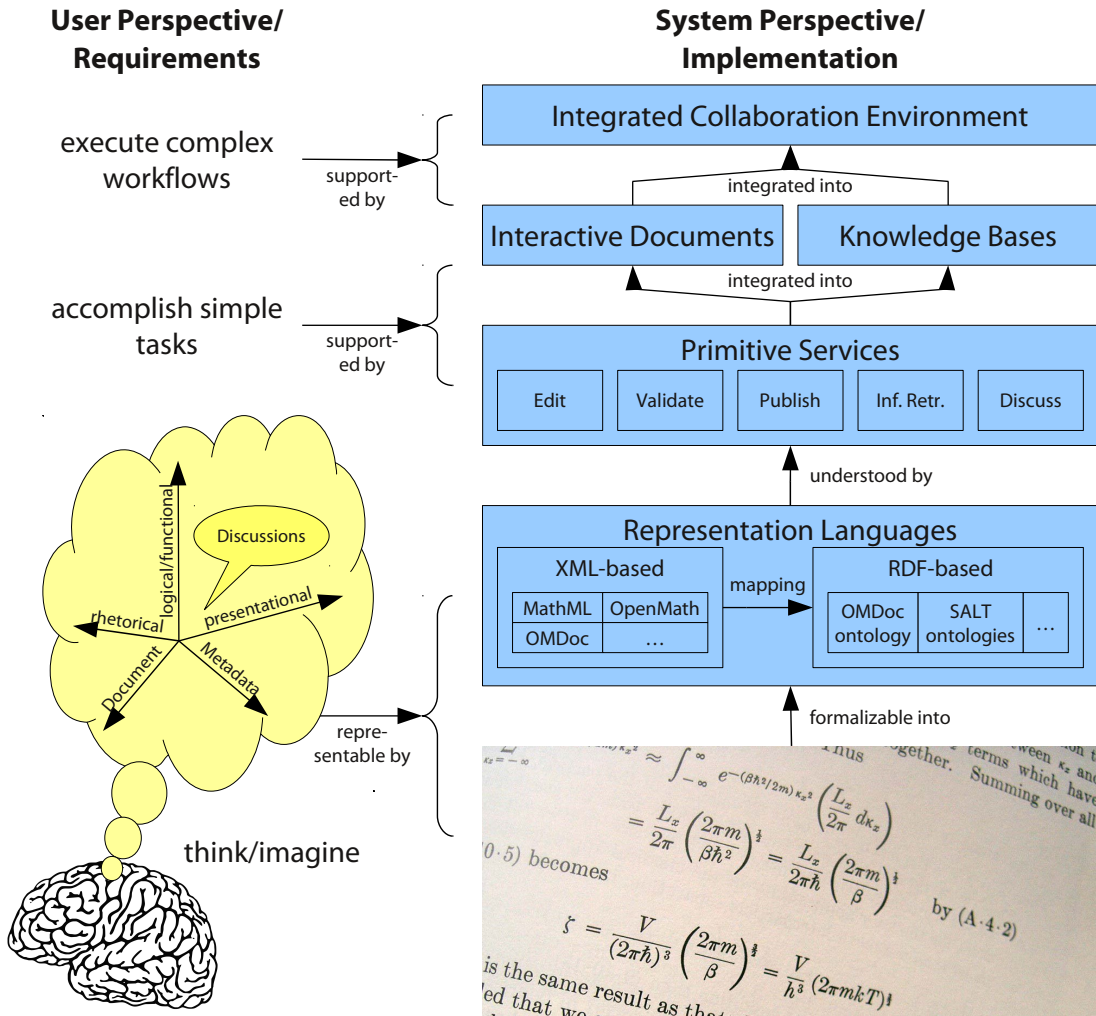


Figure 1.3: Use cases and system components covered in this thesis



## **Part II**

# **Knowledge Representation**





---

The initial discussion of how to effectively enable MKM based on web 2.0 and semantic web foundations (cf. [section 1.6](#)) has emphasized the central role of the language used for representing and exchanging mathematical knowledge – between different services, knowledge bases, and users.

**Chapter 2** reviews the structures of mathematical knowledge and establishes requirements for representing them in a reusable and exchangeable way. A review of state-of-the-art representations of mathematical knowledge enables us to pick the best candidates for satisfying these requirements: OMDoc and RDF(a) each satisfy half of the requirements.

The contribution of this thesis to representing mathematical knowledge is the combination of OMDoc with RDF(a), which has been pursued in three tracks covered by the following chapters:

**Chapter 3:** Mathematical knowledge represented in OMDoc has been made comprehensible to a wider range of services by providing an RDF vocabulary, formalized as an ontology, which captures the conceptual model of all structural dimensions of OMDoc. This also enables annotation of mathematical structures in human-comprehensible documents.

**Chapter 4:** OMDoc itself, with its good coverage of the structural dimensions of mathematical knowledge and its ability to combine logically heterogeneous, modular formalizations with informal documentation, can be used as a language for implementing and documenting ontologies and metadata vocabularies and integrating them with each other.

**Chapter 5:** RDFa, an embedded representation of RDF in XML, has been integrated into the OMDoc language to allow for coherently expressing all mathematical and related knowledge in the same language and linking mathematical documents to related external resources.



## Representing Mathematical Knowledge

A central goal of this thesis is to create an interoperability layer that supports the reuse of mathematical knowledge across knowledge bases and makes it accessible to a large number of services that support collaborative creation, organization, understanding, and application of mathematical knowledge.

In order to design such a layer, we first have to understand what structures mathematical knowledge can have. Mathematical knowledge comes in multiple structural dimensions and can be represented in different degrees of formality, as [section 2.1](#) shows. The degrees of formality range from presentations appealing to human readers to fully formalized data structures used for automated reasoning. Realistic applications, even in pure mathematics, do not only operate on the [logical and functional] structures of mathematical knowledge in the narrow sense, but also require information about non-mathematical aspects of the respective application scenario, about project organization and management, about discussions that authors and users hold *about* the mathematical knowledge, possibly about the social networks of these people, etc.

This review of structures enables us to establish precise requirements for reusably representing and exchanging mathematical knowledge (cf. [section 2.2](#)): (i) a good coverage of the above-mentioned structural dimensions, (ii) support for flexible degrees of formality, (iii) the possibility to interlink mathematical with non-mathematical knowledge, (iv) comprehensibility of the knowledge representation to human users and to a wide range of services, and (v) its compatibility with existing authoring and application environments.

Before we can analyze to what extent contemporary representation languages for mathematical knowledge already cover that diversity and then advance the state of the art in the following chapters, [section 2.3](#) provides an introduction to those [semantic] web technologies on which almost all of these languages are based: URIs/IRIs for globally identifying knowledge items, XML-based markup languages for encoding documents, RDF for web-scalable knowledge representation, and ontologies for formalizing the semantics of RDF vocabularies.

[Section 2.4](#) reviews a number of state-of-the-art representation languages for mathematical knowledge: MathML for formulæ, OpenMath for formulæ and for defining the vocabulary of symbols, from which formulæ are built, OMDoc as a richer language for mathematical vocabularies (theories) and documents, MathLang as an alternative language for formalizing documents,  $\text{\LaTeX}$

and semantics-oriented derivatives, markup languages for books and manuals, languages for formalized mathematics, and mathematical RDF vocabularies.

The result of this review, presented in [section 2.5](#), is that a combination of OMDoc and RDF(a) satisfies these requirements best. The following three chapters cover the three tracks of integrating OMDoc with RDF(a).

### 2.1 Structures of Mathematical Knowledge

*It is a melancholy experience for a professional mathematician  
to find himself writing about mathematics.  
The function of a mathematician is  
to do something, to prove new theorems, to add to mathematics,  
and not to talk about what he or other mathematicians have done.*

—G. H. HARDY [[Har40](#)]

There is little literature about the structures of mathematical knowledge in general. Working mathematicians often use them without reflecting on them (cf. the quote from HARDY above). Computer scientists developing software for MKM have to reflect on them, but they often do so from the point of view of a system specialized for a particular task – e.g. checking first-order logic proofs – and the particular conceptual model and representation language that system is based on. Thus, my understanding of the structures of mathematical knowledge is influenced by literature on concrete systems, models, languages, and ontologies<sup>1</sup>, from which I had to develop an abstraction by reverse engineering.

#### 2.1.1 General Concepts and Terminology

Before we can proceed with an overview of the structural dimensions of mathematical knowledge, I fix the meanings of some terms central to this thesis.

##### 2.1.1.1 Knowledge Items

So far, I have generally spoken of “[mathematical] knowledge”, and said that knowledge can occur in individual publications, but also in collections, such as digital libraries or, in more general terms, in knowledge bases or repositories. In traditional repositories, one document is the atomic unit of knowledge management. Entities below document level can be named and referenced (consider “section 2.1.1.1” or “definition 1”), but usually cannot be individually retrieved, edited, or annotated with metadata. Semantic Web technology, introduced in [section 1.3.3](#), supports allows for naming *arbitrary* entities – then called *resources*. I reserve the term “resource” for usage in a Semantic Web context; for general usage in this thesis, I define:

**Definition 1 (Knowledge Item)** A *[mathematical] knowledge item* is a uniquely identifiable item of temporal or permanent interest in some *[mathematical]* setting.

---

<sup>1</sup>For the purpose of this section, it is sufficient to consider an ontology a particular kind of knowledge model. [Section 2.3.4](#) covers ontologies, particularly their realization on the Semantic Web, and ontology engineering methodologies in more detail.

### 2.1.1.2 Degrees of Formality in Mathematics

This thesis assumes the MICHAEL KOHLHASE’s definitions of “informal”, “semiformal” and “formal” (see below). *Semiformal* does not denote a formalization level strictly *between* formal and informal, but a pragmatic compromise, possibly including both. I follow KOHLHASE’s definition of “*flexiforms*” but continue to use the more conventional adjective “semiformal”.

We will use the word **flexiform** as an adjective to describe the fact that a representation is of **flexible formality**, i.e. can contain both **informal** (i.e. appealing to a human reader) and **formal** (i.e. supporting syntax-driven reasoning processes) means.

Note that as defined here, the class of flexiforms is very broad, it includes arbitrary (informal) documents, datasets, and logical axiomatizations. We will pragmatically restrict the set of completely informal documents to those that are *intended to* or *could in principle* be (semi)formalized [...]. In particular, we include completely informal documents that are written with eventual (semi)formalization in mind as the starting points of a step-wise formalization process, first adding methodical and mathematical rigor, and then marking up formal elements.

Concretely, the class of flexiforms includes specifications from program verification, semantically annotated course materials, textbooks in the “hard sciences”, etc. (KOHLHASE [Koh10b])

In practice, there are many steps between “informal” and “formal”. Informality does not necessarily contradict rigorous style [GCo7], and symbolic notation is not necessarily formal. A symbolic formula may use ad hoc symbols that have not been defined, or ambiguous notations. On the other hand, rigorous natural language, often called “*mathematical vernacular*” [Bru87], has the potential to be understood by a machine. Finally, the adjective “*formalized*”, or sometimes “*computerized*”, is used for formal representations given in the native language of a symbolic computation engine, such as a CAS or a proof assistant.

In scenarios of managing, understanding, and applying mathematical knowledge, the *combination* of informal and formal representations is of particular interest. For example, in the GeoText system for managing geometry textbooks [Che10], each knowledge item (called “knowledge object” there) can have a natural language description and a diagram, which are presented to users, but also an algebraic representation for usage by a CAS. For effectively utilizing and maintaining the knowledge collection, e.g. checking its consistency, its structures are made explicit in that the knowledge objects have types and are categorized and interlinked [Che10]. Conversely, consider a software verification system: It primarily operates on computerized data structures, but explanations in natural language help the human author or user to understand the output of the system. The development of either kind of system can benefit from a language that bridges informal and formal representations.

### 2.1.1.3 Primary vs. Secondary Knowledge, Data vs. Metadata, Embedding vs. Linking

When knowledge is collaboratively maintained and when it is applied, not only that *primary* knowledge is of interest, but also *secondary* knowledge on how to maintain and apply the primary

one. A general definition of *metadata* seems applicable to data representing that secondary knowledge: “Metadata are data about data. They describe the content, quality, condition, and other characteristics of data. Metadata help a person to locate and understand data.” [Fgd]

However, the distinction between data and metadata blurs. Consider, for example, a group of researchers working on a publication (data) and discussing about it (metadata) in a LAKATOSIAN manner (cf. [section 1.1](#)): Suppose one participant in the discussion shows that a proof in the original draft does not cover all possible cases, and another participant contributes the idea of a generalized proof. If the authors consider that process of discovery worth communicating, they might preserve their discussion in the publication, e.g. by modeling its rhetorical structures. Thus, the metadata turn into data. Or take a technical view on the RDF data model (cf. [section 2.3.3](#)): Only the bare resources, identified by URIs, are data, whereas all of their properties and links to other resources are modeled as metadata. Metadata can be embedded into the data they describe, or point to the data from outside (“standoff markup”); both approaches can also be combined.

This thesis makes the following pragmatic restrictions: Logical/functional structures, rhetorical structures, document structures, and information on how to present them are considered as structures of primary interest, i.e. as *data* (see the following section for an overview and [sections 2.1.2](#) to [2.1.5](#) for details). The remaining structures of interest are divided into two classes by the ways they are used: (i) (proper) metadata for administrative purposes and for describing the application environment ([section 2.1.7](#)), and (ii) structured discourse about primary knowledge ([section 2.1.8](#)). Metadata are assumed to be embedded, whereas discussions are assumed to be external to the primary knowledge. Treating discussions as standoff data is adequate when anyone should be able to comment on a primary knowledge item. Conversely, embedding metadata allows for a uniform management workflow, where all people collaborating on a resource see the metadata at the same time, and where metadata are versioned together with the data. That helps to avoid synchronization and other maintenance problems such as incompleteness.

### 2.1.1.4 The Multi-Dimensionality of Mathematical Knowledge

Logical and functional structures – formulæ composed from symbols, whose properties are defined by axioms expressed in some logical language, or asserted and proven, – are probably the most important ones that mathematical knowledge has (cf. [section 2.1.2](#)), but not the only ones. The same knowledge can be expressed in natural language, which has rhetorical structures, such as one phrase giving evidence for another one ([section 2.1.3](#)). When knowledge is arranged in a document, e.g. for presentation in a talk or for publication on paper, it is put into a narrative order and grouped into sections. The structure of a document may reflect the logical/functional or rhetorical structures, but not necessarily. For example, prerequisites for understanding the main content may be provided in an initial section, but they can as well be listed in an appendix or cited from an external document ([section 2.1.4](#)). The way how text and formulæ are rendered is independent from the knowledge they convey. Be the definition of the binomial coefficient stated as  $\binom{n}{k} := n! / (k! \cdot (n - k)!)$  or as  $C_n^k \stackrel{\text{def}}{=} \frac{n!}{k!(n-k)!}$ , addressing different audiences, it has the same meaning ([section 2.1.5](#)). Furthermore, there are different kinds of knowledge *about* this primary knowledge, as mentioned in the previous sections.

Changing a representation or enhancing its formality w.r.t. one of these structural categories does not necessarily affect the others: For example, choosing the notation  $2^S$  instead of  $\mathcal{P}(S)$  for the power set of a set  $S$  may suggest to a human reader that the power set contains  $2^{|S|}$  elements, but unless we added a sufficiently formal definition of the power set to our knowledge base, an automated reasoner would not be able to make that inference. Or suppose the user I. M. takes an informally stated theory of functions on pairs of non-Riemannian hypersquares<sup>2</sup> and puts it on a type-theoretical foundation in order to be able to use a certain automated reasoner: That change to the logical/functional structures in our knowledge base might affect our future notational preferences in that we would prefer to write the type of a certain function as  $H \rightarrow H \rightarrow \mathbb{R}$  rather than  $H \times H \rightarrow \mathbb{R}$ , which would be more common given a set theory background. But to an average repository management facility running under the hood of our knowledge base, the whole meaning of that change will be that the user I. M. committed a new revision of some file at some time, unless he provided more detailed annotations, i.e. metadata describing the change.

These examples – and studies of real knowledge collections, as we have conducted with software engineering documents in [KKL10a] – show that the different kinds of structures only interact lightly and can therefore be considered *independent dimensions of a formality space*.

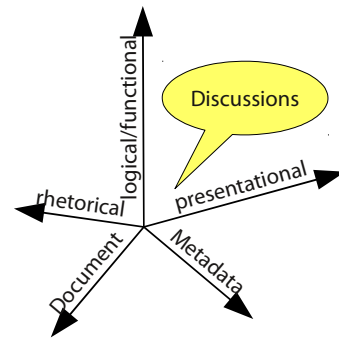


Figure 2.1: Dimensions of mathematical knowledge – largely orthogonal, partly interdependent

### 2.1.2 Logical and Functional Structures

*A mathematician is a person who can find analogies between theorems;  
a better mathematician is one who can see analogies between proofs  
and the best mathematician can notice analogies between theories.  
One can imagine that the ultimate mathematician is one  
who can see analogies between analogies.*

—attributed to STEFAN BANACH

First and foremost, mathematical knowledge has a three-layered *logical* structure of objects – composed of symbols –, statements, and theories<sup>3</sup>. Symbols comprise operators, functions, sets, and constants. New mathematical concepts (i.e. symbols) can be defined, possibly based on concepts defined previously. Mathematical *objects* comprise single symbols or compounds, such as a complex number, an application of a function to arguments, or a derivative. Some of their properties are specified as axioms. Axioms are expressed as formulae in a certain logical language, such as first-order logic (FOL). By applying rules of that logic, other properties of the mathematical concepts can be inferred. In a usual mathematical document, such properties are first asserted and then proven – or refuted. Often, the choice of what properties of a concept to model as axioms is arbitrary and merely follows established conventions.<sup>4</sup> All kinds of properties of concepts are

<sup>2</sup>an imaginary mathematical concept, which is of interest to some ideal mathematician described in [DH81]

<sup>3</sup>reusing the terminology introduced by MICHAEL KOHLHASE in [Koh06b, chapters 2.3 and 3.2] and refined in [KKo8]

<sup>4</sup>For example, a number of theorems that follow from the axiom of choice have subsequently been proven equivalent to the axiom of choice. That is, one could alternatively assume one such theorem (e.g. ZORN's lemma) as an axiom

sometimes subsumed under the term *statements*. This is the case in OMDoc (cf. [section 2.4.4](#)), which distinguishes symbol declarations and axioms, definitions<sup>5</sup>, assertions – theorems, lemmas, corollaries, etc. –, proofs – which prove assertions by applying inference rules to axioms and previously proven theorems –, and examples. Not all assertions in a realistic mathematical knowledge base need to be true: There can be conjectures whose truth is not yet known, as well as wrong assertions that have been refuted by counter-examples but are kept for instructive purposes. Groups of closely related symbols and their properties form *theories*. When reusing mathematical symbols, their names are often qualified by their theory for disambiguation, i.e. theories act as namespaces for symbols; this is also reflected by speaking of the “home theory” of a statement. For example, both the theory of real numbers and the theory of functions on real numbers have an “addition” operator. While the latter can be defined pointwise in terms of the former, both remain different; for example, one cannot use either of them to add a number to a function.

In the context of theories, statements can be classified more precisely:

We view axioms and definitions as *constitutive* for a given theory, since changing this information will yield a different theory (with different mathematical properties [...]). Other mathematical statements like theorems or proofs that support them are not constitutive, since they only illustrate the mathematical objects in the theory by explicitly stating the properties that are implicitly determined by the constitutive statements. ([Koho6b, chapter 15.1])

Moreover, the logical language used to express the statements in a theory can itself be modeled as a theory, then called *meta-theory* [Rabo8; RK11]. For example, the theory of commutative groups can be formalized with FOL as a meta-theory. FOL provides the universal quantifier that is needed for stating the axiom of commutativity as  $\forall a, b \in G. a \circ b = b \circ a$ .

In mathematical logic, a theory is simply the deductive closure of a set of axioms, that is, the – often infinite – set of logical consequences of the axioms. For knowledge management tasks, which involve, e.g., reuse of theories or management of theory changes, it is, however, reasonable to model theories as structures of their own, which are more powerful than mere namespaces. It has been found particularly useful to build theories on a minimal set of axioms and model a whole field of mathematics as a strongly interconnected graph of “little theories” reusing each other (cf. [FGT92]). The connections are called *theory morphisms* or *views*. Some of them are given by definition – then called *imports* –, others are postulated and then have to be proven. A morphism maps a symbol from a source theory to a symbol – or sometimes also a more complex expression – of a target theory.<sup>6</sup> Development graphs extend this model by a more expressive way of proving asserted theory morphisms by decomposing them into sets of paths in the theory graph (see, e.g., [Koho6b, chapter 18.5] for OMDoc’s representation of development graphs). Development graphs

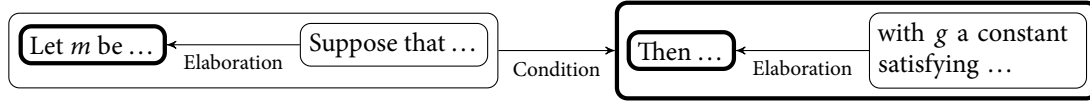
---

and from that derive the axiom of choice as a theorem. However, it is still the axiom of choice that continues to be stated as an axiom. [Bel09]

<sup>5</sup>A definition is a variant of an axiom that completely fixes the meaning of a symbol. An appropriate axiom can also do that, but definitions commonly occur in informal, textbook-style mathematics and are therefore supported by many representation languages.

<sup>6</sup>For example, the theory of integers can be linked by a view  $\{\circ \mapsto +, e \mapsto 0\}$  to the theory of monoids, where  $\circ$  is the binary operation and  $e$  the unit element of the monoid. For a more in-depth elaboration on theory morphisms and views, see [Rabo8].



Figure 2.2: RST markup of [example 2](#) (nuclei with thick outline)

help to formally model *dependencies* (cf. [section 2.1.6](#)) in structured specifications used, e.g., for software verification, and thus facilitate management of change tasks [[AHM+06](#)].

Logical/functional structures can be expressed at different *levels of formality*: Often, an author starts a document by sketching a few formulæ and some textual notes. Later, the content is elaborated both into the formal and into the informal direction: A sloppy formula is written more rigorously, rigorous text is formalized, taking previously formalized knowledge into account, and natural language explanations are added to formalized knowledge (see, e.g., [[Koho06b](#), chapter 4]). Both directions can, in principle, be automated: *Natural language processing* techniques can aid formalization (see, e.g., [[GJA+09](#)]), whereas proof explanation helps to generate natural language from formalized knowledge (see, e.g., [[Fie01](#)]). These solutions, however, can not yet cope with the full complexity of mathematical knowledge as it occurs in practice. Particularly the automated disambiguation symbolic notation (cf. [section 2.1.5](#)) is hard, as the surrounding text often has to be taken into account for disambiguation [[GJA+09](#)]. This thesis does not cover automated translation.

### 2.1.3 Rhetorical Structures

The more natural language a mathematical text contains over formulæ, the more important become *rhetorical structures*. Mathematics has developed its own style of natural language. CLAUS ZINN has analyzed the natural language of mathematical proofs in depth [[Zin04](#)], partly building on JERZY TRZECIAK’s style guide on mathematical writing. The latter lists typical phrase patterns such as the following:

#### Example 2 (Formulating a Theorem, after TRZECIAK)

Let $M$ be .....	<table border="0"> <tr> <td rowspan="3"> <table border="0"> <tr><td>Suppose that .....</td></tr> <tr><td>Assume that .....</td></tr> <tr><td>Write .....</td></tr> </table> </td> <td rowspan="3"> <table border="0"> <tr><td>. Then .....,</td></tr> </table> </td> <td>provided <math>m \neq 1</math>.</td> <td rowspan="3">[<a href="#">Trz95</a>]</td> </tr> <tr> <td>unless <math>m = 1</math>.</td> </tr> <tr> <td>with <math>g</math> a constant satisfying .....</td> </tr> </table>	<table border="0"> <tr><td>Suppose that .....</td></tr> <tr><td>Assume that .....</td></tr> <tr><td>Write .....</td></tr> </table>	Suppose that .....	Assume that .....	Write .....	<table border="0"> <tr><td>. Then .....,</td></tr> </table>	. Then .....,	provided $m \neq 1$ .	[ <a href="#">Trz95</a> ]	unless $m = 1$ .	with $g$ a constant satisfying .....	
			<table border="0"> <tr><td>Suppose that .....</td></tr> <tr><td>Assume that .....</td></tr> <tr><td>Write .....</td></tr> </table>	Suppose that .....	Assume that .....		Write .....	<table border="0"> <tr><td>. Then .....,</td></tr> </table>		. Then .....,	provided $m \neq 1$ .	[ <a href="#">Trz95</a> ]
				Suppose that .....								
Assume that .....												
Write .....												
. Then .....,												
unless $m = 1$ .												
with $g$ a constant satisfying .....												

General models for representing the discourse structure of a text can also be applied to mathematical texts. Rhetorical Structure Theory (RST), for example, which intends to offer “*an explanation of the coherence of texts*” [[MT](#)], divides a text into spans, often down to the level of subordinate clauses. RST has a rich vocabulary of relations between *nuclei* (essential text spans) and *satellites* (spans that provide additional information); for example, a satellite can give evidence to a nucleus, provide background information to facilitate understanding, or define the context in which the nucleus is to be interpreted. [Figure 2.2](#) models a phrase from [example 2](#) according to RST.

For sections and chapters on the upper levels of a document, several models of discourse in scientific publications have introduced more convenient coarse-grained blocks that correspond to the usual sections of a publication, and reserve RST for fine-grained markup [[GHC+09](#)]. A typical document in one of these models starts with an abstract and a motivation and ends with a conclusion and a list of references, and has some sections in between that provide background knowledge,

explain the actual contribution of the paper, demonstrate practical applications, summarize the results of experiments or evaluations, review the state of the art and related work, etc.

Some rhetorical structures in mathematical text directly correspond to logical structures. The phrase-level example given above is the natural language equivalent of a formal theorem. The background and contribution sections of a research paper on a formal topic could directly correspond to mathematical theories.

### 2.1.4 Document Structures

*Documents* arrange the knowledge they contain in a linear, narrative order suitable for sequential consumption (on paper, in a talk, etc.); they consist of chapters, sections, paragraphs, and references to [sections of] other documents. Document structures often loosely correspond to mathematical or rhetorical structures, but not necessarily. Sections of a document may correspond to rhetorical blocks of the whole document like introduction, conclusion, etc., but the document could also have been divided into parts that have no rhetorical meaning but were just created to meet technical restrictions. Consider an upper limit for the number of the pages of a scientific publication, or figures arranged in a way that beautifies the page layout but does not reflect what logical unit of the document they belong to. [Section 2.1.1.4](#) provides another example of where the order of a document differs from logical or rhetorical dependencies.

*Reuse* scenarios, on the other hand, encourage the creation of small document snippets that form semantic units (mathematically or rhetorically): They would be kept in a “content commons” – a term coined for the Connexions repository of educational content (cf. [section 1.4.2](#)) –, and when creating documents for end users – such as lecture notes for students –, authors would pick suitable snippets and remix them into a narrative order<sup>7</sup>. The narrative document would consist of a scaffolding text outlining the large rhetorical blocks, but obtain most of the actual content by inclusion of content commons snippets, only providing transitional texts between successive included snippets.

### 2.1.5 Presentation and Notation

Mathematical objects employ a two-dimensional notation, whose complexity is owed to the possibility to define new symbols at will. Choosing an intuitive notation for the concepts dealt with is of great importance to understanding and communication, as pointed out in [section 1.1](#). The notation of a symbol is usually introduced with its first declaration, typical phrases being “We will denote by  $Z$  the set ...”, “The notation  $aRb$  means that ...”, etc. [[Trz95](#)]. Notation can be conceived as a one-to-many mapping of structures of mathematical knowledge – primarily logical/functional structures – to an arrangement of glyphs on paper. The notation chosen for a particular object in a particular document is determined by a number of presentation context dimensions (examples taken from [[MGL+09](#)] and [[Mül10a](#)]):

**language and culture:** the French/Russian notation of the binomial coefficient  $C_n^k$  vs. the German/English notation  $\binom{n}{k}$ ; see [[Lib10a](#)] for details

---

<sup>7</sup>See [[Mül10a](#)] for an in-depth treatment of composing narrative mathematical documents from reusable units.

**level of expertise:** the explicit notation of multiplication as  $a \cdot b$ , which is common in primary school, vs. the more advanced omission of the operator symbol in the notation  $ab$

**area of application:** The square root of  $-1$  is written as  $i$  in most fields, whereas electrical engineers write it as  $j$  to distinguish it from the current  $I$ .

**community of practice:** People with a set theory background tend to include  $0$  in the set of natural numbers  $\mathbb{N}$ , whereas those with a number theory background tend to start with  $1$ .<sup>8</sup>

**individual preference:** Some mathematicians, who prefer completely idiosyncratic notations when working on their own, translate other articles into their own notation and translate their own articles to a more conventional notation before publication [Heioo, 166–167].

Figure 2.3 gives an instructive example of how Connexions, a state-of-the-art educational CMS (cf. section 1.4.2), accounts for this variety – albeit with a hard-coded repertoire and without an explicit notion of presentation contexts, which more MKM-oriented systems have, such as ActiveMath [Act; MGL+09] or JOMDoc [Jom; Mül10a].

The greatest notational variety has been observed for mathematical symbols. From the level of statements upwards, notation is more standardized and therefore usually not a subject of research. For example, it is common to start a definition with the boldface word “Definition” – translated into the respective language of the text –, followed by a number and optionally the name of the concept defined. There may be minor differences in how to number definitions, or what font to use for the word “Definition”, but they are largely independent from the particular field of mathematics.

However, *which* definition of the same concept, which proof for the same theorem, which example for the same thing, i.e. which *representation* of a thing to present to a user – these questions do depend on context. ANDREA KOHLHASE and MICHAEL KOHLHASE have pointed out the context-sensitivity of examples in their study of *framing* practices, i.e. “*view[ing] objects of interest in terms of already understood structures*” [KK09c]. Their model of framing relies on a formalization of logical structures – concretely: theory graphs –, whereas CHRISTINE MÜLLER has realized the generation of documents from mathematical knowledge items (“content planning”) using contextual information independent from the logical structure [Mül10a].

### 2.1.6 Dependencies

In any of the above-mentioned structural dimensions, there can be dependencies among knowledge items. Dependency relations and their application to change management have been explored in software verification (cf. the remark on development graphs in section 2.1.2) and software engineering [Mad92]. NORMEN MÜLLER, building on these explorations, introduced a change management approach for general semi-structured documents [Mül10b]: For any document format or, if necessary, even for particular instance documents, one has to formally define equivalence – in certain formats, e.g., the order of certain items does not matter – and dependencies. That

<sup>8</sup>This can also be considered a difference w.r.t. the area of application. For example, in theoretical computer science it is advantageous to include  $0$ , as many of the required induction proofs start at  $0$ , whereas negative integers are rarely needed.

## Collection: Test Course 1

Hide sidebars

In: [Personal Workspace](#)

Contents

Metadata

Roles

Parameters

Preview

Publish

## Set collection parameters

Please select display parameters for your course. The defaults are in the first column of each row.

Print Parameters			
Font	<input checked="" type="radio"/> Computer Modern AaBbCc123	<input type="radio"/> Times AaBbCc123	<input type="radio"/> Palatino AaBbCc123
Font Size	<input checked="" type="radio"/> 10pt		<input type="radio"/> 12pt
Paper Size	<input checked="" type="radio"/> 8.5" x 11"		<input type="radio"/> 6" x 9"
Paragraph Spacing	<input checked="" type="radio"/> Compact		<input type="radio"/> Loose

Calculus Notation		
Vector Notation	<input checked="" type="radio"/> $\mathbf{v}$	<input type="radio"/> $\vec{v}$
Scalar Product Notation	<input checked="" type="radio"/> $\langle A, B \rangle$	<input type="radio"/> $A \cdot B$
Curl Notation	<input checked="" type="radio"/> $\text{curl } A$	<input type="radio"/> $\nabla \times A$
Gradient Notation	<input checked="" type="radio"/> $\text{grad } A$	<input type="radio"/> $\nabla A$

Miscellaneous Notation			
And / Or Notation	<input checked="" type="radio"/> $\wedge, \vee$	<input type="radio"/> and, or	<input type="radio"/> $\&,  $
Real / Imaginary Notation	<input checked="" type="radio"/> $\Re, \Im$	<input type="radio"/> Re, Im	
Conjugate Notation	<input checked="" type="radio"/> $\bar{H}$	<input type="radio"/> $H^*$	
Imaginary I Notation	<input checked="" type="radio"/> $i$	<input type="radio"/> $j$	
Equation Layout <sup>1</sup>	<input checked="" type="radio"/> $\forall_x x \in [0,1,2,3] : y = x$	<input type="radio"/> $y = x$ for $x = [0,1,2,3]$	
Mean Notation	<input checked="" type="radio"/> $\bar{m}$	<input type="radio"/> $\langle m \rangle$	
Remainder (Modular Division)	<input checked="" type="radio"/> $a \bmod b$	<input type="radio"/> $\langle a \rangle_b$	

Update Properties

<sup>1</sup>The two choices are mathematically equivalent, yet are very different in presentation.

Figure 2.3: Connexions: configuring the presentation of a collection of course modules

allows for an automated identification and classification of changes to items and an analysis of their impacts on other items depending on them. This thesis assumes dependencies to be *propagation paths of changes*, a notion introduced in [Mad92; Müll0b]:

**Definition 3 (Dependency)** A knowledge item  $B$  depends on  $A$  in the way  $d_p$  (notation:  $A \xrightarrow{d_p} B$ ) iff a change to  $A$  may have an impact on the property  $p$  of  $B$ .

To make this definition precise, one has to fix the property  $p$ . Different conceptual models and representation languages have done that in different ways. Their notions of dependency are summarized here to point out the differences; otherwise see [section 2.4](#) for the full descriptions of the respective languages.

In the formal logical setting of the MMT language (cf. [section 2.4.4](#)), FLORIAN RABE chose the property of (logical) well-formedness to describe dependencies between MMT declarations [Rab08, chapter 8.4]. In the semiformal setting of MathLang (cf. [section 2.4.6](#)), which targets the formalization of narrative text, KRZYSZTOF RETEL et al. chose the reader’s ability to understand a knowledge item [Ret09; KMR+07]. From a formal point of view, this is a fuzzier concept, but, from an application point of view, it is more general in that it covers not only comprehension by automated reasoning tools but also by humans. This human-friendly notion of dependency is relevant in educational settings, where dependencies are often called prerequisites [to understanding the current knowledge item]; see, e.g., [Ullo8]. While MMT’s notion of dependency extends to the theory level, which MathLang does not, MathLang covers informal structures, such as examples, which MMT does not yet take into account. For instance, an example for some knowledge item  $I$  is considered to depend on  $I$ . This is consistent with [definition 3](#) because certain changes to  $I$  might have the impact of turning the example meaningless.

It is not always trivial to characterize the kind and direction of a dependency induced by a relation – consider the relation of a proof to the assertion it proves: (i) Changing the assertion may turn its proof meaningless, but, conversely, (ii) changing the proof may affect what a system knows about the truth of the assertion. Note, however, that the dependency relation (i) is different from (ii): In terms of [definition 3](#), the property  $p$  affected by changing the assertion in direction (i) is the meaningfulness [of the proof], whereas the property affected by changing the proof in direction (ii) is the truth [of the assertion], as it is known to an ontology-driven system. From these examples, we can conclude that there is not a single type of dependency, and that different dependency types are relevant in different application scenarios.

## 2.1.7 Metadata

[Section 2.1.1.3](#) has introduced the general distinction between data and metadata. The following definition of metadata specifically takes into account the role of metadata in MKM: “*Metadata is the set of annotations that serve to facilitate the administration of the libraries of mathematical knowledge, the search and retrieval of mathematical knowledge and the reuse of the knowledge by different mathematical applications.*” [Gogo3a] Earlier works on metadata for MKM, published in the course of the MoWGLI project, distinguish three categories of metadata [Gogo3a]:

**Administrative** metadata describe the lifecycle and revision history of a resource, the data format and the usage requirements, copyright information, as well as other general-purpose information.

**Mathematical:** classifications of mathematical knowledge items, and relations among them. In the past, models for mathematical metadata mainly focused on logical structures. Expressive state-of-the-art markup languages, such as OMDoc (cf. [section 2.4.4](#)) or MathLang (cf. [section 2.4.6](#)), promote a significant share of the representation of logical or rhetorical structures to *data* and allow for modeling them in a more elaborate way than metadata. Therefore, this section does not cover them.


**Application-specific** metadata that describe how to use mathematical knowledge in an application, for example educational annotations for e-learning applications, or that describe non-mathematical domain knowledge that is related to mathematical knowledge in a specific application environment. Some of these can also be considered administrative.

After some short remarks on using metadata and their interaction with other structural dimensions, a review of relevant metadata vocabularies follows. As with the structural dimensions of mathematical knowledge in general, metadata vocabularies are diverse, and a realistic application has to combine multiple of them. Most of these vocabularies have a concrete implementation as an XML schema or an ontology (cf. [section 3.4](#)).

### 2.1.7.1 Granularity of Metadata in Mathematical Knowledge

The usage of metadata is not limited to a particular structural level of mathematical knowledge. A mathematical theory might be annotated as “easy to learn”, except for the proof of one theorem. In a textbook covering one topic, one chapter might deviate into a different topic. A remix of document snippets is often created by an author different from the authors of the snippets and may be subject to a different license. One can even annotate subexpressions of mathematical objects with their own metadata, as is often done on paper or on the blackboard:

$$b^{-1}(\boxed{a^{-1}a})b = b^{-1}(eb) = \dots$$


 We learned that last week

### 2.1.7.2 Propagation of Metadata Along Other Structural Dimensions

Metadata may propagate up or down along (other) structural dimensions of mathematical knowledge. For example, the author of a document is also the author of any of its subsections, unless stated otherwise. Conversely, anyone who contributed to a subsection also contributed to the whole document. [Table 2.1](#) lists a few other examples. Systems that support metadata propagation have been found useful in MKM, as they relieve content authors from duplicate work (cf. [\[Lib09\]](#)). Metadata propagation along whole→part relations in the dimension of document structures – including collections of multiple documents – has so far been studied best for mathematical

Table 2.1: Examples for the propagation of metadata

Metadata field	Propagates along	Relation/direction	Policy
author	document	whole→part	if missing
subject classification (major)	any	whole→part	if missing
license	any <sup>a</sup>	whole→part	if missing
license	revision history	past→future	depends on license <sup>b</sup>
contributor	document	part→whole	merge
date of last revision	document	part→whole	replace
subject classification (minor)	any	part→whole <sup>c</sup>	merge

<sup>a</sup> think, e.g., of a formalized knowledge base as an example for licenses in the logical dimension

<sup>b</sup> When a resource has been published under a share-alike license once, all future revisions must also be published under the same or a compatible license.

<sup>c</sup> compare author vs. contributor

knowledge (cf. [Lib09] and [Koho6b, chapter 12.4]). The ActiveMath system distinguishes three inheritance policies for metadata: “if-missing” inherits a metadata field unless it is already present, “merge” forms the union set of multi-valued metadata fields, and there is an explicit policy to inherit “nothing” [Lib09]. I have built on these results and leave the investigation of metadata propagation along other structural dimensions and other relations, such as dependency, to future work. One could, for example, investigate along which axes difficulty – in an educational sense – propagates. It is reasonable to assume that it propagates along part→whole relations in the logical dimension, e.g. from an axiom to the containing theory, but possibly also along dependency relations.

### 2.1.7.3 General-purpose Metadata

The most widely used metadata vocabulary across domains is the Dublin Core Metadata Element Set (DCMES [Dcm]). It is often used to enable retrieval; many search engines index Dublin Core metadata fields. For any type of resource (commonly a document), it provides those metadata properties that are most frequently needed, including authorship (distinguished by *creator* vs. *contributor*), content descriptions (*title*, *subject*, *description*), *date*, and *language*. The MARC relators vocabulary has a more detailed notion of roles of creators and contributors, for example “author”, “editor”, or “translator”, and can be combined with Dublin Core metadata [Mar; DCM05].

Dublin Core metadata have a weak semantics but form a common denominator of many semantic web applications. They are often complemented by more specific metadata, such as the ones mentioned below. For example, the value of the *dc:subject* field can be from a domain-specific classification scheme (see below), the value of the *dc:coverage* field, denoting the spatial or temporal topic or jurisdiction a resource covers, can be from a vocabulary of geographic identifiers, or *dc:rights*, denoting property rights, can point to a formal description of a license. These general metadata are mostly embedded into the resource, at least when exporting the resource from a



database. This may have legal reasons, e.g. the license of a resource requiring all authors to be mentioned explicitly when redistributing the resource.

The DCMI Metadata Terms vocabulary [DCMo8] is a modernized and extended but backwards-compatible superset of the DCMES. It has adopted more recent best semantic web best practices, allows to describe the aspects covered by the DCMES more precisely (e.g. by introducing a proper distinction of the above-mentioned *coverage* aspects) and covers additional aspects of metadata, such as the section structure of the document (which is covered as a structural aspect of its own in this thesis, cf. [section 2.1.4](#)) and revision histories (see below).

### 2.1.7.4 Classification Schemes

Many domains have specific classification schemes. MSC (Mathematics Subject Classification [Msc]) is the one prevalent in mathematics. It assigns an alphanumerical code to resources (usually research papers). This thesis would be classified as 68T35 or 68T30, where 68 is computer science, 68T is artificial intelligence, 68T35 is “languages and software systems (knowledge-based systems, expert systems, etc.)”, and 68T30 is “knowledge representation”. For computer science, there is the ACM Computing Classification System (CCS [Acm]), in which this thesis could be classified in the following categories:<sup>9</sup>

- F.4.m** “Theory of Computation” → “Mathematical Logic and Formal Languages” → “Miscellaneous”
- G.4** “Mathematical Software” → “Documentation”
- H** “Information Systems”
  - H.3.5** “Information Storage and Retrieval” → “Online Information Services”, and then “Web-based services”
  - H.5** “Information Interfaces and Presentation”
    - H.5.3** “Group and Organization Interfaces”, and then “Computer-supported cooperative work” and “Web-based interaction”
    - H.5.4** “Hypertext/Hypermedia”, and then “Architectures”
- I** “Computing Methodologies”
  - I.2** “Artificial Intelligence”
    - I.2.1** “Applications and Expert Systems”, and then “Medicine and science”
    - I.2.4** “Knowledge Representation Formalisms and Methods”, and then “Representation languages” and “Semantic networks”
    - I.2.6** “Learning”, and then “Knowledge acquisition”
  - I.7** “Document and Text Processing”
    - I.7.1** “Document and Text Editing”, and then “Document management”
    - I.7.2** “Document Preparation”, and then “Format and notation”, “Hypertext/hypermedia”, “Languages and systems”, “Markup languages”, and “Standards”
- J.2** “Computer Applications” → “Physical Sciences and Engineering”, and then “Mathematics and statistics”
- K.4.3** “Computing Milieux” → “Computers and Society” → “Organizational Impacts”, and then “Computer-supported collaborative work”

---

<sup>9</sup>Surprisingly, this thesis is harder to classify in the ACM CCS – which, however, is much older than MSC 2010.



There are also classification schemes for things more specific than subjects of interest. GAMS, the Guide to Available Mathematical Software, features a classification scheme for mathematical problems, such as H2a1 = “one-dimensional finite interval quadrature”, combined with a directory of software that solves such problems [Gam].

### 2.1.7.5 Licensing

Licensing metadata describe the legal circumstances under which the original data may be reused, e.g. when importing them into another knowledge base, using them for educational purposes, or deriving a software implementation from a mathematical model. Embedding licensing metadata into the resources they describe is advisable, as it makes them more apparent to the users of the resources.

The Creative Commons Rights Expression Language (ccREL [AAL+08]) expresses copyright licensing terms such as permissions and restrictions that apply when (re)using a resource, whether a resource may be used commercially, whether modified version may be distributed, etc. This is of particular relevance in open collaborative web environments (cf. [section 1.3.2](#)). Creative Commons licensing metadata are supported by certain search engines, including Google and Yahoo. While ccREL is specifically tailored to open content licenses, the Open Digital Rights Language (ODRL [ODRb; Iano2]) has a much wider scope but is more complex.

### 2.1.7.6 Versioning

Put into a wider application context, semiformal mathematical knowledge also comprises technical specifications and ontologies (cf. [chapter 4](#)). For example, we have used OMDoc for formal specifications of safe and secure technical devices [KKL10a]. Once other technical systems, such as software, are based on such a [semi]formalization, compatibility considerations make versioning a requirement. As another example from engineering, consider an engineer implementing a specification: He may just have a printed copy of the latest revision, but no access to the original repository. Nevertheless, he needs certain provenance information: what author is responsible for some change of the specification, and whether or when that change has been approved by a certification agency. Similarly, environments for collaborative authoring usually archive previous revisions of the artifacts authored, so that the history of an artifact can easily be retraced, or that an old revision can be restored to revert an erroneous change. While a revision log for a document can be obtained from a versioned repository on the server side, embedding a revision log persistently at least into documents *exported* from the repository may be a legal requirement.<sup>10</sup>

The above-mentioned DCMI Terms vocabulary covers basic versioning aspects: One resource can have older versions, and any old version can be replaced by a newer one. The SIOC model (Semantically Interlinked Online Communities, cf. [BBP+08; BBD+10; BB07] and [figure 3.5](#) on page 129) for user-generated online content has a slightly richer versioning vocabulary inspired by models representing the structure of wikis [OP09]. An additional SIOC module allows for

<sup>10</sup> A well-known example used to be exporting a Wikipedia article for external reuse. The GNU Free Documentation License, which had been used until 2009, required to state the names of all authors, and thus all of them had to be listed in the metadata record of the exported document. (Actually, automated support for that was not implemented.)

describing actions of manipulating digital artifacts, including the creation of a revision being a special case [CP10]. In the context of versioned repositories for software engineering and ontology engineering, more elaborate vocabularies have been developed, such as the ModelDriven.org Architecture model [Mod], whose versioning module covers changes to generic structured or unstructured “data assets”, or the Ontology Metadata Vocabulary (OMV [HPH+; PHC+09]), which has an extension modeling changes to ontologies.

### 2.1.7.7 Application-specific Metadata

In science, technology, and engineering, where the mathematical model is only a part of the whole application environment, there is obviously also non-mathematical knowledge that needs to be modeled. But even effective application and collaborative maintenance of pure mathematical knowledge relies on non-mathematical information – for example about user interaction or maintenance workflows, which are easier to verify, maintain, and port when modeled formally.

As this thesis is not restricted to a particular application of mathematical knowledge, this section concludes with three brief examples. The review of application domain ontologies in [section 3.5.2](#) mentions further ones.

**Education:** E-learning environments, such as ActiveMath or MathDox, use educational metadata to describe properties such as the level of difficulty or interactivity of a knowledge item, its coverage of topics (e.g. in terms of classification systems), and its intended audience. ActiveMath uses Learning Object Metadata (LOM [IEE02a]) for that purpose. Furthermore, it supports markup for exercises<sup>11</sup>, covering aspects such as interaction steps, hints, and answers [GGPM05; CCJ+04]. Additionally, these environments employ user models for configuring and tracking, e.g., users’ interactions with the system, their presentational preferences, their previous knowledge, and the exercises they have mastered [MAB+01; Ullo8; CCV10].

**Science:** The PhysML language extends OMDoc towards physics. To the functional and logical structures of mathematical knowledge, which OMDoc supports natively, it adds the principal concepts of observables, systems (apparata for carrying out experiments), and experiments (a description of a test setup that allows for taking measurements) [HKSo6]<sup>11</sup>.

**Engineering:** In a case study on software engineering, covering documents such as contracts, requirements specifications, or manuals, we have, in addition to the mathematical model and versioning and certification metadata (cf. [section 2.1.7.6](#)), modeled the software process (with relations such as “refines”, “implements”, or “describes use of” among sections of documents) and the organization structure (e.g. which people are responsible for which documents or what subproject) as metadata [KKL10a; KKL10b].

---

<sup>11</sup>This is actually treated as primary knowledge; cf. the discussion in [section 2.1.1.3](#).

### 2.1.8 Discussions in Mathematical Collaboration

*Who would have guessed  
that the working record of a mathematical project  
would read like a thriller?*

—TIMOTHY GOWERS and MICHAEL NIELSEN [GN09]

Previous research has investigated two kinds of scientific discourse: One kind is embedded into scientific publications, which, e.g., make claims and argue about claims made in other, cited publications. This has often been studied in combination with rhetorical structures; see [GHC+09] for an overview. This thesis focuses on the other kind of scientific discourse, which is treated as secondary knowledge in the sense of [section 2.1.1.3](#) and held externally of the representations of its subjects, e.g. in discussion forums.

[Section 2.1.8.1](#) briefly reviews unstructured commenting and rating facilities, which are ubiquitous on the Web 2.0. In the context of collaborative problem solving (but not yet in a mathematical context), argumentation models have been designed to capture structured discussions (cf. [section 2.1.8.2](#)).

#### 2.1.8.1 Discussions and Ratings in Web 2.0 Environments for Mathematics

Most web 2.0 environments for MKM support commenting and rating, but neither in a semantic nor in a mathematics-specific way. When the semantics of comments and ratings is opaque to services, they can neither guide users in the flow of a discussion, nor can they assist with finding solutions for the problems discussed. For example, the MathOverflow site [Mate], initially mentioned in [section 1.4.2](#) and reviewed in [section 6.6.4](#), merely supports posting questions and answers, and rating answers. Other environments support more exactly classified comments and ratings, but, again, not yet to an extent that would allow for semantic assistance.

panta rhei [Pan; Mül10a], a research prototype of a browser for mathematical lecture notes, supports general types of comments, such as “advice”, “answer”, “comment”, “example”, and “question”). It allows for retrieving comments by type, but, as comments can be mixed arbitrarily – one could, e.g., post an “answer” where no question has been posted before –, it does not guide the flow of discussions. panta rhei offers three general dimensions of ratings (in the case of lecture notes: relevance, soundness, and quality of presentation). The commenting and rating facilities of the related Connexions and PlanetMath sites are briefly reviewed in [sections 9.5.2](#) and [9.5.4](#), respectively. [Section 6.3.2](#) discusses the utilization of such ratings for the purpose of validation.

#### 2.1.8.2 Argumentation Models for Discussing [Wicked] Problems

Structured discussions are commonly modeled using *argumentation models*, most of which have been inspired by IBIS (Issue-based Information System [KR70]). IBIS models the design process for complex problems – i.e. a generalization of our setting – as “a conversation among stakeholders (e.g. designers, customers, implementors, etc.), in which they bring their respective expertise and viewpoints to the resolution of design issues” ([KR70], as cited by [CB87]). In a collaborative environment, users report issues and argue about them, propose solutions that are again subject to discussion, until finally a solution is approved and implemented. Such an exchange of arguments

can be lengthy and hard to keep focused, as issues can be “*wicked problems*” exposing traits like not allowing for a “*definitive formulation*”, having solutions that are “*not true-or-false but good-or-bad*”, and the nonexistence of an “*immediate and [...] ultimate test of a solution*” [RW73].

In applications of IBIS to knowledge engineering, a solution is usually materialized in an improved version of the affected knowledge item or a new knowledge item. Later, other users, who want to understand why some knowledge item has been modeled in a particular way, can trace back the discussion that led to its creation or modification. Thus, the discussions about issues with knowledge items become part of the collective experience of the community. The gIBIS hypertext system [CB87] applied the IBIS method to system design, which served as an inspiration for subsequent applications in ontology engineering, a subfield of knowledge engineering. As a collection of semantically structured mathematical knowledge can be considered an ontology, particularly if it contains formal definitions of mathematical concepts,<sup>12</sup> this thesis considers IBIS derivatives from ontology engineering for modeling discussions about mathematical knowledge. These models, including the DILIGENT argumentation model and the Protégé Change and Annotation Ontology (ChaO), vary the original IBIS model towards a more precise structure. This is possible because ontology engineering lends itself better to formal modeling than governmental planning, the original application domain in which wicked problems and IBIS have been investigated [RW73].

The DILIGENT argumentation model has been conceived in the context of the namesake collaborative ontology engineering methodology with the design goal of making arguments more focused than in plain IBIS in order to make design decisions more traceable and allowing for inconsistent argumentations to be detected [TPS+05; TSL+07]. An argumentative thread in the DILIGENT model is structured as follows: When an issue has been raised – e.g. by verbalizing a requirement for the ontology to be designed –, collaborators can express their agreement or disagreement with it, i.e. whether they consider this issue important, justified, and legitimate. An issue is resolved by implementing a proposed and – again by posting agreements – approved idea for solving it by conceptualizing and formalizing a knowledge item (called “ontology entity” in DILIGENT) and concluding the discussion thread with an explanation of the decision taken. This decision will link to the issue that has been solved and to the idea *i* that was realized. If that idea was to create or modify a knowledge item *k*, a link “*i* resolves into *k*” will be created. Besides merely agreeing or disagreeing with an issue or idea, collaborators can also *argue* about it, i.e. justify it by examples or evaluations, or challenge it by alternative proposals or counter-examples, and others can again agree or disagree with these arguments.

ChaO is used in Collaborative Protégé, an extension of the Protégé ontology editor [TNT+08], and its web frontend WebProtégé [TVNo8]. The “annotation” subset of ChaO roughly correspond to the main concepts of DILIGENT, but the ChaO concepts are more loosely coupled. In line with Protégé’s independence from a particular ontology engineering methodology, and similarly to the panta rhei system introduced above, the ChaO model does not prescribe a certain flow of discussion but allows annotations of any type to annotate (i.e. reply to) other annotations.<sup>13</sup> In

---

<sup>12</sup>Chapter 4 gives a detailed account of this correspondence.

<sup>13</sup>There have been considerations to elaborate ChaO into a generic model for collaborative ontology engineering workflows, of which the DILIGENT methodology would then only be a special case [SFNT+08].

contrast to DILIGENT applications, Collaborative Protégé has, however, a large community of users applying it in realistic settings, particularly within biomedical informatics.<sup>14</sup>

## 2.2 Requirements for Reusably Representing and Exchanging Mathematical Knowledge

One goal of this thesis is to develop an improved language for representing mathematical knowledge in a way that enables its reuse across knowledge bases, information retrieval adequate to the structures of the knowledge, and integration with mathematical services, such as automated reasoning and computation, without compromising comprehensibility for human end-users. Having reviewed the structures of mathematical knowledge in all of their complexity and diversity in [section 2.1](#), we are now ready to specify the requirements for this language more precisely (cf. [figure 2.4](#)).<sup>15</sup>

[Section 2.4](#) reviews state-of-the-art languages for representing mathematical knowledge. Before that, [section 2.3](#) introduces the [semantic] web foundations on which almost all of these languages are based. [Tables 2.3 and 2.4 in section 2.5.1](#) summarize how well the languages reviewed satisfy the requirements established above, leading to an informed decision on which of them to choose as a basis for our improvements.

## 2.3 Knowledge Representation on the [Semantic] Web (State of the Art)

Most representation languages for semiformal mathematical knowledge that are discussed in this thesis – both the state-of-the-art representations and the new one that I have developed – are based on XML, RDF, and ontologies. This section briefly reviews those aspects of those foundational technologies of the Semantic Web that are relevant for this thesis, while leaving considerations specific to the mathematical domain to [section 2.4](#). This section focuses on URIs/IRIs, XML, RDF, and the basics of queries and ontologies. Further layers of the semantic web architecture<sup>16</sup> are touched upon in later chapters: proofs briefly in [Section 6.3](#), trust – built by making provenance information explicit – briefly in [section 6.4.1.1](#), and user interfaces and applications extensively in [part III](#).

### 2.3.1 URIs, IRIs, and the Linked Data Principles

Documents we want to retrieve, and objects we want to reason about, need to be identified. URIs (uniform resource identifiers [[BLFM05](#)]) allow for identifying things in a web-scalable way. In principle, URIs can identify anything, from information resources, such as fragments of documents retrievable on the Web, to non-information resources, such as “the Kepler conjecture”. URLs (uniform resource locators) are a special case of URIs; they describe a location where an

---

<sup>14</sup>personal communication with ALEXANDER GARCÍA CASTRO, 2009-07-29

<sup>15</sup>Here and in subsequent requirements or specification texts in this thesis, capitalized keywords are used in accordance with RFC 2119 [[Bra97](#)].

<sup>16</sup>see [[GMB08](#)] for an overview

- S:** All of the previously reviewed structures of mathematical knowledge SHOULD be supported; where this is impossible, missing dimensions MUST be compensated for by language extensions along the criteria L.E and L.→ below. We subdivide this criterion as follows:
  - S.L.{O,S,T}:** logical/functional structures: mathematical objects, statements, theories
  - S.{R,N,M,D}:** rigorous language or rhetorical structures, notation, metadata, discussions
- F:** Mathematical knowledge occurs in different degrees of formality; applications targeting human users and automated agents require both informal and formal representations. Therefore,
  - F.R:** the language SHOULD be able to represent knowledge in a wide range from informally to fully formalized, and
  - F.C:** many degrees of formality SHOULD be able to coexist in one document, interlinked with each other.
- L:** In real-world applications, mathematical knowledge is combined with multiple dimensions of non-mathematical knowledge. Therefore, our language SHOULD support interlinking of these dimensions by rich annotation facilities, but also give authors the freedom to represent some knowledge by external means and link it to representations in our language. In detail,
  - L.A:** the language MUST allow for attaching non-mathematical metadata and annotations to mathematical knowledge items, regardless of their granularity, and
  - L.→,** i.e. “L out[going]”: for linking mathematical knowledge items to external mathematical or non-mathematical resources, and
  - L.←,** i.e. read “L in[coming]”: it MUST be possible to address all mathematical knowledge items expressed in the language from outside, in order to link external representations to them, for example standoff markup (cf. the discussion in [section 2.1.1.3](#)) or existing representations in different languages.
- C:** Knowledge represented in our language SHOULD be comprehensible
  - C.S:** to arbitrary external services – therefore, the knowledge SHOULD be self-describing in a machine-comprehensible way. A formal semantics governing the operation of services is REQUIRED for that reason.
  - C.H:** to human users – therefore, published human-comprehensible documents generated from representations in our language SHOULD retain semantic annotations, so that assistive services can retrace the original knowledge and make it available to the user on request, e.g. integrated into a user interface.
- E:** Users of existing languages and authoring tools SHOULD NOT be forced to migrate. Instead, the new language SHOULD be designed in such a way that it can be embedded into existing representations or connected to them.

Figure 2.4: Requirements for Reusably Representing and Exchanging Mathematical Knowledge

information resource can be retrieved – in a Web context usually by an HTTP client. The linked data best practices<sup>17</sup>, however, advise always using HTTP URLs as URIs to ensure retrievability:

1. Use URIs to identify things.
2. Use HTTP URIs so that these things can be referred to and looked up (“dereferenced”) by people and user agents.<sup>18</sup>
3. Provide useful<sup>19</sup> information about the thing when its URI is dereferenced, using standard formats such as RDF (Resource Description Framework).
4. Include links to other, related URIs in the exposed data to improve discovery of other related information on the Web.

**Principle 3** for non-information resources – i.e. real-world things, which are not themselves retrievable from the Web, – is satisfied by way of an associated information resource – or multiple ones for multiple representation formats –, to which the client is redirected [SCo8]. Machine-comprehensible information is usually provided as RDF (see below), whereas human-comprehensible information is provided as HTML.

IRIs (internationalized resource identifiers [DSo5]) are a generalization of URIs that directly supports Unicode. Unicode is a set of characters and encodings for them, covering most human writing systems of all cultures, including mathematical symbols [Uni]. For backwards compatibility, a translation from IRIs to URIs has been specified, which escapes characters outside of the set allowed for URIs. Therefore, I continue to use the more familiar term “URI” in the following, and only use “IRI” where Unicode is relevant.

### 2.3.2 XML

Most state-of-the-art representation languages for mathematical knowledge are based on XML (eXtensible Markup Language [BPSM+o8]). XML documents have a tree data model – the infoset – and a linear text syntax. A parser transforms the latter into an infoset, whereas a serializer does the opposite. This thesis simply speaks of “XML documents” when it is clear from the context whether that means the infoset of a document or its serialization.

Within the XML family, it is easy to implement a new markup language, as one can draw on extensive tool support for processing, presenting, authoring, validating, querying, etc. Anything said about using XML in this section also applies to the particular XML-based languages<sup>20</sup> reviewed in [section 2.4](#); these reviews only add further remarks on tool support where more specialized solutions exist.

---

<sup>17</sup>See [BLo6a]; here cited as paraphrased by Wikipedia [WikioB]

<sup>18</sup>I.e., the URI is treated as a URL (uniform resource locator).

<sup>19</sup>This usually means: machine-comprehensible.

<sup>20</sup>From now on, I will use the shorter term “XML language”.



### 2.3.2.1 Semantics – The XML Infoset

The abstract data model of an XML document, the XML Information Set (infoset [CTo4]), is an ordered tree with labeled nodes (elements, attributes, or text nodes). Actually, this tree only forms the backbone of the data model, as its nodes can be given *fragment identifiers* and then referenced from other nodes, even across documents. This is not part of the foundational infoset specification; more high-level specifications, such as *xml:id* [MVW05] or XPointer [GMM+03] add this aspect, which is highly important for semantic markup. When an XML document that is retrievable from the URL <http://example.org/document.xml> contains a fragment locally identified as *toc*, the URL of the fragment is <http://example.org/document.xml#toc>.

XML allows for mixing different vocabularies in one document. Vocabularies are distinguished by their *namespace URIs*; the names of elements and attributes are pairs of a namespace URI and a local name within that namespace [BHL+09]. The namespace URI is usually a mere identifier. Some vocabulary developers make a human-comprehensible HTML page with information about the respective XML language available from that URI. RDDL (Resource Directory Description Language [Rdd]) has been suggested for providing a machine-comprehensible description of an XML language from its namespace URI in a linked data fashion, including pointers to related resources (schemata for validation, stylesheets for presentation, etc.), but hardly been adopted.

The semantics of a particular XML language – beyond the general infoset semantics – is rarely specified in a formal, model-theoretic way, but usually in a human-readable specification manual; [section 2.4](#) mentions some exceptions. Modern XML schema languages, including XSD and RELAX NG (see below), allow for annotating elements and attributes of an XML vocabulary with human- and machine-readable information [Ogb05]. However, a semantics for the content of such annotations has not been specified; therefore, machines cannot process them in a generic way. In practice, this feature is rarely used to formalize the semantics of an XML vocabulary.

### 2.3.2.2 Syntax – Well-Formed XML Documents and XML Schemata

The concrete representation (*serialization*) of an XML document has to respect certain fundamental syntactic rules that make it *well-formed*. This thesis assumes the XML syntax [BPSM+08], including XML namespaces [BHL+09], as known. Vocabulary-specific aspects of the syntax of an XML language are usually specified by a formal grammar, an XML schema<sup>21</sup>. Common XML schema languages are DTD (Document Type Definition [BPSM+08]), XSD (XML Schema Definition Language, formerly known as “XML Schema” [GSMT09]) and RELAX NG (Regular Language for XML, Next Generation [CM01]).

### 2.3.2.3 Processing, Querying, and Presenting XML

An XML parser translates a serialization of a (well-formed) document to its infoset representation, on which algorithms for processing XML are usually defined. XSLT (Extensible Stylesheet Language Transformations [Kay07]) is a high-level language for defining translations from XML to XML or to text. The related XQuery language has a very similar functionality but focuses on

---

<sup>21</sup>In this thesis, the lowercase term “schema” denotes a formal grammar of an XML language in general, regardless of the actual schema language used.



querying [BCF+07]. XSLT and XQuery have XPath in common, a (sub)language for selecting nodes from XML documents [BBC+07]. Throughout this thesis, I use XPath syntax as a shorthand for XML structures; for example, *a*[@href='http://kwarc.info'] denotes an *a* element whose *href* attribute (prefixed with an @ to distinguish it from an element) has the given value.

The *presentation* of an XML language can be defined by CSS (Cascading Style Sheets [W3Ca]). They focus on style and layout but do not support complex selections or restructurings. Therefore, complex XML documents can better be handled by *transforming* them to an XML language with a defined visual layout model, such as HTML (Hypertext Markup Language [Theo2]<sup>22</sup>) for hypertext documents or Presentation MathML for mathematical objects (cf. section 2.4.2). The transformation of a semantic source document to a presentation-oriented target document is usually implemented in XSLT. HTML again uses CSS for styling.

### 2.3.3 RDF

The Resource Description Framework (RDF [W3Cb]) has originally been designed as a data model for metadata and has then become the standard for knowledge representation on the Semantic Web.

#### 2.3.3.1 RDF Semantics

RDF's data model is the one of labeled, directed multigraphs (cf. [Hay04]) – as opposed to the tree-oriented data model of XML. RDF graphs are commonly thought of as being decomposed into “subject–predicate–object” *triples*, where subject and objects are two nodes and the predicate is the label of the edge connecting them. The subject is always a *resource*, identified globally by a URI, or locally to the current graph by a “blank node ID”, which is similar to an existentially quantified variable. The predicate is always identified by a URI. The object can be another resource, or a “literal” – an atomic value, for which an optional URI-identified datatype (e.g. string or integer) can be declared, and which can be tagged with a [natural] language identifier, such as “de” for German.

While one can merely define a *grammar* for an XML language relying on existing standards, one can specify a formal, model-theoretic semantics of an RDF vocabulary in an *ontology language* (see section 2.3.4 for details). The [edge] labels in an RDF graph correspond to *properties* (= relations, roles) in an ontology. The *type* property from RDF's built-in vocabulary is a special one: The object of such a triple is treated as a *class* (= type, concept) of an ontology. As with XML, an RDF vocabulary has a namespace URI. RDF differs from XML in that not only the namespace, but every vocabulary term has a URI. Such URIs are simply formed by concatenating the namespace URI and the local name.

Figure 2.5 shows an RDF graph, which describes myself: my name, my date of birth, and my workplace homepage address. The FOAF ontology (cf. section 3.5) and the BIO vocabulary are used. The person and his homepage are identified by URIs (abbreviated as explained below in section 2.3.3.2); his name is a literal. There is no direct way of attaching a birth date to a person;

<sup>22</sup>HTML, historically based on the XML predecessor SGML, has been reformulated as an XML language called XHTML. This thesis uses the terms HTML and XHTML synonymously and always assumes XML conformance.

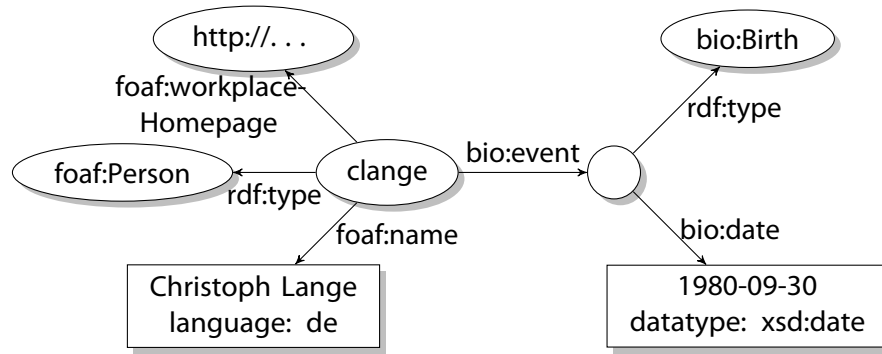


Figure 2.5: An Example RDF Graph

therefore the birth is modeled as an event in the life of the person. As there is no need for talking about this event outside the current graph, it is modeled as a blank node.

### 2.3.3.2 RDF Syntax (Serializations)

Multiple concrete encodings (“serializations”) of the RDF data model exist. The RDF/XML encoding [Beco4b] is most widely supported by software tools and libraries. Many humans prefer the Turtle text serialization [BBL08], as it is easier to read and to write manually. As most RDF graphs in this thesis are given in Turtle, [listing 2.1](#) shows the RDF graph from [figure 2.5](#) as an example. N-Triples [GB04], a restricted subset of Turtle – one triple per line, no URI abbreviations (see below) –, has been invented for expressing RDF test cases but is also used for low-level data exchange operations, such as importing RDF into a database. (Turtle and N-Triples have actually been defined as subsets of the N3 language, which has a richer syntax and a built-in vocabulary for first-order rules; see [section 2.4.10.1](#) for an example.)

Easier machine processability has also frequently been requested for XML-based serializations of RDF. RXR (Regular XML RDF [Beco4a]), one of the alternatives that have been suggested, is used for some implementations presented in this thesis. Additionally, there is the RDFa serialization for embedding RDF annotations into X[HT]ML documents. Due to its importance for this thesis, [section 2.3.3.4](#) covers RDFa in more detail.

All of the RDF serializations mentioned so far, except N-Triples and RXR, use namespace prefix bindings in order to define abbreviations for URIs. For example, with the prefix *rdf* bound to <http://www.w3.org/1999/02/22-rdf-syntax-ns#>, the abbreviation *rdf:type* would expand to <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>. In the context of RDFa, this syntax has been specified under the name CURIE (Compact URI [ABM+11, section 6]).<sup>23</sup> In the remainder of this thesis, namespace prefix  $\mapsto$  URI bindings are mostly omitted for readability; [table A.1](#) lists common namespace prefixes and URIs.

<sup>23</sup>RDF/XML uses the more restricted XML namespace prefix bindings, which do not allow, e.g., leading digits in the local name.

Listing 2.1: RDF example in Turtle

```

@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix bio: <http://purl.org/vocab/bio/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://purl.org/net/clange>
  a foaf:Person ;
  foaf:name "Christoph Lange"@de ;
  bio:event [
    a bio:Birth ;
    bio:date "1980-09-30"^^xsd:date ] ;
  foaf:workplaceHomepage <http://kwarc.info/clange/> .

```

### 2.3.3.3 Differences Between the XML and RDF Data Models

The fundamental difference between the data models of XML and RDF – ordered tree versus directed multigraphs – has implications on the suitability of either data model for representing a particular kind of knowledge. This section briefly mentions those implications that are independent from *mathematical* knowledge representation but will nevertheless have to be reconsidered when deciding how to represent mathematical knowledge.

XML is naturally suited for representing ordered structures. There are also several approaches to ordering nodes of an RDF graph, but all of them require the introduction of artificial resources representing ordered data structures – ordered sets (“sequence[s]”) or linked lists (“collections”) – and do not impose well-formedness constraints by default. Furthermore, all knowledge to be represented in RDF has to be broken down into triples. While there are standardized ways of, e.g., representing *n*-ary relations in RDF [NR06], they are cumbersome to read and write for humans when authoring RDF manually, they require additional software support for processing, and they do not go well along with RDF-based reasoning<sup>24</sup> and querying<sup>25</sup>.

Conversely, the unordered graph nature of RDF enables trivial enhancements of a knowledge base by merging two graphs. For example, the graph shown in [figure 2.5](#) and the subsequent listings could be considered as a merger of my personal profile – expressed using the FOAF vocabulary – and my biography – expressed using the BIO vocabulary. There is no comparable canonical way of merging XML trees, be they expressed in the same or in different XML languages. Many XML languages, however, provide well-defined extension points for adding fragments of another XML language that uses a different namespace.<sup>26</sup>

Listing 2.2: RDF example in XHTML+RDFa

```
<?xml version="1.0"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      prefix="foaf: http://xmlns.com/foaf/0.1/
            bio: http://purl.org/vocab/bio/0.1/
            xsd: http://www.w3.org/2001/XMLSchema#">
  <head>
    <title>Christoph Lange</title>
  </head>
  <body about="http://purl.org/net/clange" typeof="foaf:Person">
    <span property="foaf:name" xml:lang="de">Christoph Lange</span>
    (<a rel="foaf:workplaceHomepage" href="http://kwarc.info/clange/">homepage</a>)
    <div rel="bio:event">
      <h2>Biography</h2>
      <ul>
        <li typeof="bio:Birth">born on
          <span property="bio:date" content="1980-09-30"
                datatype="xsd:date">September 30, 1980</span></li>
      </ul>
    </div>
  </body>
</html>
```

#### 2.3.3.4 RDFa – Embedding RDF into X[HT]ML

The RDFa serialization combines XML and RDF in that it allows for embedding RDF graphs into X[HT]ML documents [ABM+08]. Its syntax consists of a set of additional attributes (listed in table 2.2) that can be attached to almost any element of the host language, whereas existing content of the XML document – usually text – can also be reused as RDF literals (cf. listing 2.2). RDFa offers a number of alternative ways of modeling the same RDF triple, thereby keeping redundancy and the disruption of the original XHTML structure by the introduction of dummy elements (e.g. empty *spans*) that carry annotations as low as possible. While its original specification assumes XHTML as a host language, only very few details of the specification actually depend on XHTML. The upcoming RDFa 1.1, which the work presented in this thesis builds on separates the latter details from the generic core (cf. [ABM+11]). Therefore, RDFa is now also being adopted for other XML languages; the RDFa wiki provides an overview [IL10].

Microformats and microdata are alternative syntaxes for embedding metadata into HTML, which the implementations presented in this thesis do not use. Besides the obvious justification that

---

<sup>24</sup>When representing ordered structures in an OWL ontology (cf. section 2.3.4), one has to avoid RDF collections, as the RDF encoding of OWL uses them internally for representing *n*-ary DL expressions. Instead, one has to introduce new properties for creating linked list structures [DRS+06].

<sup>25</sup>At least support for querying RDF collections, which some query processors already support by non-standard extensions, will be standardized in the upcoming version 1.1 of the SPARQL query language [HS10a].

<sup>26</sup>Adding a MathML formula, e.g., to a DocBook document is such a case; section 2.4.8.6 discusses more examples.

Table 2.2: The RDFa 1.1 attributes, quoted from [ABM+11]

Attribute	Specification
<i>@about</i>	a [...] <sup>a</sup> CURIE or URI, used for stating what the data is about (a ‘subject’, in RDF terminology)
<i>@content</i>	a CDATA string, for supplying machine-readable content for a literal (a ‘plain literal object’, in RDF terminology)
<i>@datatype</i>	a term <sup>b</sup> or CURIE or absolute URI representing a datatype, to express the datatype of a literal
<i>@href</i> <sup>O</sup>	a URI for expressing the partner resource of a relationship (a ‘resource object’, in RDF terminology)
<i>@prefix</i> <sup>b</sup>	a white space separated list of prefix-name URI pairs of the form NCName ‘:’ ‘ ’+ xs:anyURI
<i>@profile</i> <sup>b</sup>	a white space separated list of one or more URIs that reference external definitions of terms and/or prefix mappings. [...]
<i>@property</i>	a white space separated list of terms <sup>b</sup> or CURIEs or absolute URIs, used for expressing relationships between a subject and some literal text (also a ‘predicate’)
<i>@rel</i>	a white space separated list of terms <sup>b</sup> or CURIEs or absolute URIs, used for expressing relationships between two resources (‘predicates’ in RDF terminology)
<i>@resource</i>	a [...] CURIE or URI for expressing the partner resource of a relationship [...] (also an ‘object’)
<i>@rev</i>	a white space separated list of terms <sup>b</sup> or CURIEs or absolute URIs, used for expressing reverse relationships between two resources (also ‘predicates’)
<i>@src</i> <sup>O</sup>	a URI for expressing the partner resource of a relationship when the resource is embedded (also a ‘resource object’)
<i>@typeof</i>	a white space separated list of terms <sup>b</sup> or CURIEs or absolute URIs that indicate the RDF type(s) to associate with a subject
<i>@vocab</i> <sup>b</sup>	a URI that defines the mapping to use when a term is referenced in an attribute value
<i>@xmlns:prefix</i> <sup>O</sup>	a method of declaring prefix mappings as defined in [BHL+09]. Prefix mappings declared via this attribute are equivalent to those declared using <i>@prefix</i> . <sup>c</sup> If this attribute and <i>@prefix</i> declare a mapping for the same prefix on the same element, the mapping from <i>@prefix</i> MUST take precedence. Document authors SHOULD use <i>@prefix</i> , and SHOULD NOT mix <i>@prefix</i> and this attribute on the same element.

<sup>a</sup> This table omits references to “safe CURIEs”; RDFa 1.1 merely has them for RDFa 1.0 backwards compatibility.

<sup>b</sup> RDFa *profiles* can provide various ways of writing shorter [C]URI[E]s; a host language may define a default profile. In detail, a profile can map *prefixes* to namespace URIs, including the default prefix for usage with CURIEs of the form *:localname*. For an even shorter notation, the profile can define *terms* (of datatype *NCName*, i.e. XML names without colons). Terms can either be mapped to URIs individually, or the profile can define a default *vocabulary*, i.e. a namespace that holds the terms.

<sup>c</sup> Note that the default XML namespace does not have any influence on CURIEs.

<sup>O</sup> optional attribute

they have only been specified for the HTML host language, this is justified by RDFa conceptually subsuming them:

**Microformats** are small, single-purpose vocabularies for annotating HTML – mostly by (ab)using the *@class* attribute actually intended for usage with CSS. That leads to a much lower expressivity than RDFa has but makes them easy to read and write for HTML authors. Microformats have been developed, e.g., for people and organizations (hCard), social relationships (XFN = XHTML Friends Network), calendars and events (hCalendar), and licenses (rel-license) [Mic; TLo9]. Microformats lack namespaced identifiers and thus reusability and scalability. Their specifications are human- but not machine-comprehensible; however, unofficial translations to RDF exist, hard-coded per microformat [Esw].

**Microdata** is an annotation syntax proposed for HTML 5<sup>27</sup> [Hic10; TLo9]. Its set of attributes is syntactically different from RDFa but semantically covers a large subset of it. The main syntactic difference is the lack of namespace prefix bindings for abbreviating URIs, justified by usability benefits for authors [Ten09a]. Semantically, microdata retain most of the expressivity of RDF, the only exceptions currently being datatypes and XML literals [Ten09b].

### 2.3.3.5 Querying RDF

RDF graphs can be queried using the SQL-like SPARQL language; an interface that accepts SPARQL queries is called *SPARQL endpoint* (cf. [PS08] and section 6.5.2.2). As a consequence of the differences between the XML and RDF models explained in section 2.3.3.3, it may be necessary to represent knowledge using a combination of both models. Querying such combinations is not currently well supported. One possible combination is using literals of datatype *rdf:XMLLiteral*. The OpenLink Virtuoso RDF triple store (the common term for RDF databases) allows for filtering XML literals matched by a SPARQL graph pattern by XPath node tests (*xpath\_contains* predicate [Olv]). The Corese RDF engine can additionally reuse variables from the proper SPARQL part of a query in calls to its *xpath* extension function [CKKC+09]. None of these extensions has made it into the SPARQL standard yet.

RDFa enables another combination, which allows for focusing on those structures that can easily be represented in RDF, while leaving the representation of *n*-ary and ordered structures to the XML host language. The RDFa 1.1 API [SAR+11], which remains to be implemented by browsers, will at least give in-browser scripts similar means of accessing embedded RDF as the Document Object Model (DOM [W3C]) offers for XML. The XSPARQL [AKK+08] query language combines SPARQL and XQuery; however, such a query would still rely on a separate service that makes the RDFa annotations available as queryable RDF.

### 2.3.4 Ontologies

According to the most common definition, here cited in an extended version, “*an ontology is a formal, explicit specification of a shared conceptualization*” [SBF98]. The following definition makes this more precise:

---

<sup>27</sup>This thesis ignores the official spelling “HTML5” but instead consistently spells systems/languages as “name\_version”.

In this definition, “conceptualization” refers to an abstract model of some domain knowledge in the world that identifies that domain’s relevant concepts. “Shared” indicates that an ontology captures consensual knowledge, that is, it is accepted by a group. “Explicit” means that the type of concepts in an ontology and the constraints on these concepts are explicitly defined. Finally, “formal” means that the ontology should be machine understandable. (ISABEL F. CRUZ and HUIYONG XIAO [CX05])

*“An ontology typically consists of a hierarchical description of important concepts in a domain, along with descriptions of the properties of each concept.”* [Hor02] Applications of ontologies include “sharing knowledge bases, enabling communication among software agents [i.e. what this thesis calls ‘services’], integration of disparate data sets, [...], representation of semantics for services and complex software applications, helping provide knowledge-enhanced search, providing a conceptual framework for indexing content.” [GBO+08]

In the Semantic Web layer cake, ontologies assume the role of giving semantics to the vocabularies used in RDF graph – more or less formally, depending on the ontology language used. In terms of the definitions given by MICHAEL GRUNINGER et al. [GBO+08], this thesis deals with ontologies that are implemented in logical languages. In contrast to XML schema languages, which merely have a formal syntax, these languages also have a model-theoretic semantics. The ontologies presented in this thesis are semi-structured in that most terms in the vocabulary are defined by sentences in a logical language, whereas they “require extralogical conditions [...] to specify the intended interpretations of some [other] terms” [GBO+08].

The most common ontology language on the Semantic Web is the OWL Web Ontology Language [HRH+; HPSH03], which, in its current version 2, is based on the *SR**O**I**Q* description logic (DL), a decidable subset of first-order logic (FOL), or on more specific subsets thereof [MCGH+09], which also comprise the widely used RDF Vocabulary Description Language RDFS [BG04]. Other parts of this thesis particularly depend on two characteristic features of *SR**O**I**Q* and OWL/RDFS [MPSP09]:

**Open World Assumption:** Facing the distributed nature of the Web, *SR**O**I**Q* assumes an open world. That means that facts that have not explicitly been stated in the local scope of the reasoner, are not assumed to be false by default, i.e. they are assumed possible, unless they have explicitly been declared false or their falsehood follows from other facts. Additional information about some resource could easily be given on a remote host; consider the practical example of reviews of a book that are not hosted on the publisher’s homepage. The publisher would be wrong to assume that the information on their site is complete.

**No Unique Name Assumption:** When two things have different URIs, OWL does not consider them different by default. This is, once more, due to the distributed architecture of the Web and the possibility of redirecting URLs.

Two other technical terms from DL used throughout this thesis are ABox and TBox: ABox (from “assertion”) refers to statements about instances, whereas TBox (from “terminology”) refers to axioms about classes and properties.

Further ontology specification languages have arisen out of the ongoing trade-off between expressivity and computational tractability being played out in ontology research and development.



One example of an application area that requires more expressivity than that of DL is given by the emerging standards for semantic web services: the WSMO (Web Service Modeling Ontology [Wsm]) relies on F-Logic [KLW95], a first-order variant, rather than OWL. As expressive ontology languages, however, do not always meet the requirements that particular applications pose on tractability, complex domain knowledge has also been expressed in more restricted languages by employing simplifications or workarounds. This is, e.g., the case with the DOLCE upper ontology: It was originally modeled in FOL and implemented in KIF [GF+92], but a simplified “Lite” implementation in OWL was provided for semantic web services [MBG+03].

Nowadays, ontologies are as crucial to business success as software. Like in software engineering, the issues of ontology development by distributed groups, over longer lifecycles, and with intended practical deployment with measurable costs has stressed the importance of methodology. Ontology engineering ontologies, such as Methontology [FLGP97] or DILIGENT (cf. [section 2.1.8.2](#)) cover the development of an ontology from scratch: specifying requirements, conceptualizing the domain of interest, and then formalizing it in a logic or ontology language, reusing existing ontologies as appropriate. HELENA SOFIA PINTO and JOÃO P. MARTINS provide a general overview of this process and review some of the older methodologies [PM04].

### 2.4 Representing Semiformal Mathematical Knowledge (State of the Art)

*The nice thing about standards is  
that there are so many of them to choose from.  
Furthermore, if you do not like any of them,  
you can just wait for next year's model.*

—ANDREW S. TANENBAUM [Tano3]

This section reviews the state of the art in languages for representing semiformal mathematical knowledge.<sup>28</sup> The reviews focus on the coverage, the syntax, and the semantics of the respective languages, but also on existing translations from and to other languages. [Section 2.5](#) then assesses all languages reviewed against the requirements established in [section 2.2](#) and outline the development of a representation and exchange language that improves over the state of the art.

#### 2.4.1 Semantic vs. Content Markup

The languages covered in this section are markup languages in that they annotate plain text. They do not, at least not primarily, say how that text should be presented (e.g. as  $1 + 2 + 3$ ); therefore, they are not presentation markup languages. They intend to say what the text means (e.g. “the sum of the natural numbers 1, 2, and 3”). The way they do it satisfies TIM BRAY’s definition of semantic markup [Brao3]:

- “[A] human understands [the markup] in context and may reasonably consider it as a basis for action.”

---

<sup>28</sup>A more comprehensive review of the early history of mathematical knowledge representation, particularly the early history of OpenMath, can be found in [Stro3]



- “There is an expectation that there is software which when applied to the markup will produce a useful result.”

This definition is not sufficiently precise for mathematical computation. Therefore, in MKM, one traditionally distinguishes between *semantic markup* and *content markup*:

Formulae at the *semantic level* are those which the application has the deepest understanding of and on which it can better perform *computations*. In the fields of Computer Algebra Systems and theorem provers, examples of such computations include evaluation, simplification, automatic (dis-)proving, and type-checking. This level is intrinsically application-specific.

In between [the semantic level and the presentation level] is an intermediate level, which we call *content level*, whose aim is to encode the structure and, to a limited extent, the semantics of mathematical formulae. MathML Content and OpenMath are examples of markup languages that encode formulae at this level. The content level is the most effective vehicle of interoperability across MKM applications not sharing semantic foundations. ([PZo6])

The above-mentioned representation of a sum is not yet sufficient for computation, as it does not refer to a particular axiomatic definition of natural numbers and addition, on which a mathematical software system would be based. Imagine a system with a unary representation of natural numbers and a left-associative definition of the  $n$ -ary addition operator. In such a system, our  $1 + 2 + 3$  example would natively be expressed as  $\text{add}(\text{add}(s(o), s(s(o))), s(s(s(o))))$ . A system with a binary representation of natural numbers and right-associative addition would obviously require a different representation.

Content MathML and OpenMath, at least with their standard vocabulary, do not take these differences into account; hence, they are usually called content markup languages. More expressive content markup languages, such as OMDoc, however, push the limits quoted above in that they do allow for precisely defining the semantic foundations under which a mathematical object is to be interpreted. Therefore, and because of its semantic web context<sup>29</sup>, this thesis mostly speaks of “semantic markup”, and reserves the term “content markup” for MathML and OpenMath objects.

### 2.4.2 MathML

MathML (Mathematical Markup Language [ABC+10]) is an XML language that was originally conceived for embedding mathematical objects into web pages written in HTML. It features a presentation-oriented sublanguage (Presentation MathML) but also a content-oriented one (Content MathML).<sup>30</sup> For each structure of Content MathML, the MathML specification suggests – rather than prescribes – a “sample presentation” in Presentation MathML. XSLT stylesheets that perform this translation are provided. MathML is limited to representing mathematical

<sup>29</sup>Even though the semantics of the “Semantic Web” is often not strong enough for computation, there is no notion of a “Content Web”.

<sup>30</sup>These two sublanguages are also sometimes called “MathML Presentation” (short: “MathML-P”) and “MathML Content” (short: “MathML-c”).

objects but is often integrated into languages that can represent other aspects of mathematical knowledge, such as OMDoc (cf. [section 2.4.4](#)) or other languages for authoring books and manuals (cf. [section 2.4.8](#)).

Content MathML represents mathematical objects by their functional structure, similar to an abstract syntax tree. Important building blocks of mathematical objects are numbers, variables, symbols, and applications of mathematical objects to other mathematical objects. Content MathML comes with a default supply of symbols that cover high school and introductory university education. Additional symbols can be defined externally in *content dictionaries* (CDs) and then referenced from Content MathML expressions. MathML does not offer a sublanguage for writing CDs but delegates that task to other languages, such as OpenMath (cf. [section 2.4.3](#)). Presentation MathML mainly cares about the appearance of mathematical objects, but also makes some very basic semantic structure mandatory and allows for more. Identifiers, operators, and numbers have to be distinguished from each other; consider, for example,  $\langle \text{mi} \rangle x \langle \text{mi} \rangle$ ,  $\langle \text{mo} \rangle \subseteq \langle \text{mo} \rangle$ , and  $\langle \text{mn} \rangle 42 \langle \text{mn} \rangle$ . Furthermore, it is encouraged to make certain structures explicit even though they are invisible, e.g. the invisible multiplication operator<sup>31</sup>, or to group subterms into *mrow* elements.

Listing 2.3: General Structure of Annotated MathML

```
<semantics>
  <!-- the expression -->
  <annotation-xml encoding="...">
    <!-- the annotation -->
  </annotation-xml>
</semantics>
```

MathML allows for mixing presentation and content markup in various ways (“parallel markup” [[ABC+10](#), chapter 5]): The most common case in this thesis, as detailed below, is the annotation of presentational objects with content markup that makes their semantics explicit. Conversely, content markup can be annotated with presentation markup that fixes a rendering. For example, a content-markup variable with an identifier  $x_1$ , which is suitable for automated processing but does not look nice, can be annotated to render as a subscripted  $x_1$ . The general structure of an annotated expression is given in [listing 2.3](#). There can be fine-grained cross-links from parts of the expression to parts of the annotation. [Section 6.4.2.2](#) provides details on how to apply this technique; [figure 6.8](#) shows an example. This thesis mainly deals with XML annotations, but, using the *annotation* element, non-XML annotations can also be made – for example, images or code in the language of a CAS.

From version 3, Content MathML defines a sublanguage called “strict Content MathML”. It is semantically equivalent to OpenMath (see below<sup>32</sup>), which allows for giving it a model-theoretic semantics [[KR09](#)]. The non-strict syntax of Content MathML is backwards compatible to MathML 2. It offers pragmatic shorthands for many common mathematical operators and functions and is therefore more convenient to read and write for humans. For that reason, it has also sometimes been referred to as “pragmatic Content MathML” – a term that I continue to use in places where usability rather than formal semantics is of interest. The semantics of the non-strict

<sup>31</sup>It is best practice to use the special Unicode character *INVISIBLE TIMES* (U+2062). This character occupies some space on the screen, but it displays as whitespace.

<sup>32</sup>For historical reasons, this thesis prefers OpenMath syntax for content markup.

syntax is defined by mapping it to the strict one (see [ABC+10, chapter 4.6] for the specification and [listing 2.4](#) for an example).

### 2.4.3 OpenMath

OpenMath [Opeb] has been developed in the mid-1990s to facilitate data exchange between CAS by providing a uniform representation for the functional structure of formulæ, so-called *OpenMath objects* [BCC+04]. It has further been applied in areas as diverse as e-learning, scientific publishing, and interactive geometry (see, e.g., [Act; CCK+08; MLU+06; AEB07; Lur]<sup>33</sup>). OpenMath defines an abstract data model for mathematical objects and two concrete syntaxes for it, an efficiently processable binary one and a more commonly used XML one.

In addition to the XML syntax, this thesis uses an abbreviated variant of the abstract syntax to save space. The basic OpenMath XML elements are:

- *OMA* = application (abbreviated as @)
- *OMATP* = atttribute key/value pair (abbreviated as key  $\mapsto$  value)
- *OMATTR* = atttribution (container for *OMATP*; abbreviated as  $\alpha$ )
- *OMBIND* = binding (abbreviated as  $\beta$ )
- *OMF* = floating-point number
- *OMI* = integer
- *OMSTR* = string (abbreviated as "...")
- *OMS* = symbol (abbreviated as *cd#name* or *cdbase/cd#name*; the latter is the full URI of a symbol)
- *OMV* = variable (abbreviated by simply giving the variable name)

In recent years, OpenMath has been closely aligned with MathML [DK09]. Content MathML 3 and OpenMath 2 objects now share a common semantics but have different syntaxes (see, e.g., [listing 2.4](#)), which is owed to the different heritage.<sup>34</sup> While MathML has always supported content markup, it has never supported the definition of new semantic *symbols*<sup>35</sup>. This role has been filled by OpenMath and its ability to define ontologies, so-called *content dictionaries* (CDs), which introduce new symbols. A CD is a collection of (usually closely related) mathematical symbols. As an example, part of the *arith1* CD is shown in [listing 2.5](#). While every OpenMath user is free to define his own CDs for his purposes, the OpenMath Society maintains a collection of *official* CDs [DLo8] that have undergone a review process [BCC+04, section 4.5].

<sup>33</sup>The Lurch tool for writing and validating mathematical documents (see also [section 9.5.1](#)) even encodes complete documents as OpenMath objects, using a combination non-standard content dictionaries and literal XML strings for HTML-like document markup [Lur]. I consider document markup languages, such as OMDoc, DocBook, or XHTML+RDFa, a more appropriate choice due to wider tool support.

<sup>34</sup>MathML 3 having become a W3C recommendation requires a number of adaptations to the OpenMath 2 standard. This may result in a second edition of OpenMath 2, or in an incremented version number.

<sup>35</sup>in the sense of [section 2.1.2](#)

Listing 2.4: The mathematical object  $a_1 + \frac{1}{2}$  in non-strict Content MathML, strict Content MathML, and OpenMath XML

```

<apply>                                <!-- non-strict Content MathML -->
  <!-- short names for common operators -->
  <plus/>
  <!-- mixed presentation and content markup -->
  <ci><msub><mi>a</mi><mn>1</mn></msub></ci>
  <!-- built-in constructors for common types -->
  <cn type="rational">1<sep/>2</cn>
</apply>

<apply>                                <!-- strict Content MathML -->
  <!-- all symbols referenced by CD and name -->
  <csymbol cd="arith1">plus</csymbol>
  <semantics>
    <ci>a1</ci>
    <!-- annotation (same pattern as for parallel markup) -->
    <annotation-xml encoding="application/mathml-presentation+xml">
      <msub><mi>a</mi><mn>1</mn></msub>
    </annotation-xml>
  </semantics>
  <apply>
    <csymbol cd="num1">rational</csymbol>
    <cn type="integer">1</cn>
    <cn type="integer">2</cn>
  </apply>
</apply>

<OMA>                                <!-- OpenMath XML (structurally similar) -->
  <OMS cd="arith1" name="plus"/>
  <!-- attributed term -->
  <OMATTR>
    <OMATP>
      <!-- a key/value pair; key always is a special symbol -->
      <OMS cd="OMPres" name="PMML"/>
      <!-- embedding non-OpenMath content -->
      <OMFOREIGN>
        <m:msub><m:mi>a</m:mi><m:mn>1</m:mn></m:msub>
      </OMFOREIGN>
    </OMATP>
    <OMV name="a1"/>
  </OMATTR>
  <OMA>
    <OMS cd="num1" name="rational"/>
    <OMI>1</OMI>
    <OMI>2</OMI>
  </OMA>
</OMA>

```

CD authors are not forced to fix the formal semantics of symbols: the only mandatory information to be given about a symbol is its name and an informal description. Optionally, but recommended, mathematical properties of a symbol can be described informally (*CMP* = commented mathematical property) or formally (*FMP* = formal mathematical property), i.e. using OpenMath objects. Despite a number of proposals, which have been discussed throughout the last 10 years (see, e.g., [CO01]), definitional *FMPs* have not yet made it into the OpenMath standard, i.e. there is no way of distinguishing defined from asserted properties. Besides the mathematical structure – CDs containing symbol definitions containing properties –, there are metadata, for which OpenMath uses an idiosyncratic vocabulary. Parts of it, however, can be mapped on existing vocabularies, such as Dublin Core, as explained in section 3.4. An abstract data model for CDs and an XML-based reference encoding are part of the OpenMath standard. Besides the proper CD file (named e.g. *number-theory.ocd*), there can be additional files: OpenMath does not commit to a particular *type system*, so it allows for types of symbols to be specified in separate files parallel to the CD, one per type system. The most common type system in the OpenMath community is the Small Type System (STS [Dav99]); types in that system would be given in a file named *number-theory.sts*.

In the traditional application of OpenMath as a CAS interchange format, the semantics of symbols is fixed by translating OpenMath objects into respective CAS-internal representations: each OpenMath-aware CAS can choose to support a number of CDs, and then has to specify a *phrasebook* that translates between the symbols of these CDs and an internal representation understood by the CAS. The result of this translation must satisfy all *FMPs* declared for the symbols involved.

Summarizing, OpenMath is fully capable of expressing logical structures of mathematical knowledge on the object level, and – by way of CDs – partly on the statement and theory levels. The expressivity on the latter two levels is limited in that it neither supports a full axiomatic formalization of symbols nor a structured way of reusing symbols across CDs, other than referencing arbitrary symbols in the descriptions of mathematical properties of other symbols. With *CMPs* and *FMPs* there are two degrees of formality with no transition in between. Description fields and *CMPs* only permit text content<sup>36</sup>, whereas *FMPs* only permit OpenMath object content. CDs are structured like hierarchical databases, not like narrative documents. OpenMath is not intended to represent document structures; instead, one would embed OpenMath objects (or Presentation MathML annotated with OpenMath) into other languages that can express document structures. Improving over these shortcomings of OpenMath has been the primary motivation for developing OMDoc, which the following section introduces. As OpenMath objects are directly reused in OMDoc, as OpenMath CDs can be interpreted as OMDoc theories [KR09], and as OMDoc theories satisfy the specification of abstract OpenMath CD (cf. [BCC+04, chapter 4.2] and [Koho6b, chapter 15.6.2]), OpenMath can semantically be considered a subset of OMDoc.

#### 2.4.4 OMDoc

OMDoc (Open Mathematical Documents [Omd; Koho6b]) integrates and extends MathML and OpenMath. Above the layer of *objects*, it adds knowledge representation for *statements*,

<sup>36</sup>It has been proposed to extend their content model to text mixes with OpenMath objects.

Listing 2.5: Definition of the *plus* symbol within the *arith1* CD

```
<CD>
  <CDName>arith1</CDName>                                <!-- mandatory information is bold -->
  <CDBase>http://www.openmath.org/cd</CDBase>             <!-- the base URI of the CD -->
  <CDURL>http://www.openmath.org/cd/arith1.ocd</CDURL>
  <CDDate>2004-03-30</CDDate>                             <!-- date of last revision, and date of next review -->
  <CDReviewDate>2006-03-30</CDReviewDate>
  <!-- status in the review process: private, experimental, official, or obsolete -->
  <CDStatus>official</CDStatus>
  <CDVersion>3</CDVersion>                                <!-- major and minor version number -->
  <CDRevision>0</CDRevision>

  <Description>common arithmetic functions</Description>

  <CDDefinition>
    <Name>plus</Name>
    <Role>application</Role>
    <Description>The symbol representing an n-ary commutative function plus.</Description>
    <CMP>for all a,b | a + b = b + a </CMP>
    <FMP>
      <!-- abstract syntax:  $\beta(\text{quant1}\#\text{forall}, a, b, \text{@}(\text{relation1}\#\text{eq}, \text{@}(\text{arith1}\#\text{plus}, a, b), \text{@}(\text{arith1}\#\text{plus}, b, a)))$ 
           concrete syntax follows -->
      <OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0"
        cdbase="http://www.openmath.org/cd">
        <OMBIND>
          <OMS cd="quant1" name="forall"/>
          <OMBVAR>
            <OMV name="a"/>
            <OMV name="b"/>
          </OMBVAR>
          <OMA>
            <OMS cd="relation1" name="eq"/>
            <OMA>
              <OMS cd="arith1" name="plus"/>
              <OMV name="a"/>
              <OMV name="b"/>
            </OMA>
            <OMA>
              <OMS cd="arith1" name="plus"/>
              <OMV name="b"/>
              <OMV name="a"/>
            </OMA>
          </OMA>
        </OMBIND>
      </OMOBJ>
    </FMP>
    ...
    <Example>...</Example>
  </CDDefinition>
  ...
</CD>
```

modular *theories*, and narratively structured *documents*. The contribution that this thesis makes to representing semiformal mathematical knowledge is largely based on OMDoc 1.2, the latest stable version, which has been released in 2006. Most of the services for mathematical knowledge presented in [chapter 6](#), as well as the interactive documents presented in [chapter 7](#) and the collaboration environment presented in [chapter 9](#) primarily operate on mathematical knowledge represented in OMDoc or its subset OpenMath.

#### 2.4.4.1 Semantics

A major problem with OMDoc 1.2 is that its semantics is not completely formally defined. Compared to OpenMath CDs, more of the semantics of symbols can be expressed within OMDoc itself; therefore, OMDoc does not have to rely on phrasebooks. Still, the semantics of logical/functional structures, such as symbol declarations, axioms, and proofs, is only defined in a phrasebook-like way, i.e. by partial translation to languages for formalized mathematics. The  $\Omega$ mega [SBA05] and VeriFun [WS03] systems use OMDoc for communication, the latter even as its native file format [Mülo6]. Further interfaces, e.g. to CASL (Common Algebraic Specification Language [BMo4a]), Twelf [Pfe01], Mizar [BK07], and OWL<sup>37</sup>, exist, in varying states of completeness. Some of them have been bundled in the Heterogeneous Tool Set (Hets [Mos; MMLo7]). The underlying logics have been implemented as theories in OMDoc [Koho6c]. The actual formalization that is of interest can then be developed in OMDoc theories that import the former theories as meta-theories [RK11; Rabo8]; thus, the OMDoc language is independent from a particular logical foundation.

Translations from OMDoc to the target languages and back have usually been implemented using OMDoc’s presentation framework (see below); translations from non-XML languages to OMDoc are made by hooking into the parsers of existing tools and making them output OMDoc [Koho6b, chapter 25.2]. For the rather semiformal structures of documents, including informal mathematical statements, rhetorical structures, document structures, and metadata, no model- or proof-theoretic semantics has been specified at all. The phrasebook approach is harder to pursue here, as there is not a single suitable target language covering all of these structural dimensions, and as the specification language is not always quite rigorous, even partly ambiguous, as the following specification of versioning metadata demonstrates:

Recommended values [for the *dc:date/@action* attribute] include the short forms *updated*, *created*, *imported*, *frozen*, *review-on*, *normed* with the obvious meanings. Other actions may be specified by URIs pointing to documents that explain the action. ([Koho6b, chapter 12.1])

The *omgroup/@type*, *omtext/@type*, and *phrase/@type* attributes discussed in [section 3.3.2](#) provide similar extension hooks by supporting URI values. While the “recommended values” that the OMDoc specification describes informally – as the specifications of most other semantic markup languages do – could be (semi)formalized in an ontology, the main problem for web-scalable machine-comprehensibility is that no assumption is made about the “documents that explain”

<sup>37</sup>The Hets framework mentioned below can read OWL [KLM+08]. It can also read and write OMDoc, but it can only translate between OMDoc and CASL, not yet between OMDoc and OWL. Independently from that, I have developed a translation from OMDoc to the RDF representation of OWL and back (cf. [section 8.1.3](#)).

these vocabulary extensions; they are not required to be, for example, RDF vocabularies, but could even contain much less rigorous natural language than the OMDoc specification.

If a phrasebook cannot reasonably be created due to ambiguous specification of some features of a language, it is generally hard to provide application support for these features, as the translation of a document into the internal data structures of an application can already be considered a phrasebook. [Chapter 3](#) addresses this problem by formalizing relevant aspects of OMDoc's semiformal structures in an ontology language with a well-defined model-theoretic semantics plus axioms that enable inferring structural properties relevant for applications, and a phrasebook-like translation of OMDoc documents to instances of that ontology.

At the time of this writing, work on a completely revised OMDoc version (tentatively called 1.6; cf. [section 2.4.4.5](#) for a development roadmap), is in progress. The key improvement of version 1.6 over 1.2 will be a completely revised formal core, building on the Module system for Mathematical Theories (MMT) [[RK11](#); [Rabo8](#)]. This core is fully machine-comprehensible – so far to the MMT implementation and, via phrasebooks, other related software systems [[Raba](#); [RK11](#)], but its formal clarity makes it a straightforward task to additionally achieve machine-comprehensibility in a linked data sense. The syntax of the MMT language is heavily influenced by the MMT ontology relying on the Curry-Howard correspondence of proofs and terms and thus quite different from the textbook style of OMDoc 1.2. This motivated the developers of OMDoc to adopt the idea of a pragmatic and strict syntax, which they had developed for MathML before. The pragmatic syntax will largely correspond to OMDoc 1.2, and its semantics will be defined by translation to the strict syntax, which will largely be a concrete XML syntax for the MMT abstract syntax.

For addressing symbols, OMDoc adopts the syntax of OpenMath and MathML, but reinterprets and extends its semantics. A symbol is identified by *cd* and *name*, where *cd* is the name of an imported theory and the *name* is local to that theory. The *cdbase*, i.e. the base URI of the theory graph, is usually not explicitly given for each symbol reference but reconstructed by following the import. MMT extends this by named imports; MMT IRIs<sup>38</sup>, whose basic syntax is *cdbase?cd?name*, allow for referencing reused symbols by relative URIs constructed from their import paths. Besides simple imports, which literally introduce all symbols from the imported theory into the importing theory, OMDoc also supports theory morphisms and views.

### 2.4.4.2 Degrees of Formality

OMDoc allows for representing knowledge in a wide range of degrees of formality. The OMDoc specification gives an example of formalizing an excerpt from a mathematical textbook in the following steps: (i) adding top-level metadata, (ii) marking up text sections and classifying them by type of statement, (iii) representing the structure of mathematical objects using content markup, (iv) full formalization using a suitable logic as meta-theory [[Koho6b](#), chapter 4]. OMDoc inherits MathML's support for parallel markup in mathematical objects and extends it to mathematical text. Not only can the rhetorical structures of text phrases be marked up, but they can also be interlinked with parts of mathematical objects (cf. [listing 4.1](#) for an example).

<sup>38</sup>IRIs are particularly useful for a compact notation of mathematical symbols in semantic markup, as one can declare symbols whose identifier is their mathematical symbol instead of their common name.



### 2.4.4.3 Presentation

OMDoc 1.2 comes with an elaborate framework for presenting semantic markup in human-readable formats (cf. [Koho6b, chapter 19] and [section 2.1.5](#)), which will, however, be replaced in subsequent OMDoc versions, as outlined in [section 2.4.5](#). Additionally, one may directly provide mathematical objects in Presentation MathML, when their semantic structure is not relevant for the respective application, or when they are intended to be formalized later. For structuring informal text, the “rich text” (RT) module supports a subset of the list and table elements known from HTML [Koho6b, chapter 14.6].<sup>39</sup>

### 2.4.4.4 Metadata

OMDoc 1.2 allows for using metadata from a fixed vocabulary. This vocabulary and its semantics are detailed in [section 3.4.4](#), OMDoc’s metadata syntax in [section 5.1](#), and [section 5.2](#) contributes a new, extensible RDFa metadata syntax.

### 2.4.4.5 Development Roadmap

This section summarizes the roadmap for evolving OMDoc in the near future, pointing out what parts of this thesis are based on developments beyond the state of OMDoc 1.2, and what this thesis contributes to the evolution of OMDoc.

A first usable implementation of the new pattern-based presentation framework reviewed in [section 2.4.5.2](#) had already been released in 2008 (cf. [KMRo8] and [appendix C.1.2](#)), the core of the new metadata framework – a contribution of this thesis covered in [chapter 5](#) – had been settled in mid-2009, and both are already being used. Therefore, the core OMDoc developers decided to release an intermediate OMDoc version 1.3, which is essentially OMDoc 1.2 with these two changes. OMDoc 1.3 is likely to be released soon after the finalization of this thesis.

Work on other features being planned for OMDoc 1.6 is still going on. That includes redesigning the formal core of OMDoc (cf. [section 2.4.4.1](#)), an improved way of declaring presentations for symbols, statements, and theories in a style that integrates well with modeling their formal properties (cf. [section 2.4.5.3](#)), as well as formally specifying the pragmatic→strict translation for logical/functional structures (textbook language to MMT; cf. [section 2.4.4.1](#)), notation definitions (declarations to patterns; cf. [section 6.2.5](#)), and metadata (OMDoc 1.2 syntax to RDFa; cf. [section 5.4](#)). Further advancements over OMDoc 1.2, which are compatible with the 1.3 developments and have already been implemented in software, but not yet scheduled for any OMDoc release, comprise context-sensitive presentation and adaptation of documents [Mül10a].

The ActiveMath e-learning system uses yet another variant of OMDoc, which has been forked off OMDoc 1.1 in 2001. Particular improvements over OMDoc 1.1 – and, partly, 1.2, – comprise another way of defining notation by pattern matching [MLU+06], improved markup for exercises [GGPM05], as well as a – still finite – metadata vocabulary enhanced for education, and a more exact specification of metadata inheritance (cf. [sections 2.1.7.2](#) and [3.4.2](#) and [Lib09]). The work on metadata in OMDoc presented in [section 5.2](#) aims at delivering an improvement not only over OMDoc 1.2 but also over ActiveMath’s variant of OMDoc.

<sup>39</sup>In OMDoc 1.2, these elements are in the OMDoc namespace and merely have the same names as their HTML counterparts. In OMDoc 1.3, these elements will be replaced by a larger subset of the actual XHTML vocabulary.

### 2.4.5 Defining Notation in MathML, OpenMath, and OMDoc

Many state-of-the-art knowledge representations for mathematics allow for specifying a custom notation for content markup to be rendered, primarily covering symbols. There is not yet a widely accepted standard representation for that. The MathML specification, for example, merely *suggests* “typical renderings [of Content MathML as Presentation MathML] by way of examples” [ABC+10, chapter 4]<sup>40</sup> and provides a non-normative XSLT implementation (see below).

The following sections review existing approaches to defining notations – mainly those that focus on the direction of *rendering* – to the extent the services presented in [part III](#) support them.<sup>41</sup> Approaches that support *parsing* human-readable/writable input into a functional, content-oriented representation are beyond the focus of this thesis. Besides defining notations directly on the XML level, there are approaches that involve pattern matching, as well as declarative ones.

#### 2.4.5.1 XSLT: Defining Notation on the XML→XML Level

As rendering algorithms for XML content markup have often been implemented in XSLT (cf. [section 2.3.2.3](#)), the notation of symbols has traditionally been directly defined in that language. Advantages of XSLT are its expressivity and its wide acceptance as an XML→XML transformation standard, for which many efficient implementations exist. However, XSLT has also been found hard to maintain and inappropriate for capturing the practice and semantics of mathematical notation. Moreover, it is a Turing-complete programming language (cf. [Kep04]) and thus much more expressive than would be required for the content→presentation transformation.

The probably largest collections of notations natively defined in XSLT are the above-mentioned translation of Content MathML to Presentation MathML [W3Co3]<sup>42</sup>, and a related collection that defines 143 notations for the symbols of the official OpenMath 2 CDs [Opeb].

#### 2.4.5.2 Pattern Matching

Pattern matching notation definitions map patterns of content markup to fragments of presentation markup, with placeholders for structures matched by the pattern. The presentation markup fragment resembles the body of an XSLT template; the matching is usually done using literal XML. [Listing 2.6](#) demonstrates a notation definition for the binomial coefficient  $\binom{n}{k}$  (the *combinatorial binomial* symbol of OpenMath)<sup>43</sup> in the OMDoc 1.3 notation definition language, which is supported by several services presented in [part III](#). The related language of ActiveMath [MLU+06] will not be considered here in further detail. The OMDoc 1.3 notation definition language supports multiple presentation markup fragments, annotated with the *presentation context* in which they apply (cf. [section 2.1.5](#) and [KMRo8; Mül10a]). A formal semantics has been specified for an abstract syntax that corresponds to this concrete XML syntax by mapping notation definitions to a rendering algorithm [KMRo8]. [Listing 2.6](#) shows two language-dependent renderings of the

<sup>40</sup>This applies to pure Content MathML. Where Content MathML is annotated with parallel Presentation MathML markup, the latter has to be used when presenting a mathematical object.

<sup>41</sup>SHAHID MANZOOR et al. review some of the early proposals for non-XSLT notation definition languages not mentioned here [MLU+06].

<sup>42</sup>currently reflecting the state of MathML 2

<sup>43</sup>The binomial coefficient can alternatively be rendered as a fraction with an invisible stroke; the latter rendering is, however, less comprehensible to assistive technologies, as discussed in [Mül10a].

Listing 2.6: A pattern matching notation definition for the *combinat1#binomial* symbol of OpenMath

```
<notation>
  <prototype> <!-- the content markup pattern -->
    <om:OMA>
      <om:OMS cd="combinat1" name="binomial"/>
      <expr name="arg1"/>
      <expr name="arg2"/>
    </om:OMA>
  </prototype>
  <rendering context="lang:de,en">
    <!-- presentation markup fragment for German and English:  $\binom{n}{k}$  -->
    <m:mfenced>
      <m:htable>
        <m:mtr><m:td><render name="arg1"/></m:td></m:mtr>
        <m:mtr><m:td><render name="arg2"/></m:td></m:mtr>
      </m:htable>
    </m:mfenced>
  </rendering>
  <rendering context="lang:fr,ru">
    <!-- presentation markup fragment for French and Russian:  $C_n^k$  -->
    <m:msubsup>
      <m:mi mathvariant="script">C</m:mi>
      <render name="arg1"/>
      <render name="arg2"/>
    </m:msubsup>
  </rendering>
</notation>
```

binomial coefficient.

### 2.4.5.3 Declarative Notation Definitions

Declarative notation definitions in mathematical markup have, to the best of my knowledge, first been introduced with OMDoc 1.2 [Koho6b, chapter 19.3] and the QMath OpenMath/OMDoc preprocessor [GPo6a]. OMDoc 1.2 uses them for rendering, whereas QMath originally used them for parsing, but the QMath-based Sentido formula editor also uses them for rendering (cf. section 6.2.4). Instead of matching content markup patterns, they refer to a symbol and the *role* in which it occurs, the most frequent ones being a *constant* without arguments, the *application* to arguments, or, a variant of application, as a *binder* for variables in a subterm. Instead of giving presentation markup fragments, they describe presentational *properties* of the operator: the presentational symbol, its *fixity* (pre-/post-/infix), and a set of properties governing bracket elision (see below). An evolution of the declarative syntax of OMDoc 1.2 has been implemented as a part of the MMT language, the core of OMDoc 1.6. A semiformal semantics of that language is specified

in [Raba]; the semantics of an older version has been specified more formally by translation to the pattern syntax mentioned above [KLM+09].

#### 2.4.5.4 Pattern Matching vs. Declarations: Redundancy and Compositionality

Declarative notation definitions are structurally similar to axioms about symbols, whereas pattern-based notation definitions are structurally similar to concrete content or presentation markup. Declarative notation definitions are concise and as little redundant as possible: If an operator has the same appearance in both the constant and the application role, which is the case for most operators<sup>44</sup>, the definition of the notation for the application role can focus on fixity and bracketing and delegate the rendering of the operator symbol to the “constant” notation definition.

Some notations, most prominently non-compositional ones, cannot be handled by declarative notation definitions. For example, the notations  $\sin^2 x$  for  $(\sin x)^2$  cannot be obtained from composing the application of the respective notation definitions for the sine and power operators but require deep pattern matching. The use of content markup leads to further cases of non-compositional notation: Defining the one-dimensional integral as an operator that binds one variable and takes as arguments a set and a lambda abstraction of a function in this variable, e.g.  $\int_S \lambda x. f(x)$  (cf. [SK00]), subsumes the concept of an integral over an interval, but the latter is usually written as  $\int_a^b$  instead of  $\int_{[a,b]}$ , and the integrand is rather written as  $f(x)$ , followed by  $dx$ , instead of  $\lambda x. f(x)$ . A notation definition for that case depends on the argument  $S$  to be an interval and thus is non-compositional.

#### 2.4.5.5 Brackets and Operator Precedences

Most declarative and pattern notation definitions control the elision of redundant brackets in the same way. Brackets around a subterm are redundant when its constructing operator binds stronger than the one of the enclosing term (consider  $ax + y$  vs.  $(ax) + y$ )<sup>45</sup>. The different binding strengths of operators are usually modeled in a total order using numeric precedence values.<sup>46</sup> Simple rendering algorithms assume one such value per operator; more sophisticated ones – including the one employed by the services presented in chapters 6 and 7 – distinguish between the *output precedence* of the subterm constructed by an operator and its *input precedence* per argument [KLR07]: Whenever the operator  $g$  constructing the subterm  $g(b_1, \dots, b_m)$  binds stronger than the operator  $f$  of the enclosing term  $f(a_1, \dots, a_n)$  binds its argument  $a_i = g(b_1, \dots, b_m)$ , brackets around the inner subterm are redundant. Binding strength is determined by comparing the numeric value of the  $i$ -th input precedence of  $f$  to the numeric value of the output precedence of  $g$ .

<sup>44</sup>As an example, consider the addition of  $0 + 1$  (i.e.  $+$  in application role) in the ring  $(\mathbb{Z}, +, \cdot, 0, 1)$  ( $+$  in constant role).

<sup>45</sup>The inner operator in this case is the invisible times operator. This is not an elision, but a rendering for the multiplication operator that merely results in a small amount of whitespace.

<sup>46</sup>As many operators never co-occur in practice, a partial order on operators and sets of operators (e.g. arithmetical operators bind stronger than logical operators), as implemented by the PlatΩ extension of the T<sub>E</sub>X<sub>MACS</sub> scientific editor (see [AFN+07], and sections 6.2.7 and 9.5.1 for further discussion of that editor), is a more faithful reproduction. However, a partial order is prone to an accidental introduction of cycles, which make it collapse locally.

### 2.4.6 MathLang

MathLang [KWZo8] offers “an approach for computerising mathematical texts which is flexible enough to connect the different approaches to computerisation, which allows various degrees of formalisation, and which is compatible with different logical frameworks (e.g., set theory, category theory, type theory, etc.) and proof systems”. MathLang has an XML encoding that is used for most processing tasks. However, due to its verbosity, the XML encoding is not used for authoring and presentation. Authoring is facilitated by a plugin for the  $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$  editor.

Compared to OMDoc, MathLang puts an even higher emphasis on formalization of informal, but highly conventionalized mathematical vernacular. It allows for annotation of mathematical symbols and statements as well as logical structures in text. From these structural annotations, “proof skeletons” can be generated, i.e. templates in languages for formalized mathematics, which have to be completed in the target language [KWZo8]. MathLang neither facilitates reuse by modularity nor supports logical heterogeneity within one document. Metadata beyond MathLang’s native structures are not supported.

The “Core Grammatical aspect” (CGa) of MathLang assigns syntactic categories to elements of a mathematical text: *term*, *set*, *noun*, *adjective* – the building blocks of expressions, which roughly correspond to OMDoc’s object level –; *statement*, *definition*, *declaration*, *step* – roughly corresponding to OMDoc’s definitions and axioms –; and *context* – a means of referring to background knowledge that is assumed, similar to importing theories in OMDoc. “The goal of CGa’s type system is not to ensure full correctness, but merely to check whether the reasoning parts of a document are coherently built in a sensible way.” [KWZo8] The “Text and Symbol aspect” (TSa) interlinks CGa structures with natural language and presentation-oriented formulæ, similarly to OMDoc’s parallel markup.

Thirdly, the “Document Rhetorical aspect” (DRa)<sup>47</sup> represents larger chunks of mathematical text – document sections as well as mathematical statements, such as definitions, theorems, and proofs, – and their interrelations, such as that a proof justifies a theorem [Ret09; KWZo8]. This is similar to the statement level of OMDoc and our OMDoc ontology (cf. section 3.2.2). A generic dependency relation has been defined for the DRa, which can be used for validating whether the narrative order of a document respects the logical dependencies. The DRa vocabulary has been implemented as an OWL ontology (cf. section 2.4.10.2 and [Ret09; KWZo8]). This ontology serves as a formal specification of the DRa semantics, whereas the validator processes an XML representation of the DRa [Ret09].

Finally, MathLang does support annotating almost every element of a document with metadata, but the “vocabulary” is restricted to four terms related to type checking [Tilo6].

### 2.4.7 $\text{T}_{\text{E}}\text{X}$ , $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ and $\text{S}_{\text{T}}\text{E}_{\text{X}}$

$\text{T}_{\text{E}}\text{X}$  is not only a presentation markup language for high-quality typesetting (e.g. to PDF). Due to its macro processing abilities, it has also been characterized as a framework for specifying application-specific document formatting vocabularies.  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ , the most widely used vocabulary, is largely presentation-oriented.  $\text{S}_{\text{T}}\text{E}_{\text{X}}$  is a vocabulary for mathematical content markup. Another

<sup>47</sup>Despite the name, this is not related to rhetorical structures in the sense of RST, as introduced in section 2.1.3.

notable vocabulary that does not target mathematics and is therefore not covered in detail here<sup>48</sup> is SALT (Semantically Annotated L<sup>A</sup>T<sub>E</sub>X [GHM+07; GMH+07]): It allows for marking up rhetorical structures, as introduced in section 2.1.3, as well as fine-grained citations not of complete publications, but of individual claims.

### 2.4.7.1 L<sup>A</sup>T<sub>E</sub>X

L<sup>A</sup>T<sub>E</sub>X offers little semantic markup. There is no notion of a theory level. On statement level, there are some semantic constructs – the “theorem-like” environments, such as `\begin{theorem} ... \end{theorem}`. On the object level, a few operators have a content-oriented syntax, such as `\frac{num}{den}` or `\binom{n}{k}`. Bad usage practices are widespread, as authors are generally lazy to write semantically correct code as long as it just looks right. Two examples are not using the  $\mathcal{M}$ S packages that offer more semantic macros, e.g. for matrices, and not realizing that the syntax “fun” should not be used to denote a function named “fun”, as the renderer – rather heuristically – assumes that the letters denote three variables ( $f, u, n$ ) connected by invisible times operators and therefore inserts small spaces between them, yielding  $fun$ <sup>49</sup>.

**Example 4** What does  $O(n^2 + 1)$  – in L<sup>A</sup>T<sub>E</sub>X syntax: `O(n^2+1)` – mean?<sup>50</sup>

**Landau symbol:** the set of all functions that asymptotically grow at most as fast as  $n^2$ , where the  $+1$  is actually superfluous

**Function application:** the application of *some* function named  $O$  – which needs not be the Landau set constructor function – to  $n^2 + 1$

**Invisible times:**  $O$  (e.g. some variable) multiplied with  $n^2 + 1$ , where the multiplication operator is invisible

### 2.4.7.2 S<sub>T</sub>E<sub>X</sub>

S<sub>T</sub>E<sub>X</sub> [Koho8d] is essentially a T<sub>E</sub>X syntax for OMDoc, with a few minor differences mainly owed to a loose alignment of the two development roadmaps. It is intended as an “invasive technology” in that it allows for gently migrating non-semantic (L<sup>A</sup>)T<sub>E</sub>X documents into semantically structured documents, and that it brings OMDoc to users of “legacy” tex(t) editors, which has successfully demonstrated by plugins for the Emacs text editor and the Eclipse integrated development environment (cf. section 6.2.1.1).

In a process called “semantic preloading”, the author of an S<sub>T</sub>E<sub>X</sub> document defines semantic macros for the mathematical symbols to be used, which expand into presentational T<sub>E</sub>X. The following snippet introduces a symbol with one argument [Koho8d]:

```
\symdef{uminus}[1]{\prefix{-}{#1}}
```

For more complex operators, S<sub>T</sub>E<sub>X</sub> has its own declarative notation definition macros. In contrast to a mere L<sup>A</sup>T<sub>E</sub>X `\newcommand`, S<sub>T</sub>E<sub>X</sub>’s symbol definitions are scoped to theories (called “modules”

<sup>48</sup>However, I have reused its ontology, as explained in section 3.3.

<sup>49</sup>To the eye of the well-trained T<sub>E</sub>X reader, this does, of course, *not* look right.

<sup>50</sup>adapted from a presentation given by BASTIAN LAUBNER on 2007-10-30



in  $\S\TeX$ ). In addition to merely providing a semantically structured input syntax and rendering presentational  $\TeX$ , they also understand operator precedences and thus precedence-based bracket elision.

For statement, theory, and document level markup,  $\S\TeX$  predefines a large collection of macros that closely correspond to their OMDoc counterparts.  $\S\TeX$  commands roughly correspond to empty OMDoc XML elements, whereas  $\S\TeX$  environments roughly correspond to OMDoc XML elements with text content or children. An elaborate example is given in the documentation for  $\S\TeX$ 's proof module [Koh10d].

The  $\LaTeX$ XML  $\TeX \rightarrow \text{XML}$  converter [Mil] is used to generate OMDoc from  $\S\TeX$ . For each (s) $\TeX$  package, a  $\LaTeX$ XML binding is provided – a set of Perl declarations or functions that map  $\TeX$  macros to XML elements, in this case  $\S\TeX$  macros like `\symdef`, or the statement-, theory-, and document-level macros, to OMDoc elements. This conversion path works reliably on MICHAEL KOHLHASE's lecture notes, a collection of 1959  $\S\TeX$  documents containing 2021 theories that declare 2252 symbols<sup>51</sup>. The reverse direction, however, has not been implemented; see section 6.2.2 for a discussion.

## 2.4.8 Languages for Books and Manuals

A formal view on technical specifications, e.g. of software, reveals structural similarities to mathematical theories; development graphs, for example, originate from the intersection of both fields (cf. section 2.1.2). While fully formalized specifications can be written in the same languages as general formalized mathematics (cf. section 2.4.9) and then be verified automatically, there are different languages targeting human audiences, such as engineers implementing a specification, or developers using an API; DocBook, TEI, and DITA are reviewed here. Similar languages exist for e-books and courseware; this section reviews EPUB/DTBook and CNXML/CollXML. I first briefly introduce each language and then review their suitability for mathematical knowledge.

### 2.4.8.1 DocBook – Technical Manuals

DocBook, the most widely used XML language for technical manuals [Walo8; WMo8], focuses on representing structures pertinent to its main application area of software documentation. Tools are usually provided as extensions for XML editors. DocBook has originally been designed for linearly arranged software manuals, but it also supports reuse of document modules, and its XML schema is designed to be customized for other applications. It is well documented how to customize DocBook by adding or removing elements or attributes or changing their allowed content [WMo8, chapter 5]. The existing XSLT stylesheets for rendering DocBook have analogous extension hooks.

### 2.4.8.2 TEI – Humanities, Social Sciences and Linguistics

Given that the rhetorical and document structures that this thesis also deals with not only occur in mathematical or technical documents, TEI (Text Encoding Initiative [BB09]) as an approach similar to DocBook but rather addressing humanities, social sciences, and linguistics also deserves being mentioned here. TEI has originally been developed to support the digitalization and edition

<sup>51</sup>figures of August 2010

of documents born on paper. Therefore, it supports very fine-grained annotations down to words and characters. The TEI schema and XSLT stylesheets basically offer the same customization possibilities as DocBook; additionally there is a web assistant for creating custom schemata [MRB].

The TEI SIG (special interest group) on ontologies [Tei] is working on aligning sub-vocabularies of TEI with relevant domain ontologies. So far, they have established a mapping from the TEI vocabulary for names, dates, people, and places [BB09, chapter 13], for manuscript description (chapter 10), and the document header (chapter 2) to the CIDOC CRM (Conceptual Reference Model), the standard ontology for cultural heritage information [OE09].

### 2.4.8.3 DITA – Topic-based Documentation

DITA (Darwin Information Typing Architecture [Dit]) does not support any particular application scenario by default but is rather intended to offer a framework for developing languages for topic-based technical documentation that are specialized to a particular domain of application.<sup>52</sup> The topic-based paradigm is in contrast to DocBook's focus on contiguous, narratively ordered manuals. DITA offers an even higher degree of modularity and extensibility than DocBook, but still enjoys less tool support.

### 2.4.8.4 EPUB and DTBook – E-Books

EPUB, formerly known as Open eBook, is a standard for general-purpose e-books, not primarily technical manuals. It consists of three subspecifications: (i) OCF, the Open eBook Publication Structure Container Format, is a ZIP-based container format that bundles all files of a book into an exchangeable archive [Ocf]. (ii) The structure of the whole e-book bundle and its top-level (Dublin Core) metadata are described in a file in the XML-based Open Packaging Format (OPF [Opf]). (iii) Finally, the Open Publication Structure specification (OPS [Ops]) defines the encoding of single content files, such as chapters or the whole book, where most of the actual markup is reused from existing languages. A subset of XHTML can be used, but DTBook (DAISY<sup>53</sup> Digital Talking Book [Dai]) is recommended. DTBook is a semantically structured format inspired by DocBook but simpler and with less support for customization. In contrast to DocBook, it is only possible to introduce custom elements – but not attributes – in a restricted number of places.

### 2.4.8.5 CNXML and CollXML – Course Modules and Collections

CNXML, the language of the course modules of Connexions (cf. [section 1.4.2](#)) [Cnxc], is comparable to a subset of DocBook in expressivity. CNXML natively supports section structures, cross-references, bibliographies, glossaries; thanks to its educational focus, it also supports exercises. The Rhaptos CMS, which drives Connexions, supports online editing of CNXML and can import and convert non-CNXML content, such as office and  $\LaTeX$  documents.

CNXML has not been designed for representing larger units of knowledge, such as books. Course modules written in CNXML are intended to be combined into collections. Collections are managed via a form-based user interface of Rhaptos but internally represented in the CollXML

---

<sup>52</sup>That is where the reference to CHARLES DARWIN comes from.

<sup>53</sup>Digital Accessible Information Systems



**Module: Test Module 1** Hide sidebars

In: [Personal Workspace](#)

[Edit](#)
[Files](#)
[Metadata](#)
[Roles](#)
[Links](#)
[Preview](#)
[Publish](#)

### Edit module metadata

- **Module ID:** (not published)
- **License:** [Creative Commons Attribution License](#) (CC-BY 2.0)
- **Version:** \*\*new\*\*
- **Created:** Jan 3, 2006 3:24 pm
- **Revised:** Jan 3, 2006 3:24 pm

**Roles**

- **Authors:** [Christoph Lange](#)
- **Maintainers:** [Christoph Lange](#)
- **Licensors:** [Christoph Lange](#)

[Edit Roles](#)

**Title** ▾  
Enter the title of this module.

**Language** ▾  
Select the primary language for this module.  
 ☐ Choose a regional variant

**Subject**  
Select the subject categories that apply to this module.

☐ Arts
 ☐ Mathematics and Statistics  
☐ Business
 ☒ Science and Technology  
☐ Humanities
 ☐ Social Sciences

**Keywords (one per line)**

Connexions  
 CNXML  
 module  
 metadata

**Summary**  
Enter a summary of the module. May contain a limited set of CNXML. ([help](#))

This module is intended to test <term url="http://cnx.org">Connexions</term>' module and collection management.

**Google Analytics Tracking Code**  
Enter the Google Analytics Tracking Code (e.g. UA-xxxxxx-x) for this module to track its usage. ([help](#))

[Save](#)

Figure 2.6: Connexions module metadata editor

container format [Cnxc]. CollXML was preceded by a partial representation of a collection's structure in RDF [Rha]. A CollXML document refers to specific versions of the modules it is comprised of and models dependencies between modules – so-called “featured links”, which can be of type “prerequisite”, “supplemental”, or “example”, in three degrees of strength. Similarly, metadata of modules and collections are edited via a form-based user interface (cf. figure 2.6), but internally represented in a “metadata modeling language” (mdml [Cnxc]). The metadata vocabulary is fixed to authors and their roles, subjects, source references, structured revision histories, and unstructured licensing information. mdml has an idiosyncratic vocabulary, which does not reuse existing metadata vocabularies, such as DCMI Terms. Users can export collections as PDF, which is generated via  $\text{\LaTeX}$ . Authors and developers can export collections as ZIP files – containing the CollXML description, the individual CNXML modules, as well as any multimedia files used –, edit them locally, and reimport them into the system.

#### 2.4.8.6 Suitability for Mathematical Knowledge

None of these formats is directly suitable for representing most of the structures of mathematical knowledge reviewed in section 2.1.

All of them support MathML for mathematical objects in some way. CNXML supports MathML and particularly recommends using Content MathML. DTBook officially recommends using MathML for objects and offers a MathML extension module for validating documents [SKo8]<sup>54</sup>. The TEI guidelines recommend using any available representation language for mathematics, according to the requirements, and explicitly mentions MathML, OpenMath, and even OMDoc [BB09, chapter 14.2]; an extension module for validating MathML inside TEI documents is available. DocBook allows for MathML as an optional alternative to a built-in simpler representation of mathematical objects. A notable application of DocBook in MKM is the MathDox e-learning system [Matb; CCK+08], whose compound document format combines multiple XML schemata: DocBook for the basic structure of documents, Jelly for programming constructs, OpenMath for mathematical expressions, XForms for requesting user input, MONET queries (cf. section 1.4.3.2) for interacting with mathematical web services, and MathDox-specific markup for exercises. Finally, MathML can be introduced into DITA via the generic specialization mechanism [Nor09].

Some formats have limited support for statement-level logical structures. CNXML supports definitions, rules – comprising, e.g., axioms and theorems – and examples [RSCo9]. DocBook supports titled equations and examples. DITA supports [definitions of] concepts and examples. None of the formats reviewed supports mathematical theories.

For introducing further semantic markup for higher-level structures of mathematical knowledge, there are two principal approaches: (i) literally reusing elements of sufficiently expressive mathematical markup languages, such as OMDoc, or, (ii) reusing an appropriate ontology for mathematical structures, such as the ones mentioned in sections 2.4.10 or 3 – provided that the host language supports referencing arbitrary metadata vocabularies on any relevant structural level without first introducing new container elements for them via (i), i.e. if there is an RDFa-like infrastructure (cf. section 2.3.3.4). In DTBook, following approach (i) is not practically possible, as the points where extensions can be hooked in are determined by presentation – block vs. inline

---

<sup>54</sup>In this subsection, specific information about mathematics support in the respective language is cited explicitly. Otherwise, the literature references from the introductions of the languages apply, as given above.

display – rather than semantics. [Approach \(ii\)](#) is not feasible out of the box either, as DTBook only has a fixed metadata vocabulary built in; however, RDFa support is planned for the next versions of EPUB and DTBook. Each of DocBook, TEI, and DITA is sufficiently extensible to support [approach \(i\)](#). One can even integrate the attributes of RDFa into DocBook and DITA via that extension path (cf. [section 5.3](#) and [DuCo9]).

DocBook, TEI, and DITA offer varying degrees of support for [approach \(ii\)](#). DocBook’s built-in metadata vocabulary is constrained to an expressivity similar to that of DCMI Terms (cf. [section 2.1.7.3](#)); its semantics is only specified informally, though, not by a mapping to DCMI Terms. There is a workaround for adding RDF-compatible annotations to DocBook: Any DocBook element can carry XLink attributes, which can have a role (= predicate) and target (= object), and from which RDF can be harvested [Danoo]. TEI has an elaborate but finite metadata vocabulary for representing the provenance of documents and even smallest fragments of text, such as a word that has been corrected by an editor or a sentence that was obtained from digitizing an area of an image. TEI documents can reference external objects by XPointers, but without any possibility to specify a predicate type; therefore, RDF cannot be embedded into TEI in the same way as in DocBook. DITA’s built-in metadata vocabulary primarily focuses on the intended context for (re)using objects, such as the intended audience or keywords. For any data that do not fit into this schema, there is the *othermeta* element for arbitrary key–value pairs, for which URIs could be used to emulate RDF, or, even more appropriately, the *data* element, which allows for constructing nested data structures and supports RDF’s distinction of URI- and literal-typed object, as well as datatypes. Similarly, DITA supports typed links<sup>55</sup> from topics to related topics, with the possibility to use link types beyond the built-in vocabulary.

### 2.4.9 Languages for Formalized Mathematics

Existing languages for formalized mathematics are usually not markup languages, as they do not annotate plain text. However, they have certain features in common with markup languages and are therefore briefly covered here.

Most languages for formalized mathematics are native languages of a proof assistant, such as Mizar [Mizb], Isabelle [WBB+09], or Coq [Coq]. Thus, they obviously support logical/functional structures of mathematical knowledge on the object and statement levels, but less so on the theory level (cf. [Rabo8, chapter 1.3]). For storage and exchange purposes, there is often an XML encoding, to which the systems can at least export (cf. [Rabo8]). Except for notation definitions, there is little support for other structural dimensions, such as rhetorical structures – or at least natural language documentation –, document structures, and metadata. The “lowest common denominator” is to put such information into comment lines, which are post-processed by a specialized tool. Mizar formalizations have been published as journal articles ever since [For], using an automatic translation to  $\text{\LaTeX}$ . Similarly, Isabelle and Coq support  $\text{\LaTeX}$  and HTML export. Authors can intersperse formalized content with informal text and mark and certain parts of the formalized content as hidden from human-readable output. In Isabelle, informal text can contain formalized expressions as antiquotations, which the proof assistant evaluates when exporting the document [WBB+09, chapter 4]. These export facilities, however, follow exactly

<sup>55</sup>The type of a link is called “role” in DITA; “type” has a different meaning.

the order of the formal structure. The maze literate proving prototype for Mizar is potentially more flexible in that it outputs intermediate content in any XML vocabulary of the author's choice, which can then be post-processed via XSLT to obtain the final output [CGo6]. FoCaLiZeDoc (formerly named FoCDoc), the documentation generator for the FoCaLiZe language, a formal specification and proof language for developing certified software based on Coq, comes with its own documentation generator, also generates an intermediate XML document, which, similarly to Isabelle, also contains type and dependency information inferred from the original source [Foc; MPo3].

### 2.4.10 RDF with Structural Ontologies

The representation languages reviewed so far treat mathematical documents with knowledge as primary data (cf. the discussion in section 2.1.1.3). They consider a specific set of mathematical structures relevant and offer dedicated language constructs for them, plus a restricted metadata vocabulary for some other structures. RDF takes the contrary approach of representing *everything* as metadata and not privileging any particular structural dimension.

This section reviews RDF vocabularies that for representing mathematical knowledge, grouped by their approaches to representing mathematical objects, as that accounts for the largest differences. Complete representations of mathematical knowledge in RDF (cf. section 2.4.10.1) have largely been unsuccessful in MKM, the limited mathematics vocabularies supported by certain RDF-based reasoning engines being an exception. Partial RDF representations of certain properties of mathematical objects as standoff markup pointing to a primary XML representation have been used successfully (cf. section 2.4.10.2). Embedding RDF(a) into XML has not been done in MKM so far; section 2.4.10.3 discusses the potential.

#### 2.4.10.1 Complete RDF Representations: N3 Vocabularies, RDF Encodings of Content MathML, and the Semantic Memory

The cwm [BLo9] and Euler [DR10] first-order reasoning engines natively use an N3 knowledge representation [BLo6b], i.e. a superset of RDF. The standard N3 vocabularies cover a limited subset of object- and statement level structures, constrained to FOL as a meta-theory. Beyond domain knowledge, i.e. a library of basic mathematical functions, roughly corresponding to the *arith1*, *relation1*, and *trans1* OpenMath CDs, the N3 “math” vocabulary provides weak formalizations of general structural concepts such as the concept of a function. When a concrete function  $f$  is used as the predicate of an RDF triple, whose subject is an RDF collection ( $x_1 \dots x_n$ ) holding the arguments, the reasoner infers  $f(x_1, \dots, x_n)$  as the value of the object. When the object is identified by a URI or blank node ID, it can be reused in the subject of another mathematical expression. Listing 2.7 shows a sample set of facts and rules, from which Euler would infer `:ABC :side3 5`<sup>56</sup>; listing 2.8 shows the same in plain RDF<sup>57</sup>. Few RDF processors support the full N3 syntax. When an N3-aware processor is not available, or when RDF in a different serialization, such as RDFa, is used, the  $n$ -ary ordered tree structure of mathematical objects has to be broken down into explicit RDF

<sup>56</sup>Cwm cannot handle this example, as it does not support non-integer exponents.

<sup>57</sup>... except for the non-RDF quantifiers and the RDF collections. The bracketed syntax for the latter is syntactic sugar for a combination of *rdf:first* (head), *rdf:rest* (tail), and *rdf:nil* (empty list).

Listing 2.7: The Pythagorean Theorem in N<sub>3</sub> (with syntactic sugar)

```

@prefix : <#> .          # for convenience, so that we don't have to write <#ABC>

:ABC :side1 3 ;
      :side2 4 .

{ ?triangle :side1 ?a ;
  :side2 ?b .
  ?c is math:exponentiation of
    (((?a 2)!math:exponentiation
      (?b 2)!math:exponentiation)!math:sum 0.5) . } => { ?triangle :side3 ?c } .

```

Listing 2.8: The Pythagorean Theorem in N<sub>3</sub> (plain RDF)

```

@prefix : <#> .

:ABC :side1 3 ;
      :side2 4 .

@forAll :a, :b, :c, :triangle .
{ @forSome :a2, :b2, :c2 .
  (:a2 :b2) math:sum :c2 .
  (:c2 0.5) math:exponentiation :c .
  (:a 2) math:exponentiation :a2 .
  (:b 2) math:exponentiation :b2 .
  :triangle :side1 :a ;
            :side2 :b . } log:implies { :triangle :side3 :c } .

```

triples, which are harder to author than, e.g., the XML structure of Content MathML. Moreover, N<sub>3</sub>'s first-order quantification exceeds the expressivity of RDF. Combining RDF reification and N<sub>3</sub>'s "reason" vocabulary, which models the structure of proofs, allows for partially capturing the statement level. The coverage of the N<sub>3</sub> vocabularies is determined by the needs of a FOL reasoner and thus not suitable for representing *arbitrary* mathematical knowledge. The semantics of mathematical functions is not fully specified in N<sub>3</sub>; cwm and Euler merely have built-in support for evaluating them. Thus, representing mathematical objects in RDF does not necessarily replace phrasebooks that translate these representations to the languages of, e.g., higher-order proof assistants or CAS.

Two RDF encodings of Content MathML have been suggested independently from N<sub>3</sub>. These representations look similar to N<sub>3</sub>, except that the application of a function is usually modeled with the [reified] application being the subject and the function symbol and the arguments being the object(s). That makes nested expressions easier to write without the additional syntactic sugar of N<sub>3</sub>. An encoding proposed by MASSIMO MARCHIORI [Mar03]<sup>58</sup> has obvious design flaws –

<sup>58</sup>His encoding differs from the N<sub>3</sub> encoding in that order is represented using RDF's built-in container membership properties *rdf:\_n* ( $n = 1, 2, \dots$ ) instead of RDF collections, but that is a secondary issue.

such as introducing, for no obvious reason, two different ways of referencing symbols in CDs and applying them to arguments –, which another representation independently developed by ANDREW ROBBINS avoids [Rob09]. Both suggestions have neither been implemented nor taken up by the MKM community.<sup>59</sup>

As an advantage of representing mathematical objects in RDF, MARCHIORI points out that it allows for making references to bound variables more explicit: Indeed, a bound variable is always represented as a unique RDF resource, be it on declaration or on usage. Content MathML, however, optionally supports a similar explication by making occurrences of the bound variable refer to the place where it is declared via *@xref* and *@id* attributes. MARCHIORI developed an ad hoc vocabulary from the Content MathML element and attribute names, which has not been implemented as an ontology and thus has little value for information retrieval and reasoning. ROBBINS only uses a special vocabulary for the object constructors of Content MathML but the canonical OpenMath CD URIs (e.g. <http://www.openmath.org/cd/arith1#plus>) for symbols. The latter can be published as machine-comprehensible linked data, as explained in section 6.4.1.

Another full representation of (formal) mathematics in an RDF-like data model is the “semantic memory” [NS10]. Statements of the form *subject.property = object* are represented in a matrix, which has, in row *subject* and column *property*, the entry *object*. Reasoning tasks are performed by a “semantic Turing machine”, which treats the matrix as a two-dimensional tape [NS09].

### 2.4.10.2 Partial RDF Representations: XML Literals and Standoff Markup

In several other efforts of developing ontologies for mathematical knowledge, the responsibility for utilizing the full semantics of mathematical objects was left to specialized tools, such as CAS or proof assistants. As certain retrieval, matching, and other management tasks for mathematical objects can be performed reasonably on the RDF level, *partial* information from the objects – anything deemed relevant for the particular application – was represented in RDF in addition to their full XML representation. One can distinguish two different approaches:

**RDF with XML Literals:** RDF is still used as the primary knowledge representation. Within this RDF graph, the full objects are represented as XML literals.

**XML with Standoff RDF Metadata:** XML is used as the primary knowledge representation. A complementary RDF description is given as standoff metadata. These are maintained in an external RDF graph that points into the XML documents. With this division of responsibilities, any ordered or *n*-ary structures, which would be harder to represent in RDF (cf. section 2.3.3.3), are usually exclusively represented in XML. Note that that does not only affect mathematical objects, but also, e.g., narrative document structures.

The following sections briefly discuss all known examples for these approaches.

<sup>59</sup>A possible explanation in MARCHIORI’s case is that his proposal did not originate out of the MKM community but that he was an external (semantic web) expert invited to give a keynote, which consisted of a rather ad hoc sketch of possible applications of semantic web technology to MKM.

**RDF with XML Literals in OpenMath:** In an early phase of the MONET project (cf. [section 1.4.3.2](#)), an RDFS vocabulary for representing OpenMath CDs was developed [Bus01]. The CD and statement levels (*CMs*, *FMPs*, and *Examples*), as well as CD-level metadata were represented by classes and properties. The DCMES vocabulary was partly reused, where it offered suitable counterparts to OpenMath’s metadata elements (e.g. for CD and symbol descriptions). The content of *FMPs* and *Examples* was represented as an XML literal. For one selected aspect of *FMPs*, there was a dedicated RDF representation: for the information about what CDs they used symbols from. While it is not documented why this approach has not been pursued further in the course of the MONET project, one of its remaining drawbacks is that support for extending RDF queries down into XML literals is still scarce (cf. [section 2.3.3.5](#)).

**Standoff Markup in MONET:** In their final stage, the MONET ontologies do not primarily model the logical structures of CDs – except for grouping symbols in a taxonomy by CD – but focus on describing mathematical problems and the software used to solve them. The OWL-based MONET problem ontology focuses on the operator or constructor symbol at the root of the functional tree representation of a mathematical objects as a tree. Suppose the MONET broker knows a web service for computing definite integrals constructed with the *oms:calculus1#defint* symbol [CDTo4a]. The type of problem that that service solves can be modeled as follows:<sup>60</sup>

$$\begin{aligned} \text{problem:definite\_integration} &\sqsubseteq \text{problem:Problem} \\ \text{problem:definite\_integration} &\sqsubseteq \text{gams:GamsH2a} \\ \text{problem:definite\_integration} &\sqsubseteq \forall \text{problem:openmath\_head.oms:calculus1\#defint} \\ &\quad \sqcap \exists \text{problem:openmath\_head.oms:calculus1\#defint} \end{aligned}$$

The deeper structure is only represented in OpenMath; it is not used for service matching, but sent to a matching service for computation.

**Standoff Markup in HELM:** The HELM system (cf. [section 1.4.3.1](#)) generates from an original formalized representation in a non-XML language both a full XML representation and a standoff RDF graph containing a structural outline of properties relevant for searching. As HELM’s design is rooted in the formalized library of the Coq proof assistant, the terminology is slightly different from the one introduced in [section 2.1.2](#); therefore, I first clarify the terms here. The HELM ontologies, implemented in RDFS<sup>61</sup>, distinguish objects and theories. HELM objects roughly correspond to our statements in that they can be definitions, theorems (with proof object bodies, i.e. formalized proofs represented as terms), and axioms. HELM objects can have a body and a type, each of them being a term. Terms correspond to our notion of objects. Inside a term, other

<sup>60</sup>The fact that OpenMath symbols are represented as classes and not as instances, which would have allowed for simplifying the third axiom listed above to *problem:definite\\_integration*  $\sqsubseteq$  *problem:openmath\\_head.oms:calculus1\#defint* is owed to technical restrictions, which the underlying reasoner imposes for scalability reasons; compare the remarks on the Instance Store in [section 1.4.3.2](#). The authors concede that it would have been more appropriate to use instances [CDTo4a].

<sup>61</sup>Note that they were implemented before the final standardization of RDFS and therefore might require revisions if one wanted to use them today. For example, at the time when the HELM ontologies were implemented, RDFS domain and range declarations were still considered constraining (cf. the introductory comment in [BG02]).



HELM objects can occur. The occurrence of a reference to another HELM object in one HELM object is reified as a resource, which has an *h:position* and an integer *h:depth* counting the number of premises, including universal quantifiers. Positions that have been found relevant for answering queries, e.g. for finding applicable theorems for proving something (cf. [Scho2; GSCo3]), are the following, explained using the example of the theorem  $\forall a : \mathbb{N}. \forall b : \mathbb{N}. \forall c : \mathbb{N}. a \leq b \wedge b \leq c \Rightarrow a \leq c$ :

**h:MainHypothesis:** the head symbol of a hypothesis; here:  $\wedge$  (depth 3)

**h:InHypothesis:** any other symbol anywhere else in a hypothesis; here, either of the two  $\leq$

**h:MainConclusion:** the head symbol of the conclusion; here:  $\leq$  (depth 4)

**h:InConclusion:** any other symbol anywhere else in the conclusion; here: nothing else

**h:InBody:** any symbol in the proof of the theorem

HELM models theories completely separately. A theory (*hth:Theory*) is a collection of HELM objects, here represented as instances of *hth:TheoryItem* having an *hth:itemType*, one string out of “Axiom”, “Fact”, “Definition”, “Theorem”, “Lemma”, “Corollary”, “Variable”.<sup>62</sup> In a HELM theory, dependencies among statements are modeled (as subproperties of *hth:dependence*), for example a corollary being a consequence of a theorem (*hth:isConsequenceOf*), or a lemma being a prerequisite of a theorem (*hth:isPremiseOf*).

From the point of view of representing general mathematical knowledge, the HELM ontologies do not sufficiently abstract from the native knowledge representation of the Coq library. The relation from theories to HELM objects, i.e. that theories are collections of HELM objects, is not made explicit at all, not even by having theory items and HELM objects share the same URIs. The identity of theory items is hidden behind an indirection introduced by an idiosyncratic identification mechanism. They are primarily identified by their XPath location in the theory XML document and are assigned an *hth:ident* property of type *hth:HelmsID* used for their dependency interlinking.

**Standoff Markup in MoWGLI:** A metadata model with a wider coverage of mathematical structures, including informal representations and educational content, was developed in the MoWGLI project and implemented as an RDFS ontology [Gog03b]. A closer look shows that it is almost completely unusable due to design errors; nevertheless, it serves as an instructive example of a comprehensive integrated mathematical metadata vocabulary.

The MoWGLI metadata model reused vocabulary from the HELM ontologies, existing general and educational metadata ontologies, the OMDoc XML schema, and the metadata vocabularies of ActiveMath’s OMDoc extension. For the latter two, an RDFS model was newly developed. The MoWGLI metadata model does not assume anything about the representation of the actual data; they could, e.g., be given as OMDoc documents or Coq proof scripts.

Shortcomings of the MoWGLI metadata model include ambiguities and errors in its own modeling, its tampering with the semantics of reused vocabularies (such as DCMES), its limited

<sup>62</sup>Part of that information would also be available from the object XML files, but on the object level, statement types are not represented in RDF. Moreover, representing the item type as an instance or class would enable a better querying/reasoning behavior, as, e.g., using OWL, it would be possible to restrict the set of item types.



documentation, and its use of certain bad RDFS modeling practices. The following paragraphs mention examples for each:

The most serious problem is that, apparently, one cannot express the direction of a *mowgli:Relation* between two mathematical knowledge items. MoWGLI reifies relations in a way inspired by HELM’s occurrences. A relation has a *mowgli:kind*, as, for example, *mowgli:example\_for*, and *dc:relation* is given the domain *mowgli:Relation* and range *xsd:anyURI* (why not *rdfs:Resource*?) for setting arbitrary resources into relation with each other. There is a notion of a direction of a relation (*mowgli:relDirection*) – *mowgli:straight* or *mowgli:reverse* –, but that does not help: In the absence of order in RDF, there is no way of saying which related resource comes first and should therefore be the source of the relation. Secondly, MoWGLI classifies metadata properties, its own as well as the ones reused from other ontologies, by their purpose (cf. [section 2.1.7](#)). For each category of metadata, there is a class (e.g. *mowgli:Lifecycle*), and that class is then declared as the *domain* of the respective metadata properties (e.g. *dc:date*). But, of course, one does not intend to express the date of a lifecycle, but the date of, e.g., a mathematical document. Thirdly, and slightly more subtly, properties that are intended to be used with the object of some *other* property *p* are erroneously declared a subproperty of *p*. For example, *mowgli:role*, which specifies the role that a person took when acting as the *dc:contributor* to a resource<sup>63</sup>, is declared a subproperty of *mowgli:role* itself.

The DCMES vocabulary, of which an RDFS implementation exists, is extended by a number of subproperties that do not match with the semantics intended by the DCMES specification. For example, *mowgli:example\_for* is intended to be a *mowgli:kind* of a *mowgli:Relation*, but *mowgli:kind* itself is declared a subproperty of *dc:relation*, from which an RDFS reasoner would infer `<#R> dc:relation mowgli:lemma_for for some mowgli:Relation #R`, which does not make sense. In the case of some sub-vocabularies, such as LOM (cf. [section 2.1.7.7](#)), the MoWGLI metadata documentation refers to their exhaustive documentations. In other cases, however, it reuses underspecified vocabulary, such as OMDoc’s underspecified *dc:date/@action* (cf. [section 2.4.4.1](#)), without further clarifications. Further bad RDFS practices in the ontology include the declaration of subclasses of *rdfs:Literal* that have their own properties, as well as populating certain XML namespaces (e.g. OMDoc’s) with RDF resources.

It is not clear whether this compound ontology has ever been used. Except for its specification, no trace in the form of annotated documents is left.

**Standoff Markup in PML:** PML (Proof Markup Language), an “*interlingua for sharing explanations generated by various automated systems such as hybrid web-based question answering systems, text analytics, theorem proving, task processing, web services execution, rule engines, and machine learning components*” [MDS+07], has been implemented as an OWL ontology consisting of modules for provenance, information manipulation or justification, and trust. PML assumes that facts and proofs have been written in some other language and merely adds standoff markup. Resources annotated that way can be referenced by URI or, in the case of text-based languages such as KIF, by byte offset. The justification module supports unproven conclusions or goals, assumptions, direct assertions, and antecedent→consequent justifications backed by inference rules. The provenance

<sup>63</sup>This is actually not that trivial to model, as explained in [section 5.2.4](#), as the role does not only depend on the person, but on the person *and* on the resource to which that person contributed.

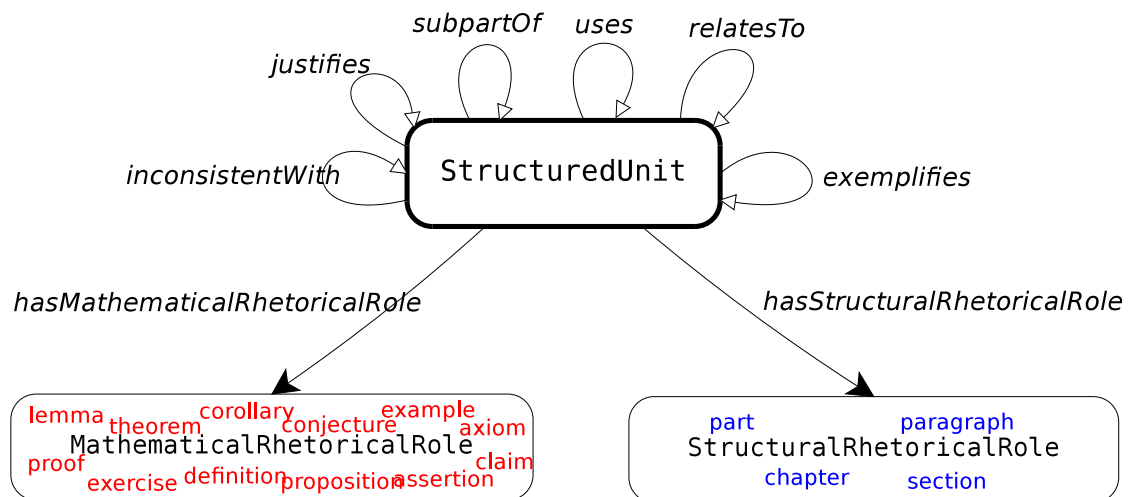


Figure 2.7: Part of the MathLang DRa ontology [Ret09, chapter 4.2]

module has a vocabulary for describing inference rules – again, not down to the object level. Thus, PML has a similar expressivity as OMDoc’s proof module (PF [Koh06b, chapter 17]), as far as the statement level is concerned. Finally, the trust module allows for expressing degrees of belief in informations and trust in agents.

**Standoff Markup in MathLang (DRa):** The Document Rhetorical aspect (DRa) of MathLang, covering statement-level logical structures and document structures (cf. section 2.4.6), has been implemented as an OWL ontology, whose main classes and properties are shown in figure 2.7 [Ret09]. Any text chunk of interest in the document being annotated is represented as an instance of *dra:StructuredUnit* and can have a “mathematical rhetorical role” – a statement type in our terminology – and a “structural rhetorical role” – a document section type. The instances of *dra:MathematicalRhetoricalRole* and *dra:StructuralRhetoricalRole* are limited to those shown in the figure, but with the *dra:hasOtherMathematicalRhetoricalRole* and *dra:hasOtherStructuralRhetoricalRole*, there is the possibility to assign custom roles to a structural unit. However, these are string-valued datatype properties, which prevents an author from describing a custom role in further detail. The possible relations that can hold between structural units are modeled as subproperties of a generic property *dra:specifies*, which is treated as a dependency relation (cf. section 2.1.6).

#### 2.4.10.3 Embedded RDFa Metadata

Directly embedding RDF metadata into XML data is a flexible compromise between the two approaches reviewed in the previous sections. RDFa, traditionally used to explicate the semantics only of certain salient parts of an XHTML document, also allows for a full formalization, intertwined with presentation markup or the semantic markup of a non-XHTML host language. With XHTML as a host language, the mathematical structures of interest would obviously have to be fully modeled in RDF(a), possibly using a combination of the ontologies reviewed above. On

the object level Presentation MathML would be a natural host language. With RDFa annotations using, e.g. ROBBINS's Content MathML ontology, one would obtain the expressivity of Content MathML and could thus, theoretically, replace it altogether – however, at the expense of RDFa being harder to read and write manually and no longer being directly compatible to OpenMath, which is why embedding embedding RDFa is not currently on the agenda for MathML.

RDFa metadata embedded into a semantic markup language can focus on capturing those aspects of the mathematical knowledge that the host language cannot express itself. “*RDFa alleviates the pressure on markup language designers to anticipate all the structural requirements users of their language might have*” [ABM+11]. For example, in the case of DocBook (cf. [section 2.4.8.6](#)), which already supports (Content) MathML objects, a vocabulary as rich in mathematical metadata as, e.g., MoWGLI, would give a good coverage.

Embedding RDFa into XML once more raises the question of how to represent order. RDFa authors can choose whether (i) they want to leave that to XML, as discussed in [section 2.3.3.5](#), or whether (ii) they additionally want to represent order explicitly in RDF. The latter requires extra work, particularly when employing RDF collections, but has the advantages of making information about order immediately accessible to RDF-aware tools – as far as they can handle it; see the comment in [section 2.3.3.3](#).

## 2.5 Designing an Improved Representation and Exchange Language

The requirements for reusably representing and exchanging mathematical knowledge established in [section 2.2](#) can be satisfied by combining two state-of-the-art representations. [Section 2.5.1](#) assesses all languages reviewed in [section 2.4](#) against the requirements. [Section 2.5.2](#) explains why a combination of OMDoc and RDF is the most suitable foundation for improvements and outlines how I have combined them, preparing the reader for chapters 3 to 5.

### 2.5.1 Assessment of the State-of-The-art Languages

[Table 2.3](#) summarizes how well the state-of-the-art languages for representing and exchanging mathematical knowledge satisfy the requirements established in [section 2.2](#). A detailed assessment follows:

**MathML 3** only covers mathematical objects. It has native support for representing functional/-logical structures as content markup, and for presentation markup, but not for specifying the content→presentation translation. It satisfies requirement F by combining presentation and content markup into interlinked parallel markup. Even presentation markup allows for some structural annotations, e.g. *mrow*. The content markup can be as formal[ized] as permitted by the CDs that are used – see the discussion for OpenMath below. MathML neither prescribes a particular set of CDs nor a particular CD language for defining new symbols; however, its built-in vocabulary is borrowed from the rather informal official OpenMath 2 CDs. The annotation mechanism is sufficiently flexible to embed non-mathematical information and supports typed links to external resources (e.g. `<annotation definitionURL="link-type" src="link-target"/>`), which results in an expressivity comparable to RDF(a). Incoming

Table 2.3: How the languages reviewed satisfy the knowledge representation requirements from figure 2.4<sup>a</sup>

► satisfies Requirement <sup>a</sup> ►	Structure Coverage						Formality		Linking			Comprh.		Cpt. E	
	S.L.*		S.R	S.N	S.M	S.D	F.R	F.C	L.A	L.→	L.←	C.S	C.H		
	T														
	O	S													
MathML 3	++	-	-	-	-	-	++	++	+	+	-	-	+	○	
OpenMath 2 Objects	++	-	-	-	-	-	+	○	○	-	+	-	○	-	
OpenMath 2 CDS	++ <sup>b</sup>	○	○	-	○	-	○	○	-	-	-	-	○	-	
OMDoc 1.2/\$fEX	++ <sup>b</sup>	++	+	++	+	○	-	++	+	○	+	-	-	-/○	
MathLang	++	++	-	-	-	-	++	+	-	-	○	○	-	-	
DocBook 5	++ <sup>b</sup>	-	-	-	-	-	-	-	-	+	+	-	-	-	
TEI P5	++ <sup>c</sup>	-	-	-	-	-	++	++	+	-	+	-	-	-	
DITA 1.1	++ <sup>c</sup>	-	-	-	-	-	-	-	+	+	+	+	-	○	
EPUB 2.0.1/DTBook 3	++ <sup>c</sup>	-	-	-	-	-	-	-	-	-	+	-	-	-	
CNXML o.7/ColXML/mdml	++ <sup>b</sup>	+	-	-	-	○	-	-	-	-	+	-	-	-	
Formalized languages	++	++	+/++	-	+	-	○	○	-	-	-	+	-	-	
RDF(a) 1.1	(depends on vocabulary, see table 2.4)														○
OMDoc 1.3/1.6	++ <sup>b</sup>	++	++	++	++	++ <sup>e</sup>	-	++	+	++ <sup>e</sup>	++ <sup>e</sup>	++ <sup>e</sup>	○ <sup>e</sup>	+	○

<sup>a</sup> legend (summarizing figure 2.4): S.\* = coverage of knowledge structures (S.L.{O,S,T} = logical/functional structures: mathematical objects, statements, theories;

S.{R,N,M,D} = rigorous language or rhetorical structures, notation, metadata, discussions), F.\* = degrees of formality supported (F.R = range of degrees, F.C =

coexistence of different degrees), L.\* = ability to link to non-mathematical knowledge (L.A = annotation, L.→ = outgoing links, L.← = incoming links), C.{S,H} =

comprehensibility to services/humans, E = compatibility with existing languages and tools

Symbols: ++ requirement satisfied excellently (and setting an example for other languages), + very well (but leaving room for improvement), ○ sufficiently well for

basic modeling tasks, - insufficiently (includes that a feature is missing by design)

<sup>b</sup> built-in support for MathML/OpenMath objects<sup>c</sup> via MathML extension<sup>d</sup> Dublin Core and similar vocabularies built in, others available via non-RDfA extensions<sup>e</sup> contribution of this thesis: OMDoc with a specified translation to an ontology (chapter 3) and extended by RDfA (chapter 5)

links can point to any subexpression, as all of them can have an *@id* attribute. (This property is shared by almost all XML-based semantic markup languages thanks to their usage of XML ID [MVW05] or comparable capabilities and therefore not restated for the other languages below.) The comprehensibility of MathML representations to services depends on the degree of formality of the CDs used. MathML can be presented to humans in a comprehensible way, as many browsers can directly render Presentation MathML, which can still carry content markup annotations accessible to appropriate in-browser services.<sup>64</sup> MathML can be used wherever the host language officially embeds it, which is often the case, or where the host language permits vocabulary extension.

**OpenMath 2 Objects** have similar properties as MathML objects. OpenMath does not support presentation markup itself but relies on the well-tested combination with Presentation MathML; therefore, in practice, it is comparable to MathML w.r.t. the **F** and **C.H** criteria. Integrating objects with external resources is harder than in MathML, as OpenMath does not support URIs. If the URI of an external resource fits into the *cdbase/cd#name* format it could debatably be treated as an OpenMath symbol; if not, one might declare as a symbol in some CD a non-standard function that constructs a URI from a string<sup>65</sup>. Machine-comprehensibility of OpenMath objects is achieved by a hard-coded phrasebook that translates them into the native language of some system. In that sense, the qualities of OpenMath as an object-level markup language are somewhat independent from a particular choice of CD representation language. However, semantically stronger CDs enable an easier creation of phrasebooks; a fully formalized CD could even act as a phrasebook itself. MICHAEL KOHLHASE and FLORIAN RABE have proposed an algebraic semantics of OpenMath objects that (weakly) defines their meaning independently from CDs [KR09]; however, for full machine-comprehensibility, CDs have to be taken into account additionally (see below).

**OpenMath 2 CDs:** With mathematical properties, examples, and type signatures, OpenMath's "reference encoding" for CDs partly covers statement-level logical/functional structures. An OpenMath CD shares with a theory the property of grouping related symbol declarations. Mathematical properties of symbols can be represented formally as *FMPs* containing OpenMath objects – albeit without distinguishing defined from asserted properties – or completely informally in plain text, without cross-links between both representations. Non-mathematical annotations and metadata, as well as links to external resources cannot be embedded into OpenMath CDs at all. Links pointing into OpenMath CDs are restricted in that only whole CDs and symbols have URIs, whereas mathematical properties don't. Traditionally, an OpenMath CD has been considered a set of instructions for a human programmer on how to implement a phrasebook, but not necessarily as something that is machine-comprehensible in itself. KOHLHASE's and RABE's proposed formal semantics for OpenMath includes a model-theoretic semantics for CDs, which, however, only covers *FMPs* [KR09].

<sup>64</sup>Section 6.4.2 and chapter 7 addresses the separate question of whether authors and services already take advantage of this expressivity of MathML.

<sup>65</sup>Such a constructor would be used as `@(www1#uri,"http://...")`.

**OMDoc 1.2** and its semantically mostly equivalent **TEX** input syntax inherit the functionality of MathML and of OpenMath objects. Other than that, they support expressive theory-level structures, such as theory graphs. OMDoc theories are supersets of OpenMath CDs, as noted in [section 2.4.3](#); thus, OMDoc serves as an alternative, more expressive CD language. Besides logical/functional structures, OMDoc supports rhetorical structures, document structures, and notation definitions. OMDoc supports a wide range of degrees of formality and extends parallel markup to mathematical text by fine-grained links between text and mathematical objects. Resources other than mathematical objects and mathematical text can be embedded into documents, but embedding and interlinking with mathematical knowledge does not work at arbitrary levels of granularity. Other than presentation-oriented hyperlinks, links to external resources are only supported within the limits of the DCMES vocabulary and by way of bibliographic citations. Knowledge represented in OMDoc 1.2 can be translated to a number of formalized languages via hard-coded phrasebooks; other than that, no formal, machine-comprehensible semantics has been specified.<sup>66</sup>

**MathLang** compares to OMDoc in its expressivity for formal and informal logical/functional structures and document structures, except that there are no theory level and no rhetorical structures. MathLang documents cannot link to non-mathematical knowledge. While MathLang’s native representation is only understood by MathLang-specific tools, there is a translation to XML. In contrast to the other aspects of MathLang, the DRa graph of a document, which partially covers statement-level logical structures and document structures, can be translated to an RDF graph, whose semantics is backed by an OWL ontology.<sup>67</sup>

**DocBook 5** focuses on document structures, structures pertinent to its main application area of software documentation, and a fixed set of metadata. It hardly has native markup for representing different degrees of formality and interlinking such representations; however, appropriate external linking vocabularies could be used with DocBook’s XLinks. Embedding arbitrary literal-valued metadata into a document is not supported. DocBook does not have a machine-comprehensible semantics, and the tools available for DocBook do not support generating semantically annotated human-comprehensible documents. This holds similarly for all following semantic markup languages and is thus not restated below.

**TEI P5** with its focus on literature is obviously not suited for *mathematical* knowledge but listed here nevertheless as a prime example of an expressive semantic markup language. In its own domain of literature, it can express knowledge in a wide range of degrees of formality. It supports fine-grained interlinking of different representations of the same knowledge, for example when annotating primary sources, such as scanned facsimiles [[BB09](#), chapter 11], preparing scholarly editions of a text (chapter 12), aligning multilingual texts (chapter 16). Arbitrary additional information can be embedded into a document, or provided as standoff

---

<sup>66</sup>This will change at least for the logical/functional core of OMDoc 1.6, whose formal semantics has already been specified in terms of the logical meta-language MMT, which allows for modeling theories within a logical framework.

<sup>67</sup>That makes MathLang perform better than OMDoc in this category; however, for reasons explained below, I chose OMDoc as the basis for my work. The state of OMDoc 1.2 shown in this table is the one before my extensions, which were made roughly at the same time as the introduction of the DRa into MathLang and, by “convergent evolution”, led to a comparable result; cf. [[Ret09](#)] and the discussion in [section 6.3.4](#).



markup pointing into the original document, whereas linking to external resources is restricted in that no link types are supported. Certain sub-vocabularies of TEI have been given a formal semantics by mapping them to relevant domain ontologies.

**DITA 1.1** performs as badly as DocBook w.r.t. the **S** and **F** requirements, except that topics can be interlinked. However, DITA has stronger support for adding arbitrary metadata and links. Due to its design focus on specialization, it may be integrated into existing workflows more easily than, e.g., DocBook.

**EPUB 2.0.1**, which was deemed an interesting candidate due the official MathML support of its recommended document format **DTBook 3**, clearly fails to satisfy all further requirements.

**CNXML 0.7**, combined with **CollXML** and **mdml**, supports, compared to EPUB, not just MathML objects but also some statement-level structures and informal dependencies between modules. Other than that, it is not more suitable for our purposes than EPUB.

**Formalized languages** are of interest here insofar as they also support informal content. This is usually not the case, the literate programming approaches mentioned in [section 2.4.9](#) being an exception. Languages for formalized mathematics usually do not support links out of or into formalizations. Each language comes with its own set of services that understand formalizations in the respective language, which have a strong model- or proof-theoretic semantics for logical and functional structures. These languages are usually committed to a particular logical foundation and therefore hard to translate into other languages. Existing translations have usually been hard-coded for a pair of two specific languages or logics (cf. [[Rabo8](#), chapter 1.1.3.3] and the Hets system [[Mos](#); [MMLo7](#)]).

**RDF's** suitability for representing mathematical knowledge stands and falls by the availability of vocabularies, i.e. ontologies, as [table 2.4](#) shows. Discussions, metadata, and various application domains are covered well by existing ontologies, a few ontologies for rhetorical and document structures also exist<sup>68</sup>, but ontologies with a good coverage of the logical/functional core of mathematical knowledge have been scarce so far (cf. [section 2.4.10](#)). RDF – without N3 extensions – is not a suitable model for fully representing mathematical objects, as argued in [section 2.4.10](#); on that level, I merely consider it as a *complement* to an XML representation language, following the MONET and HELM approach. Thus, RDF outlines coexisting with a complete XML representation and **RDFa 1.1**, embedded into a suitable XML host language, remains as a viable option. The degrees of formality that RDF supports depend on the ontologies that are used and on the complexity of the respective ontology languages. Where adequate ontologies are available, interlinked informal and formal representations can coexist in one RDF graph. Linking from and to RDF resources, and thus attaching arbitrary additional information, is supported excellently due to RDF's strong roots in URIs and its graph data model (cf. [section 2.3.3.3](#)). RDF, when published in compliance with the linked data principles (cf. [section 2.3.1](#)), is always machine-comprehensible in that a machine can simply retrieve information about resources by dereferencing URIs. However,

<sup>68</sup>As they have not been used in mathematical scenarios so far, I have not reviewed them in this chapter but cover them in [section 3.3](#).

Table 2.4: Structural coverage of RDF vocabularies/ontologies

Structures	Logical/functional			Rhet.	Notation	Metadata	Discussion
	Objects	Stmts.	Theories				
N <sub>3</sub> Vocabularies	+	○	–	–	–	–	–
OpenMath CD	○	○	○	–	–	○	–
HELM	+	+	○	–	–	–	–
MoWGLI	+	++	○	–	–	+	–
MathLang DRa	–	+	–	–	–	–	–
PML	–	++ <sup>a</sup>	–	–	–	–	–
SALT	–	–	–	++	–	–	+
OntoReST	–	–	–	++	–	–	–
DILIGENT	–	–	–	–	–	–	+
DCMI Terms	–	–	–	–	–	++	–
<b>OMDoc<sup>b</sup></b>	○	++	+	– <sup>c</sup>	+	– <sup>d</sup>	–
<b>OpenMath CD<sup>e</sup></b>	○	○	○	–	+	○	–
<b>SIOC Argum.<sup>f</sup></b>	–	–	–	–	–	–	++

<sup>a</sup> proofs only<sup>b</sup> contribution of this thesis, presented in [section 3.2.2](#)<sup>c</sup> intentionally delegated to SALT<sup>d</sup> intentionally delegated to DCMI Terms, ccREL, the OpenMath CD ontology, and other vocabularies<sup>e</sup> contribution of this thesis: a modernized ontology presented in [section 3.2.3](#), which I have developed for the purpose of maintaining OpenMath CDs<sup>f</sup> contribution of this thesis, presented in [section 3.6](#)

the expressivity of the ontologies used determines the extent to which the mathematical semantics of an RDF representation is actually machine-comprehensible.<sup>69</sup> By embedding RDFa annotations into XHTML, human-comprehensible documents can retain as much semantics as desired. As RDFa can also be embedded into semantic markup languages for extending their metadata vocabulary, users can continue to use these languages but still benefit from RDF's linking capabilities.

## 2.5.2 Towards an RDF-extended OMDoc as an Exchange Language

The improved exchange language presented in this thesis is based on OMDoc, combined with RDF. [Table 2.3](#) shows at first glance that no single language satisfies all requirements for reusably representing and exchanging mathematical knowledge, but that OMDoc (including MathML or OpenMath objects) and RDF(a) are complementary in their coverage of the requirements.

The existing markup languages for mathematics, headed by OMDoc, lead the way w.r.t. coverage of mathematical structures and combining formal and informal representations. Moreover, these

<sup>69</sup>On consequences of the lack of expressive ontologies for linked data applications, compare [section 6.4.1.1](#) and [\[JHY+10\]](#).



existing languages are reasonably well accepted by the MKM community, as opposed to, e.g., RDF. In terms of markup capabilities, TEI is also competitive, were it not for its focus on literature. OMDoc supports both MathML and OpenMath for representing mathematical objects. MathML performs much better than OpenMath at a first glance, but in many practical cases the parallel content and presentation markup accounting for MathML's advantage w.r.t. the **F** and **C.H** criteria can be generated automatically from content markup, be it Content MathML or OpenMath.

While RDF does not perform well in those categories where OMDoc excels, extending OMDoc by RDF will compensate for OMDoc's deficiencies in the other categories. I have pursued this extension in three complementary tracks: translating OMDoc to RDF, using OMDoc as an expressive ontology language for RDF vocabularies, and embedding RDFa into OMDoc.

**Translating OMDoc to RDF:** Concerning requirement **C.S**, only OMDoc's representation of the logical/functional structures of mathematical knowledge is currently comprehensible to services – a small number of OMDoc-specific services –, whereas for the other structural dimensions not even a formal semantics has been specified. The full conceptual model behind OMDoc has so far only been implemented in the form of an XML schema, which is not suited for semantically annotating human-comprehensible documents (e.g. XHTML), as required by **C.H**. Both requirements have been addressed by formalizing and implementing OMDoc's conceptual model as an *ontology* and specifying a translation from the OMDoc XML schema to that ontology, i.e., on instance level, a translation from OMDoc XML documents to RDF graphs<sup>70</sup> using that ontology. The availability of an OMDoc ontology both makes OMDoc's conceptual model available to all services that understand RDF and ontologies<sup>71</sup>, and it provides a vocabulary for annotating mathematical structures in human-comprehensible documents using XHTML+RDFa. Note the twofold potential of the latter: (i) Documents originally authored in OMDoc can be published on the Web as XHTML+RDFa, preserving the original semantics, and (ii) existing XHTML+RDFa editors can be reused for authoring documents that are semantically equivalent to OMDoc documents (cf. requirement **E**).

**Writing Expressive RDF Vocabularies in OMDoc:** An adequate OMDoc ontology has to satisfy the following requirements:

**Complexity/Expressivity:** OMDoc's expressivity and its wide coverage of the structural dimensions of mathematical knowledge require a complex ontology. This does, however, not preclude the provision of a simplified subset for scalable reasoning.

**Documentation:** Due to the incompleteness and ambiguity of OMDoc's XML schema and specification w.r.t. certain aspects of informal markup, the ontology will need a comprehensive and comprehensible documentation for developers and users, as, in such cases, a simple reference from an ontology concept to a section of the OMDoc specification will not suffice to clarify how to use that concept.

<sup>70</sup> Alluding to the role of ontologies as providing semantics for the XML syntax, such an XML→RDF translation is sometimes called “lifting”; the opposite being “lowering”. This terminology is particularly common in the field of web services [AKK+08]. This thesis generally speaks of “translation”, or of “extraction”, in cases where the RDF graph obtained from an XML document only represents a structural outline of the knowledge.

<sup>71</sup> A service that only understands RDF graphs but not the entailments given by the OMDoc ontology will, however, merely be able to understand and utilize a subset of the mathematical structures represented that way.

**Integration/Reuse:** Certain dimensions of mathematical knowledge, such as rhetorical structures, document structures, administrative metadata, and application-specific information, are already covered by existing ontologies, which should be reused – literally, or by adapting them and integrating them with the OMDoc ontology.

All of these requirements are well satisfied by the OMDoc language, which is expressive and supports literate programming and modularity. Due to its independence from a particular logical foundation, ontologies in any desired logic can be expressed in OMDoc. Treating ontologies as a specific kind of mathematical knowledge and representing them in OMDoc (including MathML/OpenMath for axioms/rules and RDFa for metadata) once more satisfies all requirements from [section 2.2](#) from an ontology engineering perspective – including requirement E, in that tools for managing mathematical knowledge can now also be applied to ontologies.

**Embedding RDFa into OMDoc** extends the expressivity of OMDoc beyond the primary formalization intent that OMDoc supports by its native syntax. Wherever authors wish to maintain information from additional structural dimensions in the same document as the primary mathematical knowledge, RDFa enables them to embed such annotations, metadata, and links into OMDoc. The extensibility of OMDoc’s vocabulary enabled by integrating RDFa addresses requirements **L.A** and **L.→**, whereas the additional dimensions of knowledge that can now be expressed inside OMDoc documents improve OMDoc’s utility w.r.t. requirement **S**.

By a similar argument, the availability of the OMDoc ontology also enables a new dimension of knowledge – i.e. the mathematical one – to be expressed in languages whose primary formalization intent is not mathematics, such as DocBook or DITA, without requiring extensions to their XML schema. The only prerequisite is that they support RDFa or a similar extension mechanism for metadata (cf. [section 2.4.8.6](#)). This addresses requirement E once more, as authors can continue using existing tools for these languages.

Each of these extension tracks enables more coherent information processing. Where, previously, mathematical knowledge had to be represented in OMDoc/XML and processed, e.g., in XSLT or XQuery, and other dimensions of knowledge had to be represented, e.g., as external RDF graphs and queried in SPARQL, *all* dimensions of knowledge can now be uniformly represented in the same language – either XML or RDF, as appropriate, – and thus be processed cohesively.<sup>72</sup> Secondly, representing ontologies in OMDoc allows for integrating their formalization with their documentation and thus managing and utilizing both in combination. Note that the commitment to OMDoc/XML and RDF does not conceptually restrict the generality of the approach, as tools for translating other, e.g. non-XML languages, to OMDoc and RDF exist.

The following chapters are structured as follows: [Chapter 3](#) introduces the ontologies that formalize the semantics of structures of mathematical knowledge in an RDF-compatible way and specify the XML→RDF translation. OMDoc is used as a heterogeneous language for implementing

---

<sup>72</sup>In practice, it makes sense to have both representations, e.g. by maintaining XML and automatically translating it to RDF. [Chapter 6](#) shows that some services – be it as a matter of principle or due to a better availability of implementations – work better on an XML representation, whereas others work better on RDF; [section 6.7](#) discusses that once more.

these ontologies in their full expressivity and for enhancing their documentation (cf. [chapter 4](#)). A variant of our XML→RDF extraction allows for extracting the OWL-compatible subset from the OMDoc implementation of an ontology. The embedding of RDFa into OMDoc, extending OMDoc’s coverage to more than the built-in logical/functional, document, and rhetorical structures, is introduced in [chapter 5](#).

## Acknowledgments

The investigation of structures of mathematical and application-specific knowledge mentioned in sections [2.1.1.4](#), [2.1.7.6](#) and [2.1.7.7](#), from which we developed the notion of multiple dimensions of formality, has been made in collaboration with ANDREA KOHLHASE and MICHAEL KOHLHASE [[KKL10a](#)]. The elaboration on logical structures of mathematical knowledge in [section 2.1.2](#) is heavily based on prior work by MICHAEL KOHLHASE [[Koh06b](#); [KK08](#)] or in collaboration with him [[LK09](#)], but adapted to the topic of this thesis. Part of the review of argumentation ontologies has been made in collaboration with ULDIS BOJĀRS, TUDOR GROZA, JOHN BRESLIN, SIEGFRIED HANDSCHUH, TUUKKA HASTRUP, and STÉPHANE CORLOSQUET [[LBG+08](#); [LHCo8](#)], with further advice from THOMAS SCHANDL, CHRISTOPH TEMPICH, MAX VÖLKE, and STEFAN DECKER. The brief review of state-of-the-art ontology languages and methodologies in [section 2.3.4](#) is partly based on joint work with JOHN BATEMAN. The work on defining notations for mathematical symbols, as reported in [section 2.4.5](#), has been done in collaboration with MICHAEL KOHLHASE, CHRISTINE MÜLLER, NORMEN MÜLLER, and FLORIAN RABE [[KLR07](#); [KLM+09](#)].



# Ontologies for Structures of Mathematical Knowledge

*OWL is a small subset of first-order logic stated in a different notation.  
First-order logic is a small subset of English stated in a different notation.  
If you can't state clearly in English exactly what you want to do,  
none of those other notations will help you.*

—JOHN F. SOWA [Sow10]

The previous chapter has reviewed the structural dimensions of mathematical knowledge – logical/functional structures, rhetorical structures, document structures, presentational information, metadata, application- and environment-related information, and discussions about knowledge items – and existing languages for representing them. The goal of this chapter is to make mathematical knowledge represented in expressive markup languages such as OMDoc comprehensible to a wider range of services by capturing the conceptual models of these languages in ontologies. This chapter introduces the vocabulary and semantics of these ontologies, roughly separated by the structural dimensions identified before ([section 3.1](#)), and then discusses how to translate knowledge represented in markup languages into an ontology-based representation – concretely: an RDF graph –, so that ontology-aware services can understand it (cf. [section 3.7](#)).

## 3.1 Overview of the Ontologies by Structural Dimension

The main ontology for logical/functional structures ([section 3.2](#)) has been newly derived from the conceptual model behind the OMDoc language, a closely related one from the OpenMath CD language. This chapter also subsumes the representation of notation under logical/functional structures, for the following reason: Knowledge about how to present mathematics is most frequently represented in an explicit semiformal way when it concerns the notation of symbols. The definition of a notation for a symbol is closely related to the declaration of that symbol; often, both are maintained in the same place. As the ontology covered in [section 3.2](#) is the only one that has

been developed from scratch, instead of reusing existing ontologies, that section initially treats of methodological issues.

Rhetorical structures and document structures are often modeled in combination and have already been implemented by ontologies that we can reuse. These ontologies and their integration with the ontology for logical/functional structures are discussed in [section 3.3](#). Similarly, most metadata vocabularies are available as reusable ontologies ([section 3.4](#)). The same holds for many application domains, such as science, and for the environment in which mathematical knowledge is maintained, e.g. social networks ([section 3.5](#)). I have also been able to reuse an argumentation ontology but had to add support for discussing problems related to mathematical knowledge items ([section 3.6](#)).

## 3.2 Logical and Functional Structures, and Notation

None of the ontologies for logical/functional structures of mathematical knowledge that have been reviewed in [section 2.4.10](#) is immediately reusable for our purposes. This is due to a limited coverage in most cases (N3 vocabularies, OpenMath CD ontology, MONET, HELM, MathLang DRa), or to shortcomings in their conceptualization (such as HELM not being sufficiently abstract) or formalization (such as MoWGLI exposing blatant technical flaws, or MathLang DRa not being sufficiently extensible). Thus, I have developed a new ontology based on the conceptual model of OMDoc, as OMDoc has an excellent coverage of the logical/functional structures, both in informal and in formal representations. While OMDoc's theories subsume abstract OpenMath CDs, the OpenMath CD language is still widely in use. Thus, I have derived a simple alternative ontology directly from the OpenMath CD specification, to allow for making the knowledge contained in OpenMath CDs accessible to services in a more straightforward way, but also discuss how that ontology could be aligned with the OMDoc ontology.

The two new ontologies borrow certain modeling approaches from the aforementioned ones. Aligning the new ones with the existing ones would be conceivable as a further step, but it is not urgent, as most of the latter are virtually extinct, and the remaining ones are apparently not widely in use.<sup>1</sup>

### 3.2.1 Methodology

For developing an ontology, we have to consider the steps of requirements specification, conceptualization, formalization, and implementation (cf. [section 2.3.4](#)). While the requirements have largely been assessed, a lot of conceptualization and formalization remains to be done. OpenMath 2 is largely covered by an abstract information model, but OMDoc 1.2 is not. The existence of such a model facilitates the development of an ontology, as pointed in the context of a recent related effort, the development of an ontology for the Statistical Data and Metadata Exchange format: *“The task of mapping SDMX to RDF is greatly aided by the fact that the SDMX standard is separated into an*

---

<sup>1</sup>The index of Ping the Semantic Web lists one document annotated with terms from the <http://www.w3.org/2000/10/swap/math#> namespace of the N3 vocabularies, compared to, e.g., more than a million annotated with FOAF [Opea]. The MathLang source code, including the OWL implementation of the DRa ontology, is not freely accessible.

abstract information model (SDMX-IM) and concrete XML and UN/EDIFACT based syntaxes.” [CFG+10].

### 3.2.1.1 Requirements Assessment

High-level requirements have already been specified in [section 2.2](#). The roadmap from [section 2.5.2](#) entails the more concrete requirement that our new ontology should capture the conceptual and formal model behind all of OMDoc’s markup for logical/functional structures, excluding the full functional structure of mathematical objects, which will continue to be expressed in MathML/OpenMath.

### 3.2.1.2 Conceptualization

[Section 2.1.2](#) provides a high-level conceptualization of logical/functional structures. For most languages reviewed in [section 2.4](#), further, more detailed conceptualization is provided by their specifications. There are, however, remaining ambiguities and gaps in the specifications. Some of these open questions can be answered by studying an implementation of the specification, be it the XML schema or a tool or library that processes the respective representation. Thus, I had to recover part of the conceptualization by reverse-engineering the specification and available implementations and instance documents, or finally by asking the developers to clarify remaining issues.

For example, in the conceptual model of OMDoc there is the general notion of a relation between semantically equivalent representations of different degrees of formality but does not always make it explicit in its markup. When an informal text section `<omtext xml:id="t"/>` makes the same statement as a (semi)formal definition `<definition xml:id="d" for="symbol"/>`<sup>2</sup>, this can be expressed by pointing to the latter using `<omtext xml:id="t" verbalizes="#d"/>`. The content of a *definition* element does not have to be fully formal; once again, formal and informal representations can be mixed.

Listing 3.1: Implicit definition of the exponential function

```
<definition xml:id="exp-def" for="exp" type="implicit">
  <CMP>The exponential function equals its derivative and evaluates to 1 for
    an argument of 0.</CMP>
  <FMP> <!-- the same as an OpenMath or Content MathML object, -->
     $\exp' = \exp \wedge \exp(0) = 1$  <!-- here given in  $\LaTeX$  to save space -->
</FMP>
</definition>
```

Consider the definition given in [listing 3.1](#).<sup>3</sup> Here, the *CMP* and the *FMP* are in the same relation as the *omtext* and the *definition* in the example before. This relation is implicit in the markup, as there is no ambiguity in what the *CMP* verbalizes. From the specification, the fact that both relations are the same, or similar, is not completely obvious either. For various occurrences of *@verbalizes* attributes, the specification states that they refer to “formal representations”. However, for the

<sup>2</sup>Note the higher degree of formality: When using the *definition* element, one has to make explicit what symbol it defines by referencing that symbol by name.

<sup>3</sup>This is a modified version of [Koho6b, listing 15.4].

CMP–FMP relation, the specification merely states that “if an FMP group has CMP siblings, all must express the same content,” [Koho6b, chapter 14.2] and leaves the final step of understanding the conceptual model to the reader.

### 3.2.1.3 Formalization

In the next ontology engineering step, the formalization, the concepts as well as their properties and relations identified in the conceptualization stage are represented in a formal model, where they are “defined through axioms that restrict the possible interpretations for the meaning of those concepts” [PMo4]. Concerning “restrict[ing] the possible interpretations”, note that the intended semantics cannot always be fully represented in the logic chosen to formalize the axioms, as there is an inherent trade-off between expressive logics and logics for which relevant reasoning problems are decidable or even efficient to implement. I chose the *SR<sub>Q</sub>IQ* DL, which corresponds to the DL profile of the OWL 2 language<sup>4</sup>, for formalizing most of the semantics of logical structures of mathematical knowledge. However, not everything can be formalized in that logic. The following sections discuss detailed cases where a higher expressivity is required.

For the logical meta-language MMT, which will become the logical/functional core of OM-Doc 1.6, a formal semantics has been specified [Rabo8] – albeit not in a logic commonly used for RDF vocabularies – and implemented in the reasoning component of the MMT software library in a custom way that is, however, not too different from an OWL ontology [Raba]. The formal specification covers the full theory, statement, and object level of OMDoc, but without any semi-formal and informal representations so far, such as text sections, examples, or structured proofs. The ontology-like part of the implementation (the modules *jomdoc/ontology/{ABox,TBox}.scala*<sup>5</sup>), which has not yet been specified formally, covers the theory and statement level, part of the object level (occurrences of symbols in mathematical objects) and defines dependency w.r.t. well-formedness between concepts on the theory and statement level (cf. section 2.1.6). Further hints for formalizing the OMDoc ontology can again be found in the OMDoc 1.2 specification, but subject to similar difficulties as mentioned above for the conceptualization. The XML schema is of some help: For example, elements that occur in similar contexts probably have something in common, such as corresponding to classes that have a common superclass. The parent→child element relation can often be interpreted as a whole→part relation, whereas the exact type of whole→part relation is not clear from an XML schema. Most of the markup languages reviewed in section 2.4 have a RELAX NG schema; if they had an XSD schema that made use of element datatypes, one could possibly derive some more formal semantics from it. Relations other than a class hierarchy and some kind of whole→part relation cannot easily be derived from an XML schema. Except for document-internal referencing mechanisms (*ID/IDREF*), which some schema languages support, references to other objects are expressed using URIs, which a schema validator can only syntactically validate but not dereference.

---

<sup>4</sup>In the following descriptions of formalizations, I prefer the OWL terminology (e.g. “classes” and “properties”) over the DL terminology (e.g. “concepts” and “roles”).

<sup>5</sup>not to be confused with the JOMDoc library, which currently covers OMDoc 1.2 and 1.3 [Jom]



### 3.2.1.4 Implementation and Beyond

Finally, the ontology is implemented. As said above, OWL 2 DL is not sufficiently expressive for capturing all aspects of OMDoc. Another shortcoming of OWL, which becomes apparent later in this chapter, where the ontologies for the other structural dimensions of mathematical knowledge and their interactions are studied, is its weak support for modularity and reuse. For that reason, the implementation of the ontologies is covered separately in [chapter 4](#), which presents OMDoc in a different role – as an expressive language for logically heterogeneous and modular ontologies with comprehensive documentation facilities. For now, it suffices to say that I have used *SR<sub>OTQ</sub>* for formalization wherever possible, and that all of these axioms can be extracted from the heterogeneous OMDoc implementation into an OWL ontology for use with existing reasoners. The simpler OpenMath CD ontology, introduced in [section 3.2.3](#), has, however, directly been implemented in OWL.

Further stages of the ontology lifecycle, most importantly maintenance, did not yet apply to the short period of using the ontologies in the applications presented in [part III](#). As the ontologies are strongly based on existing languages with their own lifecycles, they will be subject to maintenance as soon as the underlying languages will undergo semantic revisions having an impact on the corresponding ontologies.

## 3.2.2 The OMDoc Ontology

The core of the OMDoc ontology is shown in [figure 3.1](#). A complete listing of classes and properties is given in [appendix B.1](#). Large parts of the vocabulary of the schema and the ontology are identical except for spelling conventions. Following the methodology described above, I consulted the RELAX NG XML schema of OMDoc as a source for conceptualization wherever possible.

### 3.2.2.1 Class Hierarchy

In a first step, I introduced concepts corresponding to most of the elements on the statement and theory levels. In the formalization phase, they were grouped into a class hierarchy, including appropriate disjointness axioms<sup>6</sup>. Abstract superclasses were added, following OMDoc's model of grouping related structural entities – for example, dividing statements into those that are constitutive for a theory and those that are not. Subclasses were added where elements permit further specialization by *@type* attributes, e.g. in the case of definitions and assertions.

After the classes, properties were added to the ontology (see below). In some cases, finding right domains and ranges for them suggested further refinements of the class hierarchy. Initially, for example, both declared and asserted types were represented by the same class *Type*, but then it turned out that only asserted types have to be justified by assertions (property *typeJustifiedBy*), which led to the introduction of a subclass *AssertedType* used as the domain of that property.

### 3.2.2.2 Representing Proven and Refuted Assertions

To faithfully model the subclasses of *Assertion*, property restrictions were imposed. Some types of OMDoc assertions have a proof (*Theorem*, *Proposition*, *Corollary*, *Lemma*), others do not need a

<sup>6</sup>Disjointness axioms are particularly important for ontology-based validation (cf. [section 6.3.3.1](#))

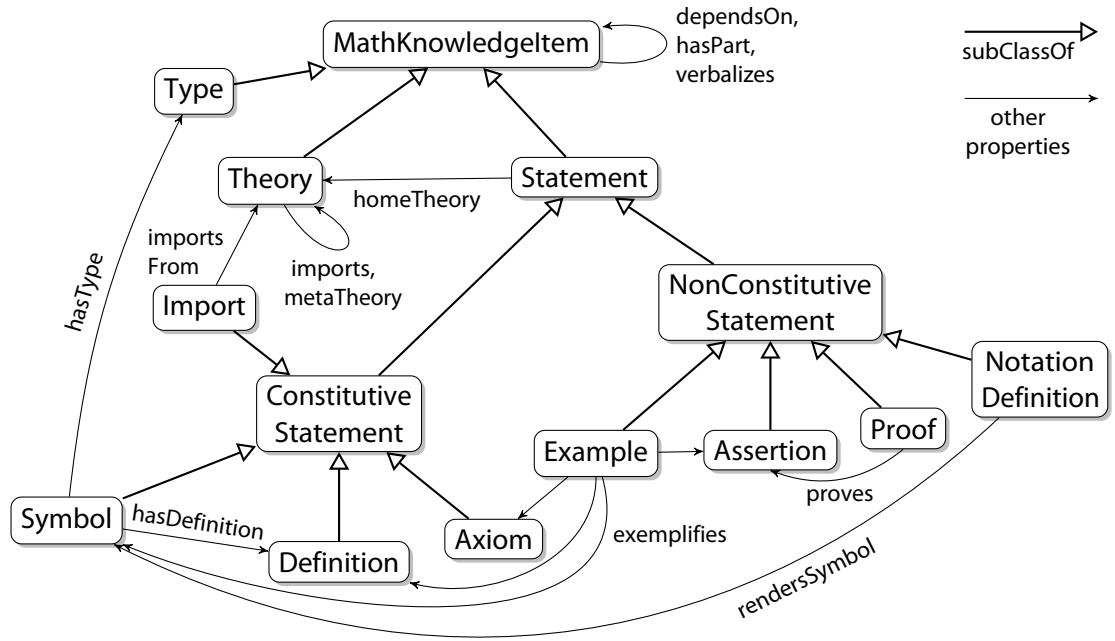


Figure 3.1: The core of the OMDoc ontology (slightly simplified)

proof (*Conjecture*, *Obligation*, *AssumptionAssertion*<sup>7</sup>, *Postulate*, *Rule*, *Formula*), and, finally, *FalseAssertion* has been refuted by a counter-example. This is reflected by the three subclasses *ProvenAssertion*, *UnProvenAssertion*, and *RefutedAssertion* of *Assertion*. Note that these sibling classes are not declared disjoint, as, in collaborative formalization, it is a frequently occurring situation that one author considers an assertion “true” and provides a “proof”, whereas another author refutes the same assertion by a counter-example – compare the remarks on IMRE LAKATOS’s “Proofs and Refutations” in [section 1.1](#). Now, we can state that proven assertions have a proof<sup>8</sup>, whereas refuted assertions have a counter-example<sup>9</sup>:

$$\text{ProvenAssertion} \sqsubseteq \exists \text{provedBy}. \text{Proof}$$

$$\text{RefutedAssertion} \sqsubseteq \exists \text{refutedBy}. \text{Example}$$

These axioms require the existence of a proof or a counter-example, but, due to the open world assumption of DL, that proof or counter-example neither has to exist in the current document

<sup>7</sup>This corresponds to *assertion[@type='assumption']*, but has to be distinguished from a different class *Assumption* corresponding to the *assumption* element.

<sup>8</sup>An existential restriction is sufficient, as we *provedBy* has a declared range of type *Proof*, i.e. all targets of a *provedBy* link are implied to be *Proofs*. Equivalently, we could have used a cardinality restriction such as  $\text{ProvenAssertion} \sqsubseteq \geq 1 \text{provedBy}$ .

<sup>9</sup>This thesis uses the so-called German DL notation, which is, e.g., defined in [HPSH03].

nor in our knowledge base.<sup>10</sup> In mathematical practice, this is most commonly the case with propositions, whose proof exists in the literature, or with lemmas corollaries, if their proof is trivial and therefore omitted. For a DL reasoner, the required proof or counter-example merely exists *somewhere*, and we can describe some of its properties. Note that, if it exists within reach, determining its validity and thus the truth or falsehood of the assertion is beyond the scope of an ontology that models knowledge *structures*; see [section 3.2.2.8](#) below for a discussion.

### 3.2.2.3 Representing Different Degrees of Formality

An approach different from subclassing was chosen to capture OMDoc’s support of flexible degrees of formality. Where OMDoc often distinguishes informal from formal content by different markup wrapped around the proper content, I decided to always represent the same concept by the same class, and to treat the informal/formal distinction as a secondary one. This is motivated by the high-level goal of making knowledge widely reusable and comprehensible. For a large number of applications, it does not make a difference whether, for example, a proof is formal or not. It does make a difference whether a valid proof of a theorem exists, regardless of who validated it – a human user or a machine. Now, let us more concretely see how OMDoc distinguished informal from formal content, and how the ontology reflects that. On the statement level, there is usually a formal *statement-type* element and an informal *omtext*[@type=‘statement-type’] element. Within statements, formal mathematical properties (*FMP*) are distinguished from informal ones (*CMP*), as in OpenMath CDs. On the object level, which this ontology only covers marginally, there can be formal content markup or informal presentation markup. All of these can be combined flexibly; consider, for example, a proof whose outer structure is given as a “formal” *proof* element, but whose steps are represented by “informal” *omtext* elements, which may again contain content or presentation markup objects.

The informal/formal distinction is modeled by the *formalityDegree* property, whose range is *FormalityDegree*, a class with the three different instances *Informal*, *Rigorous*, and *Computerized* so far, the latter of which is currently only used for completely formalized proof objects. Thus, an informal definition, written as `<omtext xml:id="d" type="definition">` in OMDoc, will be represented as follows in RDF:

```
<#d> a oo:Definition ;
      oo:formalityDegree oo:Informal .
```

Similarly, both *CMP* and *FMP* map to the same ontology class *Property*. Finally, to facilitate the axiomatization of classes with a fixed formality degree, such as *Symbol* – a symbol declaration –, which is always formal, the following classes have been defined:

$$\begin{aligned} \text{RigorousKnowledgeItem} &= \text{formalityDegree:Rigorous} \\ \text{InformalKnowledgeItem} &= \text{formalityDegree:Informal} \end{aligned}$$

<sup>10</sup>Possibilities for enforcing the existence have been investigated in research on ontological reasoning; [section 6.3.4](#) provides pointers to relevant works.

One aspect we cannot formalize in *SRQIQ* is that a statement may have at most one informal and one formal property. This is because “one property” may have equivalent variants in multiple natural languages or multiple logics. For example, a statement can have *CMPs* in English and German (multilingual group, cf. [Koho6b, chapter 14.1]), and *FMPs* in FOL and DL (multi-logic group, cf. [Koho6b, chapter 14.2]). If we do not model the multilingual/multi-logic *group* as an entity of its own – which would merely defer the problem – the case of multilingual *CMPs* might be formalized as follows<sup>11</sup> in FOL<sup>12</sup>:

$$\begin{aligned} &\forall s \in \text{Statement} \forall p_1 \in \text{Property} \forall p_2 \in \text{Property} \forall l_1 \in \text{xsd:language} \forall l_2 \in \text{xsd:language}. \\ &\langle s, p_1 \rangle \in \text{hasProperty} \wedge \langle s, p_2 \rangle \in \text{hasProperty} \wedge p_1 \neq p_2 \wedge \langle p_1, l_1 \rangle \in \text{dc:language} \wedge \langle p_2, l_2 \rangle \in \text{dc:language} \\ &\rightarrow l_1 \neq l_2 \end{aligned}$$

### 3.2.2.4 Properties – Parthood, Dependency, and Others

Almost all *properties* of the OMDoc ontology are, in terms of DL, object properties, i.e. relations between two resources. Datatype properties, i.e. literal-valued properties, almost exclusively occur in metadata vocabularies that have been incorporated into the OMDoc language, but not within the OMDoc namespace. Three generic, orthogonal properties form the top of the object property hierarchy: a whole→part relationship (*hasPart*), dependency (*dependsOn*), and a verbalizes/formalizes relationship between semantically equivalent mathematical knowledge items having different degrees of formalization (*verbalizes*). Whole→part and dependency relations are treated as orthogonal, because there is no reasonable way of fixing their direction of dependency: The whole depends on its parts, but the whole also provides context for its parts.

The *properties* of the ontology abstract from the XML schema to a greater extent than the classes. Most properties representing whole→part relationships are derived from parent-child containments in the XML tree, or they represent other relationships derived from attributes pointing to [the URIs of] other elements. Purely logical whole→part relationships exist between theories and their statements, and between statements and their substatements. Other whole→part relations, often but not necessarily coinciding with the logical ones, occur in the document structure and are covered in the following section. The primary interrelation on theory level is the import relation. On statement level, there is a great diversity of interrelations. In the XML syntax most of them are established by a single URI-valued attribute that always has the same name – *@for*. The intended semantics can partly be deduced from the types of statements linked that way, but I chose to make it more explicit by introducing separate properties for the *@for* attributes of separate elements, again relying on the informal descriptions of the respective elements in the OMDoc specification. Inverse properties have been added for most properties to allow for more convenient query formulation.

<sup>11</sup>This formalization assumes that only informal properties can have a language. This can be formalized in *SRQIQ* by declaring *Property*  $\sqcap$  *InformalKnowledgeItem* a domain of *dc:language* and taking advantage of the formality degrees *Informal* and *Rigorous* being different individuals. Furthermore, we assume that, in the RDF graph extracted from OMDoc, all properties of a statement have been declared different from each other. This is necessary when using OWL reasoners, as they do not make a unique name assumption.

<sup>12</sup>For convenience and coherence, I treat DL as a subset of FOL and continue to use the German DL notation where it does not interfere with FOL notation.

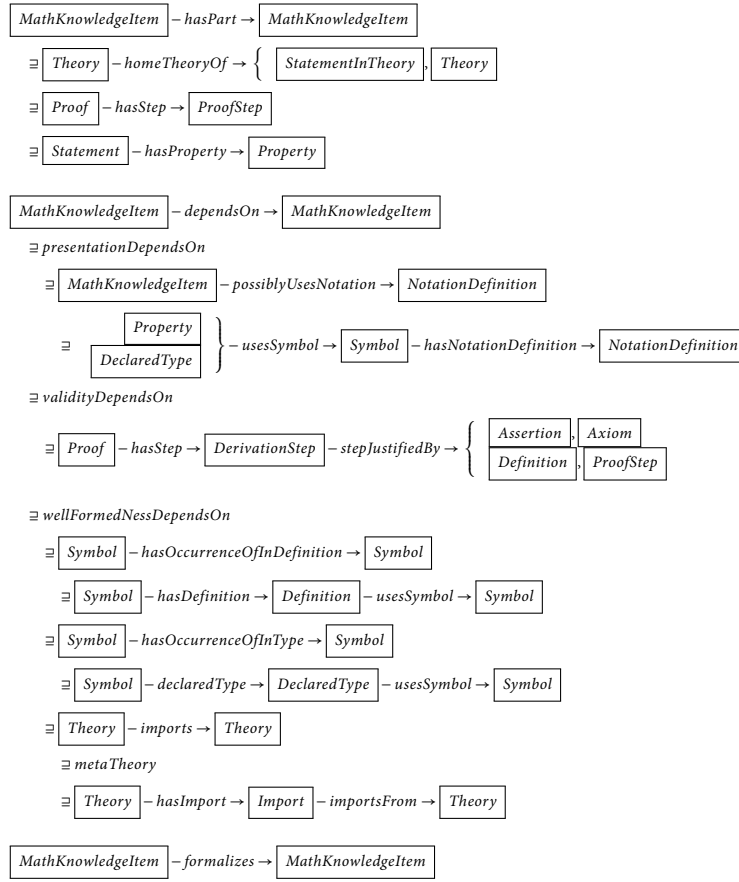


Figure 3.2: Main properties of the OMDoc ontology (subproperties indented; slightly simplified)

A hierarchy of properties has been established under *dependsOn* and *hasPart*, as shown in [figure 3.2](#). The latter generic property has another generic subproperty *hasDirectPart* to denote immediate parthood. This property finally has obvious subproperties that directly correspond to the XML schema, such as *homeTheoryOf* for theories containing statements, or *hasStep* for proofs consisting of steps. The main reason for modeling direct parthood separately is that the parthood given here is both transitive and inverse functional: (i) If a proof step is a part of a proof and that proof is a part of a theory, the proof step is an indirect part of the theory. (ii) A certain proof step can only be given once in a proof, i.e. the proof, of which the step is a part, is a function of the step. For decidability reasons, *SRIOQ* does not allow cardinality restrictions – of which [inverse] functionality is a special case – to be imposed on transitive properties (cf. [\[MPSP09, chapter 11.2\]](#)). Therefore, any cardinality restrictions that hold have to be declared for non-transitive subproperties such as *hasDirectPart*.

As outlined in [section 2.1.6](#), different applications have different notions of dependency. In the context of OMDoc, MMT’s notion of dependency w.r.t. well-formedness is understood best so

far. I made it a subproperty *wellFormedNessDependsOn* of *dependsOn* and simply reengineered those cases of dependency supported by the MMT library (cf. [section 3.2.1.3](#)) in *SRITQ*. Well-formedness dependency can be modeled by a simple subproperty in the case of theories importing theories. In other, more involved cases, dependency propagates along chains of multiple properties. Imports, for example, can be represented as simple theory→theory links in OMDoc, but they can also be represented as morphisms with symbol mappings. In the latter case, it makes sense to treat them as first-class objects on statement level. The importing theory  $t_1$  would then contain an import via the *hasImport* property, and the import would refer to the imported theory  $t_2$  via the *importsFrom* property. By the following property chain axiom (also known as role inclusion axiom), one can infer  $t_1 \text{ imports } t_2$  from that:  $\text{hasImport} \circ \text{importsFrom} \sqsubseteq \text{imports}$ .

Dependencies on statement level can be inferred similarly, as shown in [figure 3.2](#); I have modeled property chain axioms for a symbol  $s_1$  depending on another symbol  $s_2$  that occurs in the type or definition<sup>13</sup> of  $s_1$ .

As a proof of concept, another inference rule for a dependency relation that affects proofs has been formalized, stating that the validity of a proof depends on statements, which can be both external (definitions, axioms, previously proven assertions) or local (nested subproofs and their dependencies); cf. [figure 3.2](#).

The above-mentioned *usesSymbol* property, whose domain is a (mathematical) *Property*, is currently the only property that extends down into the level of mathematical objects. It flattens the functional structure of a formal property as well as the rhetoric structure of an informal property by treating all occurrences of symbols equally, regardless of the depth of the mathematical object or phrase tree in which they occur.

The only datatype property that currently exists in the OMDoc ontology is *hasText*, holding the full text content of an informal mathematical property (*CMP* in XML). Like *usesSymbol*, this does not always represent the structure of the XML markup without loss. A *CMP* may consist of plain text, but it may also contain mathematical objects, textual references to mathematical symbols (see [section 4.3.2](#) for an example) and annotated rhetorical structures.

### 3.2.2.5 A Non-Design Goal: Maintaining Integrity by Sealing Theories

While the ontology generally tries to be as faithful as possible to the intended semantics of the OMDoc XML language, one particular design principle behind the latter cannot be satisfied in an ontology. The markup for certain relations on the theory and statement level is deliberately restricted in OMDoc XML to support knowledge base administrators in maintaining the integrity of theories.

The division of statements into constitutive and non-constitutive ones and the encapsulation of constitutive elements in theory elements [whereas non-constitutive ones may be added externally and point to their home theory] add a certain measure of safety to the knowledge management aspect of OMDoc. Since XML elements cannot straddle document borders, all constitutive parts of a theory must be contained in

<sup>13</sup>So far, this only covers simple definitions (class *SimpleDefinition*), i.e. definitions that define a symbol in terms of one mathematical object, e.g.  $s := f(t)$ . It remains to be done for implicit and recursive definitions, whose content is more complex.

a single document; no constitutive elements can be added later (by other authors), since this would change the meaning of the theory on which other documents may depend on. (MICHAEL KOHLHASE [[Koho6b](#), chapter 15.1])

Thus, it becomes possible to seal a theory by write-protecting it. Several statement-level relationships are also affected by that decision in that they can only be stated in one direction, pointing from non-constitutive statements to constitutive ones.<sup>14</sup> For instance, the *example/@for* attribute points from an example to the statement it exemplifies – a symbol declaration, a definition, an axiom, etc. When the theory containing a statement is write-protected, one can still add examples – which is reasonable, as examples do not change the meaning of a theory.

In a semantic web setting, which commonly assumes an open world (cf. [section 2.3.4](#)), such restrictions cannot be enforced in the same way as in a self-contained knowledge base. Syntactically, any author is free to assert additional properties about a theory, simply by using the URI of that theory as a subject in his own RDF triples. Within the same knowledge base, change management mechanisms driven by dependency relations such as the ones mentioned above can help to assess the impact of such additional statements on the original theory. When taking into account knowledge from the whole Web, provenance and trust mechanisms, which are a subject of semantic web research (cf. [section 2.3](#)) but not of this thesis, may be employed to maintain integrity; for example, a system could prefer statements from the original source document of the theory over external statements.

### 3.2.2.6 The Presentational Dimension – Symbol Notation Definitions

Notation definitions for symbols (cf. [sections 2.1.5](#) and [2.4.5](#)) do not belong to the logical/functional structures in a narrow sense but to the dimension of presentational structures. However, they are often managed in combination with the logical/functional structures, as argued in [section 3.1](#). The OMDoc ontology features a statement-level class *NotationDefinition*, which, by its *rendersSymbol* property, points to a symbol declaration, or to more than one of them, when it defines a specific notation for a certain co-occurrence of symbols, as explained in [section 2.4.5.4](#). For pattern-based notation definitions, *rendersSymbol* flattens the structure of the content markup pattern and thus loses information, which is similar to our treatment of informal and formal mathematical properties described in [section 3.2.2.4](#).

From the interaction of notation definitions with logical/functional structures, one can observe another kind of dependency: dependency w.r.t. presentation. When a symbol occurs in a mathematical object, changing a notation definition given for that symbol may affect the presentation of the object. For example, in a system that renders content markup to presentation markup, the object might have to be re-rendered. That is reflected by the property *possiblyUsesNotation* shown in [figure 3.2](#).

Note the restriction “possibly”: When multiple notations have been defined for a symbol, it is non-trivial to determine which one is used for rendering an object in which that symbol occurs

<sup>14</sup>This explains some of the link directions in OMDoc, but not all of them. For example, the reason for a proof pointing to the assertion that it proves, which actually seems to contradict the common practice of mathematical textbooks and in languages for formalized mathematics, where a proof “belongs” to a theorem or immediately follows it, is that there can be multiple proofs for an assertion, possibly even without the author of the assertion being aware of it.



(cf. [Mül10a, chapter 4] for further details). The selection of notations may depend on a static – i.e. completely determined by the markup of the document [collection] – and dynamic – i.e. determined when a user requests to view a rendered object – presentation context.<sup>15</sup> An example for the former is the language, in which the document has been written. An example for the latter is the profile of the user reading the document. For now, I consider all possible uses of a notation definitions constituting a dependency, albeit a weak one. Strong static dependencies could possibly be determined by an ontology, using rules beyond the expressivity of DL, but determining dynamic dependencies has to be left to a context-aware rendering engine, such as the one described in [Mül10a, chapter 4].

### 3.2.2.7 Current Coverage

In its current state, the OMDoc ontology does not yet cover the logical/functional structures of mathematical knowledge to the same extent as OMDoc. This section discusses the current coverage and points out possible extension paths. I first report for each module of the OMDoc specification and schema to what extent the ontology covers it – in case it models logical/functional structures – and what still has to be done.

**MOBJ (Mathematical Objects):** The OMDoc ontology does not aim at fully representing mathematical objects in RDF, for reasons discussed in the review of RDF in [section 2.5.1](#). The flat representation of symbol occurrence in mathematical objects suits those services discussed in the remainder of this thesis, whose implementation is based on the OMDoc ontology. The current representation exactly meets their information retrieval requirements, without introducing further complications. Beyond that, a representation that emphasizes symbols occurring in certain key positions of mathematical objects has successfully been used in the HELM and MONET projects and would be straightforward to apply in the OMDoc ontology as well. Even a full, possibly redundant, preservation of mathematical objects as XML literals, as has been done for OpenMath CDs, would work.

**MTXT (Mathematical Text):** Those elements of the MTXT module that represent [informal] logical structures are covered by the OMDoc ontology. This includes a flat representation of the text of informal mathematical properties. Using a rhetorical ontology to represent their phrase structure in more detail is comparable to the HELM/MONET approach to mathematical objects; this is discussed in [section 3.3](#).

**DOC (Document Infrastructure)** is not related to the logical/functional structure of mathematical knowledge but discussed in [section 3.3](#).

**DC (Dublin Core Metadata), CC (Creative Commons Metadata)** and other metadata vocabularies are discussed in [section 3.4](#).

**RT (Rich Text Structures):** In itself, OMDoc's rich text syntax (HTML-like lists, tables and links, as well as indices) does not have a mathematical semantics. If the pertinent structure of lists

---

<sup>15</sup>This is not exactly the distinction that CHRISTINE MÜLLER makes in [Mül10a], but it is better suited for the explanation that I am giving here.



and tables in mathematical documents should ever become relevant for applications, it would again be straightforward to enhance the document ontology (cf. [section 3.3](#)) accordingly.

**ST (Mathematical Statements)** are mostly covered. The formal elements of the statement level will undergo a major revision with OMDoc 1.6. Aspects still missing in the current version of the ontology will be added then. This includes roles of symbols, the detailed content and relations of definitions of different types (simple, implicit, and recursive), and alternative definitions (which might be abolished). The proof status of assertions (e.g. “satisfiable”) is discussed below. Finally, those metadata that make OMDoc theories compatible to OpenMath CDs – such as the next review date of a CD – are borrowed from the OpenMath CD ontology, whose metadata vocabulary is introduced in [section 3.4.5](#).

**PF (Proofs and proof objects)** are mostly covered, except for references to methods, i.e. inference rules of some calculus employed in order to make derivations, and the implications of proofs on the truth of assertions (see below). The order of proof steps is only represented insofar as they have explicit links to each other.

**ADT (Abstract Data Types)** have not yet been covered at all.

**CTH (Complex Theories), DG (Development Graphs):** The only aspect of complex theories that has been covered so far is the representation of imports as first-class objects and the corresponding inference of dependencies. The coverage will be extended in parallel to the integration of MMT into OMDoc 1.6, which will yield major changes compared to OMDoc 1.2 with regard to the representation of theories.

**EXT (Applets, Code, and Data)** have not yet been covered at all.

**PRES (Presentation Information)** is very basically represented so far, as explained in [section 3.2.2.6](#). As in the case of the **MOBJ** and **MTXT** modules, the full structure of pattern-based notation definitions is not represented so far, but flattened. More vocabulary, covering presentation context dimensions as well as properties of declarative notation definitions, such as roles of symbols, is to come along with the merger of the pattern-based and declarative notation definitions of OMDoc 1.3 and MMT towards OMDoc 1.6.

**QUIZ (Infrastructure for Assessments)** has not yet been covered at all.

### 3.2.2.8 Future Directions: General and Truth-Related Dependency Relations

Besides enhancing the ontology by classes and properties that immediately represent the elements and attributes mentioned in the previous section, there is also potential for further inferencing by refining the class and property hierarchy, particularly with regard to dependency relations.

Particularly on the statement level, there are properties for which a reasonable common super-property has not yet been found, or probably does not exist. Consider, for example, the properties *exemplifies* and *proves*: In an educational setting, one could adopt MathLang’s notion of dependency (cf. [section 2.1.6](#)) and say that both examples and proofs provide additional background knowledge about the item they refer to – an example shows how to apply a symbol/definition/axiom/assertion in a particular setting, whereas a proof demonstrates why an assertion is true. In a

formalized library, however, examples and proofs are quite different, proofs being much more important. Therefore, I leave the introduction of such superproperties to future elaboration. Another possible future direction is modeling mereological relations such as parthood more exactly. In recent research, multiple types of them have been distinguished by their transitivity properties and the kind of entity types they relate (cf. [KAo8]).

The general view on dependencies introduced in [section 2.1.6](#) particularly allows for multiple kinds of dependency relations. So far, I have modeled dependencies affecting well-formedness and presentation. Besides identifying further dependency relations, I consider it worth investigating whether the *general dependsOn* property, i.e. the superproperty of all specific dependency properties, will prove useful in practical applications. My initial assumption is that it will – when maintaining knowledge bases that serve multiple purposes; consider an interactive educational system, where the knowledge is not only prepared in a human-comprehensible way but also used internally for computations that the users try out interactively, such as automated reasoning or computer algebra. A first observation on reasoning with subproperties of the transitive *dependsOn* property is that this may introduce strongly connected components in the dependency graph. Consider the example given in [section 2.1.6](#) that there can be two dependency relations  $Proof \xrightarrow{d_1} Assertion$  and  $Assertion \xrightarrow{d_2} Proof$ : This would make any proof/assertion pair mutually dependent via the *dependsOn* property, thus creating a strongly connected component  $C$ . Now if there is another knowledge item  $i$  depending on one member of  $C$ , such as an example for using the assertion, it will depend on all members of  $C$ . For a generic knowledge management application, this means that editing either member of  $C$  can affect some integrity property of  $i$ .

Concerning assertions and proofs, it should be noted that the ontology does not have any notion of truth. This is not as serious a shortcoming as it seems at first glance. First, I summarize how the ontology already enables useful facts to be inferred from a purely structural representation of assertions and proofs. Then, I estimate the additional benefits of enhancing the ontology by a notion of truth. With the ontology in its current state, we can already require, e.g., theorems to have a proof (cf. [section 3.2.2.2](#)). We can also infer from the structure of a proof what axioms or assertions its validity depends on (cf. [section 3.2.2.4](#)). On this ground we can add an axiom to the ontology stating that the truth of an assertion depends on the validity of at least one proof given for it, and further capture OMDoc’s notion of a *grounded proof*, whose truth only depends on axioms, well-defined definitions, or on assertions again having grounded proofs, as conceptualized in [Koho6b, chapter 17.2].

Beyond the structural validations enabled by such ontology axioms, one can easily imagine knowledge management use cases where a real notion of truth would help, e.g. in collaborative formalization efforts such as the ones mentioned in [section 1.4](#). Think of a query for all assertions in a knowledge base that are known to be true, or an inference rule saying that all assertions having a valid proof are true. In any case, these properties would have to be established by external means, such as type checkers, proof assistants or human experts. This is evident for assertions made in a logic that is more expressive than the one of the ontology, and for their proofs. If a logic at most as expressive as the one of the ontology is employed, the truth of an assertion can be determined by the same reasoner that is operating on the representation of the logical structures represented

using the ontology, but still that reasoner would not generally be able to check a proof given for such an assertion.<sup>16</sup>

Finally, a notion of truth in the ontology would allow for implementing what the OMDoc specification conceptualizes as the “*proof-theoretic status of an assertion*” [Koho6b, figures 15.6 and 15.7]: For an assertion  $\mathcal{F}$  with a sequent structure (i.e. using  $\langle \text{FMP} \rangle \langle \text{assumption} \rangle \mathcal{A}_1 \langle \text{assumption} \rangle \dots \langle \text{assumption} \rangle \mathcal{A}_n \langle \text{assumption} \rangle \langle \text{conclusion} \rangle \mathcal{C}_1 \langle \text{conclusion} \rangle \dots \langle \text{conclusion} \rangle \mathcal{C}_m \langle \text{conclusion} \rangle \langle \text{FMP} \rangle$ ), OMDoc can express whether  $\mathcal{F}$  is, among others, a tautology (meaning that the assumptions and some of the conclusions are satisfied in all models of the current theory; justified by a proof of  $\mathcal{F}$  and a refutation of  $\neg \mathcal{F}$ ), a theorem (meaning that there are models of the assumption set  $\mathcal{A}$  and that all of them satisfy some conclusions  $\mathcal{C}_i$ ; justified, as said above, by a proof of  $\mathcal{F}$ ), or satisfiable (meaning that there are some models of  $\mathcal{A}$  that satisfy some  $\mathcal{C}_i$ ). These statuses can be modeled as another hierarchy of subclasses of *Assertion*, e.g. *Tautology*  $\sqsubseteq$  *Theorem*  $\sqsubseteq$  *Satisfiable*  $\sqsubseteq$  *Assertion*<sup>17</sup>, also with complements and disjointness axioms for some of the other proof statuses from [Koho6b, figure 15.7], and the existence of [valid] proofs or models would have to be represented by additional properties.

### 3.2.3 The OpenMath Content Dictionary Ontology

Despite some deficiencies, the OpenMath CD language is still widely in use. In order to make knowledge represented in this language comprehensible to users and services in a semantic web setting, it has to be translated to an ontology-based representation as well. I first discuss possible migration paths and then present my approach – a simple ontology directly derived from the OpenMath 2 abstract information model for CDs and its reference XML encoding.

#### 3.2.3.1 Why an OpenMath CD Ontology?

OpenMath’s restricted CD language did not perform particularly well in the assessment of knowledge representation capabilities in section 2.5.1 because of its poor coverage of logical/functional structures and different degrees of formality. Furthermore, the more expressive OMDoc language fully subsumes it, as noted in section 2.4.3. Nevertheless, we have to face the fact that it is being used more widely for implementing CDs<sup>18</sup>, which may contain mathematical knowledge worth reusing. OMDoc may more faithfully capture the full semantics of mathematical concepts, integrated with informal documentation, but the OpenMath CD language is easy to learn and apparently sufficient for a weak formalization of an interchange vocabulary used in application settings focusing on a restricted set of tasks. So the task remains to translate OpenMath CDs to an ontology-based representation.

<sup>16</sup>personal communication with FLORIAN RABE, 2010-01-14

<sup>17</sup>It is not yet certain whether the *Theorem* class in this equation is the same as the *Theorem* class mentioned in section 3.2.2.2, which on a purely structural level represents an assertion with a proof.

<sup>18</sup>This judgment is in terms of the numbers of users, not necessarily in terms of the number of existing CDs. Consider, for example, the modules of MICHAEL KOHLHASE’S  $\text{\LaTeX}$  lecture notes mentioned in section 2.4.7.2, which declare more symbols than all OpenMath 2 CDs contributed to the collection at [openmath.org](http://openmath.org) (1613 symbols as of August 2010). For full compliance with the OpenMath standard, they would merely have to be enriched by those metadata fields that the standard mandates (see below).

The JOMDoc library for OMDoc (cf. [Jom]) includes a rudimentary XSLT implementation of a translation of OpenMath CDs to OMDoc and vice versa (*cd2omdoc.xsl*). It does not currently cover the full CD language – support for most CD-level metadata and for symbol roles is missing – but would be straightforward to extend. The OMDoc theories obtained that way could then be translated to RDF using the OMDoc ontology described in [section 3.2.2](#).

However, a case study central for this thesis dealt with providing an integrated collaboration environment in which people from the OpenMath community could manage the official and contributed OpenMath CDs (cf. [chapter 10](#)). In that setting, I considered it more appropriate to design an ontology that directly captures OpenMath’s conceptual model of CDs, for the following reasons:

**Consistent terminology:** Terms from any ontology chosen for representing the knowledge managed in an integrated environment are likely to appear on the user interface, either directly or translated from a technical to a more natural language. It has to be assumed that seeing familiar terms facilitates working with the system.<sup>19</sup>

**Consistent structure:** Knowledge in OpenMath CDs is structured differently from OMDoc theories. In an OpenMath CD, mathematical properties and examples are linked to one symbol declaration (as children of the respective *CDDefinition* element, cf. [listing 2.5](#)). In contrast, an OMDoc theory hosts declarations, mathematical properties (i.e. definitions, axioms, and assertions), and examples on the same level and allows them to point to each other (by *@for* attributes, cf. [listing 3.1](#)), including links with multiple targets, such as an example for a definition and an axiom.<sup>20</sup> Moreover, OpenMath, does not embed type signatures into symbol declarations, as OMDoc does, but stores them in external signature dictionaries. Again, in the envisaged system, users would browse and edit the CDs along the structures defined by the representation – markup language and ontology – chosen for them. By the same argument as above, I considered the original OpenMath way of structuring preferable.

**Lower error-proneness:** While I was free to choose the knowledge representation internally used by the system, the system was required to act as a frontend to a repository hosting the CDs in an OpenMath representation. Therefore, if OMDoc had been chosen for representing the CDs in the system, a potentially error-prone translation not only from OpenMath CDs to OMDoc theories but also vice versa would have been necessary.

The following sections introduce the OpenMath CD ontology and then discuss possibilities to align it with the OMDoc ontology.

### 3.2.3.2 Structures of OpenMath 2 CDs

The OpenMath CD ontology tries to directly capture the abstract information model of a CD. Most of its terminology is borrowed either from the XML reference encoding for CDs [BCC+04,

---

<sup>19</sup>This assumption has been confirmed by the evaluation summarized in [section 10.4](#).

<sup>20</sup>Actually, all of these links are redundant as soon as there is an *FMP* contains a mathematical object with explicit references to symbols; however, linking, e.g., an axiom to a *particular* symbol emphasizes the salience of that symbol for the axiom.

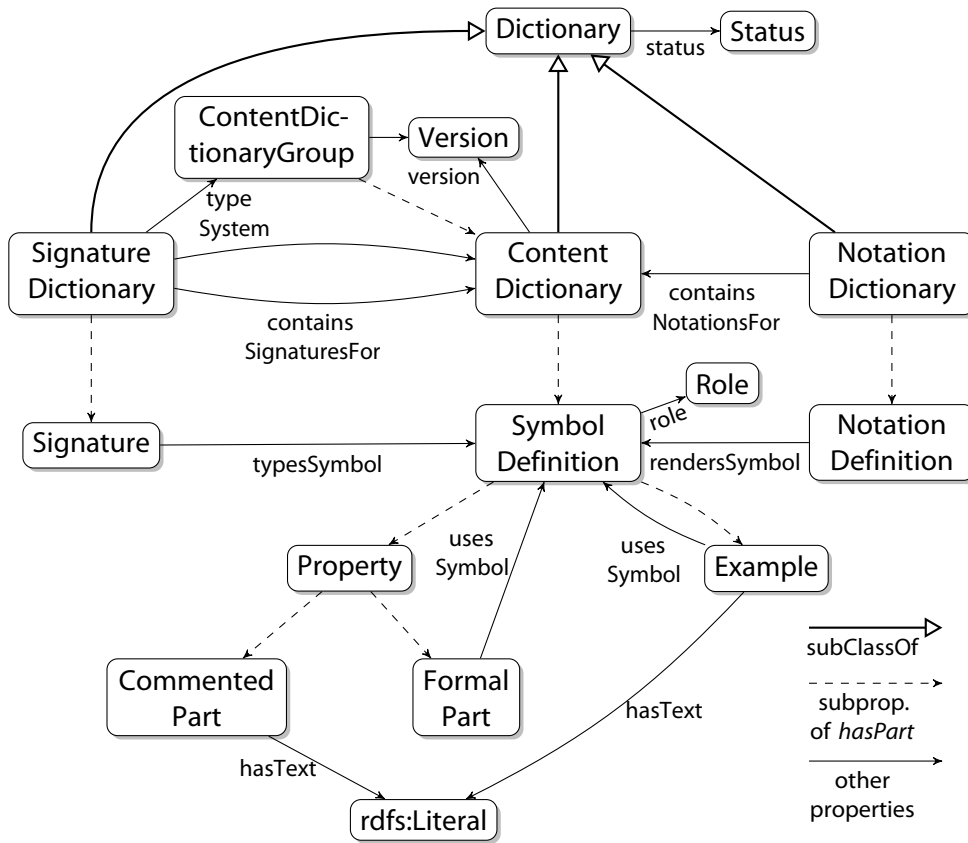


Figure 3.3: Classes and object properties of the OpenMath CD ontology (slightly simplified)

chapter 4.3] or from the abstract CD specification [BCC+04, chapter 4.2]. In some cases, I had to make a choice, for example in the case of symbol definitions. The abstract CD specification calls them “symbol definitions”, whereas the XML encoding calls them *CDDefinitions*. I consider the former more intuitive.

The ontology has classes and properties for all structural entities found in CD groups, CDs, type signatures, and notation definitions<sup>21</sup>. Figure 3.3 shows most of the ontology; a full listing of classes and properties is provided in appendix B.2. The ontology also covers metadata of CDs as discussed in section 3.4.5.

As pointed out in the previous section, the structure below the CD level is a tree, in which occurrences of symbols in mathematical objects inside *FMPs* or examples are the only cross-references: CDs exclusively contain symbol definitions, and mathematical properties and examples are exclusively parts of symbol definitions. Above CD level, there are CD groups. Instead of declaring to support a large number of individual CDs, an OpenMath-aware system can more conveniently declare to support a smaller number of complete CD groups. One CD can be a member of multiple groups; for example, the *arith1* CD is a member of the *mathml* group (MathML compatibility,

<sup>21</sup>Notation definitions are not part of the OpenMath 2 standard, but there is some consensus that notation definitions, official or not, should be stored in notation dictionaries in parallel to the CDs, similar to signature dictionaries.

subsuming all symbols that exist in pragmatic Content MathML) and of the *arith* group (basic arithmetic functions), and users are free to introduce additional groups. Thus, comparing the sub-CD level to the super-CD level, there are two different notions of containment: exclusive containment, captured by a transitive property *hasPart* and an inverse functional subproperty *hasDirectPart* as in the OMDoc ontology (cf. [section 3.2.2.4](#)), and non-exclusive membership, captured by a transitive property *comprises*.

The way the ontology models mathematical properties of symbols follows the proposed addition of a container element, which holds either a single *CMP*, a single *FMP*, or a pair of a *CMP* and an *FMP*, which are assumed to be semantically equivalent. In OpenMath 2, a symbol definition can have *CMP* and *FMP* children in arbitrary order according to the RELAX NG schema and the specification, but if there are semantically equivalent *CMP/FMP* pairs, it is recommended to put them in sequence.

The occurrence of symbols in *FMPs* and examples is represented in the same way as in the OMDoc ontology (cf. [section 3.2.2.4](#)), i.e. by flattening the functional structure of mathematical objects using a *usesSymbol* property. OpenMath's *CMPs* are limited to flat text anyway; thus, representing their content using a *hasText* property does, in contrast to OMDoc, not lose structural information. The *Example* element supports mixed content of text and OpenMath objects; here, *usesSymbol* and *hasText* lose information about the positions where objects are embedded into the text.

### 3.2.3.3 Modeling Mandatory CD and Symbol Properties

The specification of abstract CDs mandates CDs and symbol definitions to have certain pieces of information (cf. [listing 2.5](#) for an example). These requirements can be modeled using DL property restrictions. A symbol, for example, must have exactly one identifier of datatype *NCName* (an XML name without colons) and may have up to one role out of binder, attribution, semantic attribution, error, application, and constant. This is modeled by the following axioms<sup>22</sup>:

$$\text{SymbolDefinition} \sqsubseteq =1 \text{dct:identifier}$$

$$\text{SymbolDefinition} \sqsubseteq \forall \text{dct:identifier.xsd:NCName}$$

$$\text{SymbolDefinition} \sqsubseteq \leq 1 \text{Role}$$

$$\text{Role} = \{ \text{Binder}, \text{Attribution}, \text{SemanticAttribution}, \text{Error}, \text{Application}, \text{Constant} \}$$

### 3.2.3.4 Dependencies

The weak semantics of OpenMath CDs does not allow for inferring as many dependencies as the OMDoc ontology does. Of the three cases of dependency w.r.t. well-formedness that the OMDoc ontology currently covers (cf. [section 3.2.2.4](#)), two can also be used with OpenMath CDs:

1. *Symbol–hasOccurrenceOfInType–Symbol*, when applied to OpenMath's *SymbolDefinitions*, would work in the same way as in OMDoc. Note, however, that the STS type signatures,

<sup>22</sup>Note that the DCMI Terms ontology, which is reused for some CD metadata (cf. [section 3.4](#)), is only implemented in RDFS. Therefore, the OpenMath CD ontology has to declare appropriate property types for those DCMI Terms properties it reuses. It declares *dct:identifier* an *owl:DatatypeProperty*.



which are most commonly used with OpenMath CDs, are deliberately weak, as discussed in [section 6.3](#). For example, the only dependencies that can be inferred from the type signature of `arith1#plus`, as shown in [listing 6.1](#) on [page 200](#), are  $\text{arith1\#plus} \xrightarrow{d} \text{sts\#mapsto}$  and  $\text{arith1\#plus} \xrightarrow{d} \text{sts\#nassoc}$ , and, in one further step,  $\text{arith1\#plus} \xrightarrow{d^*} \text{sts\#nary}$  (via  $\text{sts\#mapsto}$ ).

2. OpenMath CDs do not have a notion of importing other CDs.<sup>23</sup> One can, however, look up all CDs from which symbols are used in examples and *FMPs* of one CD. This can be captured by the following ontology axioms, where  $\text{definesSymbol} = \text{definedIn}^{-1}$ :

$$\text{definesSymbol} \circ \left\{ \begin{array}{c} \text{exemplifiedBy} \\ \text{hasProperty} \circ \text{hasFormalPart} \end{array} \right\} \circ \text{usesSymbol} \circ \text{definedIn} \sqsubseteq \text{usesDirectly}$$

3. In the absence of definitional *FMPs* (cf. [section 2.4.3](#)), OMDoc's *hasOccurrenceOfInDefinition* cannot be used in OpenMath CDs.

Dependency w.r.t. presentation can be defined for OpenMath CDs exactly in the same way as for OMDoc (cf. [section 3.2.2.6](#)).

### 3.2.3.5 Related Work, Discussion, and Future Alignment with OMDoc

Finally, I discuss two directions for evolution the OpenMath CD ontology: A comparison with an earlier approach at representing CDs in RDF indicates possible internal improvements, whereas an alignment with the OMDoc ontology will facilitate knowledge exchange and reuse.

I did not consider an existing older ontology for OpenMath CDs reusable (cf. [section 2.4.10.2](#) and [Busoi]), because (i) it covers less of OpenMath (no symbol roles, no type signatures, no CD groups), and (ii) it is only implemented in RDFS and therefore cannot fully model the concept of an abstract CD – which requires some DL constructs<sup>24</sup>, as demonstrated in [section 2.4.10.2](#). However, it has two notable features that may be worth adopting: (i) It represents the full content of *FMPs* and *Examples* as XML literals, as discussed in [section 2.4.10.2](#). For the applications presented in this thesis, that approach was not necessary, but it is worth noting that adding this feature to my ontology would make it fully satisfy the abstract CD specification and thus establish RDF graphs in terms of the OpenMath CD ontology as a full standards-compliant representation of CDs. (ii) In the sample RDF graph in [Busoi], one mathematical property is shared among multiple symbols. While that might make sense from a mathematical point of view – consider the example  $\text{odd}(n) = \text{even}(n-1)$  –, the OpenMath standard does not allow it, and therefore my ontology does not support it. (I have declared the relation of a mathematical property to a symbol functional.) When translating an OpenMath CD from XML to RDF, it would, furthermore, be non-trivial to determine what symbols an *FMP* referencing multiple symbols should be attached to. For example above-mentioned *odd/even FMP* most reasonably describes the *odd* and *even* symbols, but less so the minus operator. This question can probably only be answered by manual annotation – as, e.g., in OMDoc, where one would say `<axiom for="odd even">`.

<sup>23</sup>The deprecated *CDUses* element, which still occurs in some old CDs, was merely a materialization of information that could also have been looked up by inspecting the examples and *FMPs*.

<sup>24</sup>To be fair, it should be noted that OWL was not yet available when that ontology was implemented.



A final question that remains to be answered is how to exchange and reuse knowledge represented in terms of the OpenMath CD ontology, given that the OMDoc ontology offers a more expressive alternative. [Section 3.2.3.1](#) points out the structural differences between OpenMath CDs and OMDoc theories, whereas [section 3.2.3.4](#) shows how structural similarities enable us to reuse complex axioms already formalized in one ontology. The structural differences could be bridged by axioms that align both ontologies. Research on ontology alignment has yielded results that we can build on; for example, ISABEL F. CRUZ and HUIYONG XIAO have explored the use of ontologies for the particular purpose of integrating data represented in different XML schemata [[CX05](#); [CX09](#)]. Aligning the OpenMath CD ontology and OMDoc the OMDoc ontology will enable us to study the correspondence of OpenMath CDs and OMDoc theories in more detail than the purely syntactic translation mentioned in [section 3.2.3.1](#), and it will complement MICHAEL KOHLHASE's and FLORIAN RABE's investigations on a formal semantics for OpenMath ([[KR09](#)]) by also covering informal descriptions and metadata.

This section concludes with an initial outline of possible alignment steps. One can generally observe that, while OMDoc is much more expressive than the OpenMath CD language, the latter is more restrictive in what information it requires to be given (cf. [section 3.2.3.3](#)). Every OpenMath symbol and every OpenMath CD can be interpreted as an OMDoc symbol or theory, respectively; the OpenMath semantics proposed by KOHLHASE and RABE justifies that [[KR09](#)]. Conversely, however, not every OMDoc symbol or theory meets the OpenMath requirements, as they lack certain mandatory metadata. From that point of view, one could declare OpenMath symbols a subclass of OMDoc symbols, and OpenMath CDs a subclass of OMDoc theories. Structural differences on the statement level are also easy to handle. For example, an OpenMath CD that defines a symbol with a mathematical property and an example can be interpreted as an OMDoc theory having a symbol declaration, an axiom, and an example, the latter two pointing to the symbol. It is probably safest to interpret all mathematical properties as axioms. So far, all of these alignment axioms can be expressed in *SRIOQ*. Figuring out which *FMPs* are real axioms, i.e. indispensable, is, however, beyond the capabilities of a structural ontology, as it requires reasoning in the logics in which the *FMPs* are expressed.

## 3.3 Rhetorical and Document Structures

Rhetorical structures in the sense of RST (cf. [section 2.1.3](#)) and document structures (cf. [section 2.1.4](#)) are two separate dimensions of knowledge but often studied in combination. For both structural dimensions, reusable ontologies exist. This section introduces them and discusses how they can be integrated with the OMDoc/RDF representation that I have chosen for mathematical knowledge.

### 3.3.1 SALT and Related Rhetorical and Document Ontologies

While logical/functional structures of mathematical knowledge may occur on their own, e.g. in formalized knowledge bases, rhetorical structures are usually studied in the context of documents written in, e.g.,  $\text{\LaTeX}$  or an XML language. Two very similar families of ontologies suitable for modeling rhetorical structures in mathematical documents are SALT [[GHM+07](#); [GMH+07](#)] and

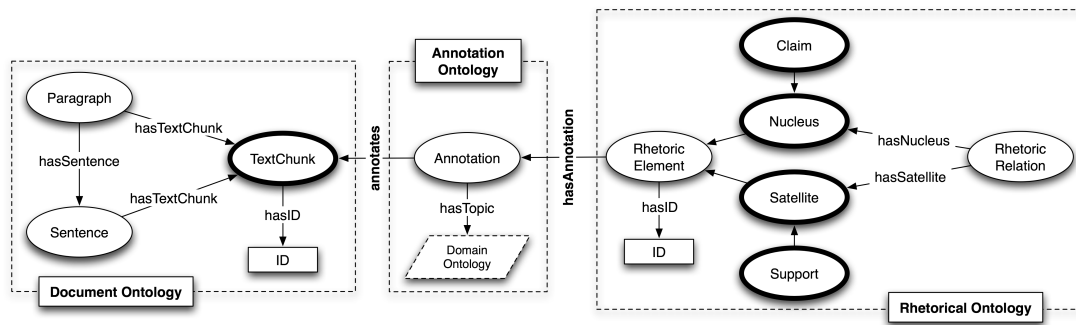


Figure 3.4: The three-layered architecture of the SALT ontologies (simplified) [GMH+07]

OntoReST [NJSSM+09]; further related models and ontologies have been reviewed in [GHC+09]. SALT and OntoReST are relevant for the following reasons:

- Both have a good coverage of RST-style rhetorical structures.
- Either use case is related to mathematical collaboration: SALT focuses on annotating and linking scientific publications on the Web and has, as OMDoc, a  $\text{\LaTeX}$  input syntax (cf. [section 2.4.7](#)). OntoReST focuses on consistency checking in concurrent collaborative writing.
- Both allow for an arbitrarily fine-grained annotation of phrases. SALT additionally focuses on cross-document links for justifying statements by citing the claims made [and justified] in external publications [GMH+07].
- Both are, in principle, open for integration with arbitrary domain knowledge – which would be mathematical knowledge in our case.

Both approaches comprise three ontologies. As that model has originally been introduced by SALT and then adopted by OntoReST, and as OntoReST focuses on rhetorical structures, I explain the model of SALT here, as shown in [figure 3.4](#):

**The Document Ontology** models the outline of the document substrate – sections, paragraphs, sentences, and text chunks (in OntoReST: “spans”) below sentence level [GH09b]. The latter remain in the original representation of the document; SALT provides standoff markup via start and end pointers to their positions in the full text. Additionally, one can represent the linear order of document units by numbering them.

**The Annotation Ontology** connects instances of document ontology classes with annotations of their rhetorical structure and with background knowledge from domain ontologies, such as the topic of a section [GH09a]. While rhetorical structures are the primary focus of SALT, the mechanism is sufficiently general to also permit annotation of other structural dimensions.

**The Rhetorical Ontology** covers RST-style rhetorical relations [GH09c]. Their nuclei and satellites, subsumed as “rhetorical elements”, are linked to text spans in the document via the annotation ontology. Rhetorical elements can optionally be annotated with their *argumentative* roles in a DILIGENT-like way (cf. sections 2.1.8.2 and 3.6.1). As an alternative to the elaborate RST model, coarse-grained rhetorical blocks can be applied on top level of a document. OntoReST only implements RST, but in a stronger OWL formalization that supports consistency checking [NJSSM+09].

The document tree serves as a substrate carrying the other structural dimensions of knowledge. The other dimensions – the rhetorical dimension in the case of a SALT document, and both the rhetorical and the logical/functional dimension in the case of a well-structured mathematical document – have their own tree structures – RST or a logical theory/statement/object structure – and are tied to the document tree by the annotation ontology. This “annotation” approach allows for modeling each dimension independently. Particularly in the case of document sections corresponding both to a rhetorical and a mathematical entity – and possibly to other entities from yet other dimensions –, this approach scales better than the MathLang approach (cf. section 2.4.6), where each unit of a document can only be annotated as a document section and as a mathematical statement.

### 3.3.2 Mapping OMDoc’s Rhetorical Markup to SALT

Syntactically integrating the SALT ontology into OMDoc is largely straightforward. OMDoc 1.2 already has markup for document and rhetorical structures. The *omgroup* element represents a document section, *omtext* a paragraph or a mathematical statement, and *phrase* any chunk of text below the paragraph level, e.g. a subordinate clause. *omtexts* and *phrases* can be cross-linked and can have RST-like rhetorical roles. These roles roughly correspond to RST relation types and have been underspecified so far, as the specification text of *omtext/@type* shows:<sup>25</sup>

[The *omtext/@type* attribute] can take e.g. the values *abstract*, *introduction*, *conclusion*, *comment*, *thesis*, *antithesis*, *elaboration*, *motivation*, *evidence*, *transition* with the obvious meanings. In the last five cases *omtext* also has the extra attribute *@for*, and in the last one, also an attribute *@from*, since these are in reference to other OMDoc elements. [...] Further types of text can be specified by providing a URI that points to a description of the text type. (MICHAEL KOHLHASE [Koho6b, chapter 14.3])

A semantics for this syntax can now be defined by mapping it to concepts from the SALT ontology. Let us reconsider the rhetorical structure of [example 2](#) from [section 2.1.3](#), as depicted in [figure 2.2](#). In OMDoc, this can be represented as shown in [listing 3.2](#). Note that OMDoc does not support the “condition” relation natively; therefore, we reference the respective term in the SALT ontology. This would translate into the SALT RDF graph shown in [listing 3.3](#), where the resource URIs are taken from the document fragment IDs, where these exist, and further resources, which do not directly correspond to document fragments, are represented as blank nodes.

<sup>25</sup>*phrase/@type* is specified analogously. *omgroup/@type* does not provide rhetorical values by default but also allows for URI values.

Listing 3.2: OMDoc markup of our RST example

```

<p xml:id="paragraph">
  <phrase xml:id="condition" for="#claim"
    type="http://salt.semanticauthoring.org/ontologies/sro#Condition">
    <phrase xml:id="condition-core">Let  $m$  be ...</phrase>
    <phrase xml:id="condition-detail" type="elaboration" for="#condition-core">
      Suppose that ...
    </phrase>
  </phrase>
  <phrase xml:id="claim">
    <phrase xml:id="claim-core">Then ...</phrase>
    <phrase xml:id="claim-detail" type="elaboration" for="#claim-core">
      with  $g$  a constant satisfying ...
    </phrase>
  </phrase>
</p>

```

Listing 3.3: SALT RDF graph obtained from [listing 3.2](#)

```

<#paragraph> a sdo:Paragraph .

_:annotation
  a sao:Annotation ;
  sao:annotates <#paragraph> .

_:relation
  a sro:Condition ;
  sro:hasNucleus <#claim> ;
  sro:hasSatellite <#condition> ;
  sro:hasAnnotation _:annotation .

<#claim>
  # a subclass of Nucleus reserved for the most important claims of the document
  a sro:Claim ;
  sro:hasNucleus <#claim-core> ;
  sro:hasSatellite <#claim-detail> .

<#claim-core> a sro:Nucleus .
<#claim-detail> a sro:Satellite .

<#condition>
  a sro:Elaboration ;
  sro:hasNucleus <#condition-core> ;
  sro:hasSatellite <#condition-detail> .

<#condition-core> a sro:Nucleus .
<#condition-detail> a sro:Satellite .

```

### 3.3.3 Discussion and Future Work: Alignment with OMDoc

The SALT ontology provides a good coverage of document and rhetorical structures and integrates well with the existing markup of OMDoc. Its extensible annotation approach fits particularly well into our scenario of representing multiple dimensions of knowledge. This section points out some remaining problems and discusses how they can be addressed.

In [listing 3.3](#), only the top-level paragraph is represented in terms of the document ontology and annotated with the top-level rhetorical relation. A full representation of sentences and subordinate clauses in terms of the document ontology, with annotation links to the finer rhetorical elements would capture the structure of the OMDoc document more faithfully. That could only be achieved by introducing additional RDF resources that do not correspond to markup in the document. For example, the *claim* phrase is only marked up as a rhetorical element, but not explicitly as a document unit. In documents with logical/functional markup, the same holds, e.g., for mathematical statements, which only have statement markup but no section markup. This may present a challenge for user interfaces. They either have to give users intuitive access to such implicit document structures, insofar as they are needed as a base for exploring the other structural dimensions that are attached to them via annotations, or they have to concentrate on the structural dimension that is most relevant in the respective application scenario.

Another problem that can be seen from [listing 3.3](#) is a slight structural mismatch between the conceptual models of OMDoc 1.2 and SALT – concretely: OMDoc’s phrases and SALT’s rhetorical elements. The OMDoc 1.2 markup does not have a notion of rhetorical elements (nuclei and satellites), which participate in rhetorical relations such as “elaboration”. Instead, it has a single type of link, represented by *phrase/@for*, and gives phrases rhetorical roles such as “elaboration for another phrase” – which, by the way, includes support for nuclei participating in multiple rhetorical relations. The SALT model is closer to the original RST but creates more overhead for the document author. (The actual SALT L<sup>A</sup>T<sub>E</sub>X package does not cover rhetorical relations, but only rhetorical blocks and claims [[Gro09](#)].)

Besides such differences on the statement and phrase level, OMDoc 1.2 lacks native vocabulary for rhetorical structures on document level, be it fine-grained rhetorical relation types that are supported by *omtext* and *phrase*, or the coarse-grained rhetorical blocks of SALT. However, the *omgroup/@type* attribute also supports arbitrary URI values – which could be taken from the SALT rhetorical ontology. Conversely, some of the rhetorical relations that OMDoc supports – with an unspecified semantics, however, – are not supported by SALT. [Table 3.1](#) lists the differences and suggests steps for resolving them. In case of conflicts, the original RST reference was consulted [[MT](#)]. Where SALT misses RST relations, they could possibly be reused from OntoReST, which has full RST support [[NJSSM+09](#)].

## 3.4 Metadata

Most of the metadata vocabularies introduced in [section 2.1.7](#) have been implemented as ontologies. Some can be reused directly, whereas certain metadata propagate along other structural dimensions in a way that suggests customizing them by additional axioms. Other metadata vocabularies still have to be implemented as ontologies in order to become usable in our setting.

Table 3.1: Rhetorical relations in OMDoc 1.2, RST, and SALT

OMDoc 1.2	SALT		RST		Suggested action
	Relation	Block			
abstract	–	✓	summary		check whether it is needed as a fine-grained relation; if so, add “summary” to the ontology
introduction	preparation	motivation background	motivation preparation background		replace by “motivation”/“preparation”/“background”, which are more exact; add missing relations to ontology <sup>a</sup>
conclusion	consequence <sup>b</sup>	✓	justify <sup>c</sup>		use “consequence” on phrase level
comment	–	–	–		remove, or map to non-rhetorical metadata term, e.g. <i>rdfs:comment</i>
thesis	–	–	–		the target of an antithesis? If so, probably redundant $\Rightarrow$ remove
antithesis	✓	–	✓		keep as is
elaboration	✓	–	✓		keep as is
motivation	–	✓	✓		add relation to ontology
evidence	✓	–	✓		keep as is
transition	–	–	joint? sequence? <sup>d</sup>		model as two relations <sup>e</sup>

<sup>a</sup> probably with different names, as SALT, unfortunately, keeps both relations and blocks in the same namespace

<sup>b</sup> does not exactly match “conclusion”, does not exist in RST

<sup>c</sup> the inverse direction

<sup>d</sup> RST defines these as multinuclear relations, which connect two or more nuclei. This does not reflect the OMDoc practice, where a transition is different in type from the two sections it connects.

<sup>e</sup> A transition from *A* to *B* can be considered as participating in two relations at the same time: summarizing *A* and preparing *B*.

### 3.4.1 Directly Reusable Ontologies

The authoritative specification of some metadata vocabularies is given as an abstract model, but an officially endorsed ontology exists. This is, e.g., the case with the Dublin Core vocabularies [NPJ+08], the combination of MARC relators with Dublin Core [DCM05], and Learning Object Metadata [IEE02b; NPBo3]. Usually, in these cases the alternative implementation to an ontology is an XML schema. For other vocabularies, such as ccREL [AAL+08], SIOC and its action module [BBP+08; BBD+10; BB07; CP10], the ModelDriven.org Architecture Ontology [Mod], or OMV [HPH+; PHC+09], the ontology itself is the normative implementation. The stable ODRL version 1.1 has only been implemented as an XML schema, whereas for the upcoming version 2 an abstract core model and a common vocabulary will be specified, with concrete encodings in XML, RDF and a microformat [ODRa].

### 3.4.2 Propagation of Metadata Along Other Structural Dimensions

Propagation of metadata (cf. section 2.1.7.2) has so far been specified for OMDoc 1.2 [Koho6b, chapter 12.4] and, in a more detailed way, for its ActiveMath variant [Lib09]. In ActiveMath, for example, all DCMES metadata except *dc:title* and *dc:identifier* are subject to the “if-missing” inheritance policy, whereas certain educational metadata, such as the learning context, are subject to the “merge” policy. The only direction of propagation supported by OMDoc 1.2 is the syntactic parent→child direction. This is also the default behavior of ActiveMath, but there one can also request the metadata of an individual entity to be inherited from a particular referenced entity. A consistent propagation against *structural* dimensions is, however, not possible, unless declared completely manually.

So far, the implementation of this metadata propagation is purely algorithmic. It can partly be formalized in *SRQIQ*. The following axiom establishes, for example, that *dc:contributor* should be merged in the part→whole direction of the document ontology<sup>26,27</sup>:

$$sdo:hasPart \circ dc:contributor \sqsubseteq dc:contributor$$

We are not limited to inheriting the same property; we could alternatively state that only *dc:creators* of parts are recognized as contributors to the whole:

$$sdo:hasPart \circ dc:creator \sqsubseteq dc:contributor$$

Most of the metadata propagation specified in OMDoc 1.2 and its ActiveMath variant can, however, not be realized in *SRQIQ*. “if-missing” inheritance, for example, assumes the absence of a metadata field. Due to the open world assumption of DL, a metadata field is only considered missing when explicitly declared so ( $entity \in \forall field.\perp$ ) – which a document author would usually not do.<sup>28</sup> Secondly, the semantics of “if-missing” is inherently algorithmic (or temporal), as applying this rule changes the state of the knowledge base.

<sup>26</sup>OMDoc 1.2 and ActiveMath specify a whole→part inheritance for *dc:contributor*, which is wrong.

<sup>27</sup>In the current implementation of the SALT document ontology, a generic whole→part relation does not exist. I assume it does, being the common superproperty of *sdo:hasSection*, *sdo:hasSubSection*, *sdo:hasFigure*, *sdo:hasTable*, *sdo:hasParagraph*, *sdo:hasSentence*, and *sdo:hasTextChunk*.

<sup>28</sup>Section 6.3.3.1 points out possibilities for enforcing the local existence of metadata fields.



Another mechanism that cannot be formalized in full generality is inheritance from arbitrary referenced entities  $e'$ , as it would require unrestricted quantification over properties. Suppose that all properties  $p$  whose values  $v$  can be inherited by reference have been classified as subproperties of *InheritableProperty*; then, “merge” inheritance by reference could be modeled as follows in second-order logic:

$$\begin{aligned} &\forall e, e' \in \text{MathKnowledgeItem} \forall p \in \text{InheritableProperty} \forall v. \\ &\langle e, e' \rangle \in \text{inheritsMetadata} \wedge \langle e', v \rangle \in p \rightarrow \langle e, v \rangle \in p \end{aligned}$$

### 3.4.3 Formalizing and Implementing Classification Schemes as Ontologies

Classification schemes are a particular type of metadata vocabulary that is often not available as an ontology. They can be used in conjunction with ontologies in two ways. Either the identifiers of their categories are used as literal values of metadata fields such as *dc:subject*, or, if one wants to make more explicit what classification scheme has been used, one can introduce refined subproperties of *dc:subject*, such as *mnp:primarySubject* or *mnp:secondarySubject* from the MathNet ontology [Mata], which are required by specification to have an MSC value (cf. section 2.1.7.4). However, implementing a classification scheme as a proper ontology, where each category is a resource of its own, has further advantages: The hierarchy of categories can be made explicit, and URIs can be used more flexibly in SPARQL queries. In the course of the MONET project, an ontology for the problems of GAMS has been implemented as a simple class hierarchy [Mona]. The ACM CCS has been implemented as an ontology [DGN+03], drawing on the “classification” vocabulary of the Learning Object Metadata ontology. Similarly, the MSC 2010 is currently being translated to a taxonomy that uses the SKOS ontology<sup>29</sup> [W3c].

### 3.4.4 Mapping the OMDoc 1.2 Metadata to RDF

The vocabularies mentioned in section 3.4.1 are suitable for mapping the metadata syntax of OMDoc 1.2 to RDF, but not generally in a straightforward way. OMDoc 1.2, and, analogously, the corresponding version of  $\mathcal{J}\mathcal{T}\mathcal{E}\mathcal{X}$ , has hard-coded support for two metadata vocabularies: the DCMES with MARC relators and idiosyncratic versioning extension, and ccREL. This section introduces the mapping, whereas section 5.1 addresses the OMDoc 1.2 metadata syntax in detail, including sample listings.

#### 3.4.4.1 Mapping Dublin Core and Creative Commons Metadata

OMDoc’s DCMES and ccREL metadata can directly be mapped to the respective ontologies – with few exceptions, where the OMDoc 1.2 syntax permits expressions that are not covered by the ontologies. For Dublin Core metadata, DCMI Terms is actually a preferable translation target, as its RDFS formalization is richer but still backwards-compatible to the DCMES. OMDoc additionally allows for modeling the role – in terms of the MARC relators vocabulary – that a *dc:creator* or *dc:contributor* took. These roles have been specified as subproperties of *dct:creator*, *dct:contributor*,

<sup>29</sup>personal communication with PATRICK ION, 2010-07-30

and *dct:publisher* [DCMo5] and can thus be translated to RDF in a way that is compatible with the translation of the Dublin Core metadata.<sup>30</sup>

OMDoc's syntax for ccREL metadata does not completely conform with the ccREL specification either. To understand that, one has to acknowledge that the rights management markup in OMDoc had been designed before Creative Commons started recommending RDFa. As embedded markup was required and Creative Commons at that time only suggested the impractical workaround of putting RDF/XML into XML comments of the document to be annotated, MICHAEL KOHLHASE modeled a custom XML syntax, closely but not exactly following the Creative Commons RDF schema. OMDoc 1.2 does not respect the ccREL semantics in that its CC syntax allows for constructing licenses that the abstract model of ccREL does not permit. For example, it is possible to say `<cc:permissions derivative_works="prohibited">`, although derivative works are only intended to be permitted<sup>31</sup>, whereas the only action that a Creative Commons license can prohibit is commercial use.<sup>32</sup>

### 3.4.4.2 Ontologies for Representing Revision Histories

Finally, OMDoc 1.2 has a simple vocabulary for recording revision histories: The additional *@who* attribute for the *dc:date* element refers to the URI of a *dc:creator* or *dc:contributor* in the same metadata record, and the *@action* attribute refers to an action verb out of the set “updated”, “created”, “imported”, “frozen”, “review-on”, and “normed”. Both of these features and their syntax have been inspired by the Open Packaging Format [Opf], a part of the EPUB specification (cf. section 2.4.8). However, a semantics has not been specified rigorously (see the citation in section 2.4.4.1). Thus, a versioning ontology has to be chosen (see sections 2.1.7.6 and 3.4.1 for an overview) that captures the semantics that has probably been intended for these annotations. Syntactically, the person and the action are provided as annotations to a triple, which itself records the date of a revision of a resource (e.g. *#fermat-proof*, *dc:date*, *1637-06-13T00:00:00*). While this *could* be modeled by RDF reification – i.e. conceiving triples as resources that can again be annotated –, reification is widely considered problematic (see, e.g., [CS04]), and all ontologies on consideration model a revision history as a linked list of revisions. As in the case of the translation of OMDoc's rhetorical markup to the SALT ontology discussed in section 3.3.2, the RDF representation of a revision history will therefore be structurally different from the XML syntax.

DCMI Terms can express revision histories as lists of resources linked via *dct:replaces/dct:isReplacedBy*. It does not have a dedicated vocabulary for actions that led to the creation of a resource but several specific subproperties of *dct:date*, describing the date when a resource was

<sup>30</sup> MARC relators on *dc:publisher* are missing in the OMDoc 1.2 specification. Moreover, the OMDoc 1.2 XML schema supports all MARC relators both on *dc:creator* and on *dc:contributor*, which conflicts with their specification. In such cases, I recommend deprecating the respective OMDoc 1.2 syntax and do not specify a translation to RDF.

<sup>31</sup> Under the open world assumption of semantic web reasoning, not explicitly granting the *permission* to create derivative works does not imply that they are prohibited.

<sup>32</sup> Once more, I recommend deprecating the non-conforming OMDoc 1.2 syntax and do not specify a translation to RDF. Note, however, that the RDFS implementation of ccREL cannot fully capture the intended semantics of the ccREL abstract model. As pointed out in [HPR+08], it does allow for prohibiting derivative works. From the range of *cc:prohibits*, an RDFS reasoner would infer (under an open world assumption!) that *cc:DerivativeWorks* is an instance of *cc:Prohibition*. The latter is not a contradiction in the RDFS, which does not allow for declaring *cc:Prohibition* disjoint with *cc:Permission*.

*dct:created*, *dct:validated*, *made* *dct:available*, *dct:issued*, *dct:modified*, *accepted* (*dct:dateAccepted*), *copyrighted* (*dct:dateCopyrighted*), *submitted* (*dct:dateSubmitted*). SIOC Core also supports lists of revisions, via *sioc:previous\_version* and *sioc:next\_version*, and additionally transitive closures of the former properties (*sioc:earlier\_version/sioc:later\_version*) [BBD+10]. The SIOC action module allows for associating actions with arbitrary artifacts [CP10]. Where DCMI Terms focuses on annotating [versioned] resources, SIOC employs an action-centric model. Actions are instances of the *sioca:Action* class and linked to – possibly multiple – artifacts via action verbs, which are subproperties of *sioca:object*. It is recommended to distinguish between a mutable object – such as this thesis –, which an action *sioca:modifies*, and its immutable revisions – such as the first draft of this thesis –, which an action *sioca:creates*. This distinction between resources and their revisions, which the action module makes, is, however, not shared by SIOC Core, which does not enforce the distinction of a resource from its latest revision. Most of the terms of the OMDoc action vocabulary are not available out of the box; however, extending the supply of SIOC’s action verbs is encouraged. Another shortcoming of the initial version of the SIOC action module is the lack of inverse properties of the action verbs, which would more intuitively match OMDoc’s resource-centric markup. Both DCMI Terms and the SIOC action module are prepared for being combined with other ontologies. Hardly any of the DCMI Terms properties has a specific domain declared, and the SIOC action developers argued “[The intent] of *DigitalArtifact*, representing the objects manipulated through Actions [...] is to represent any component of the Web-based applications targeted by SIOC. It is therefore a superclass of most SIOC classes, such as *sioc:Item* and *sioc:Space* [...]. However, for the sake of openness, we do not want to restrict *DigitalArtifact* to those classes, as they may not cover other kinds of digital artifacts that may emerge in the future.” [CP10]

OMV and the ModelDriven.org Architecture Ontology are both more expressive than DCMI Terms and SIOC and more expressive than OMDoc 1.2. OMV has a rich supply of actions representing structural changes. Instances of *omv:Ontology* are linked via *omv:hasPriorVersion*. To that mere sequence of revisions, a parallel sequence of changes can be connected. An *omvc:ChangeSpecification* connects two ontology versions by its properties *omvc:changeFromVersion* and *omvc:changeToVersion* and consists of a list of one or more *omvc:Changes* chained together by *omvc:hasPreviousChange*. A change has an author (an *omvc:Agent*), a date, and a few more properties. OMV has a lot of change subclasses specific to RDFS and OWL ontologies built in, e.g. classes describing the addition of a subproperty to a property or the splitting of one ontology class into multiple ones – without, however, referring to the affected entities of the ontology. We could easily add change types for mathematical knowledge items, e.g. a change type for adding a type signature to a symbol declaration. Summarizing, OMV can model the history of revisions to an *ontology*. By chapter 4, an OMDoc theory can be considered an ontology. For modeling revision histories of other structures, OMV is, however, too restrictive, as extreme modularization into little theories is not always an option. The versioning module of the ModelDriven.org Architecture Ontology [Mod] is directly applicable, as it allows for representing revision histories of generic “data assets”. An *mdv:Data\_asset* is distinguished from its *mdv:Data\_asset\_versions*. An *mdv:Data\_asset\_change* represents a transition from one *mdv:Data\_asset\_version* to the following one. A set of *mdv:Data\_asset\_changes* forms a *mdv:Transaction*, which is performed by an *mda:Authority*.

In conclusion, there are multiple ontologies for modeling revision histories, which allow for a more or less detailed representation. None of them exactly matches the expressivity of OMDoc 1.2,

but at least one of them will have to be chosen as a target of the OMDoc→RDF translation – most reasonably DCMI Terms, whose resource-centric view is least disruptive w.r.t. the OMDoc 1.2 syntax. However, in evolving documents in an environment with changing knowledge management requirements, flexible switching from one versioning ontology to a more or less expressive one might be required. Aligning the ontologies presented here would support that, but such an alignment will not be trivial, due to the structural differences the ontologies. It will not be possible in DL but most likely require FOL, as, e.g., DCMI Terms represents the type of a change as a date-valued property, whereas SIOC connects an action that has a date to a resource using an action verb, and OMV has classes representing changes.

#### 3.4.5 A New Ontology for OpenMath CD Metadata

In the particular case of OpenMath CDs, new metadata vocabulary had to be formalized and implemented. It has been realized as a part of the OpenMath CD ontology (cf. [section 3.2.3](#)). For some of the metadata of CDs and symbols, DCMI Terms (cf. [sections 2.1.7.3](#) and [3.4](#)) or RDFS could be reused, for example *dct:identifier* for the *Name* of a symbol<sup>33</sup>, or *rdfs:comment* for *CDComment* elements<sup>34</sup>. Others were found to be unique to OpenMath, such as the date of the next revision of a CD, or its status in the review process. [Table 3.2](#) lists both the reused and the new metadata fields. These metadata can also be reused when implementing OpenMath CDs as OMDoc theories according to [[Koho6b](#), chapter 15.6.2].

### 3.5 The Application Environment

Most contemporary systems for managing *mathematical* knowledge do not use an ontology-based formalization of the environment in which the mathematical knowledge is used. Nevertheless, a lot of existing ontologies cover aspects such as user interaction, maintenance workflows, and application scenarios, and can potentially be used for the benefit of MKM. This section provides a brief overview.

#### 3.5.1 Users and their Interaction with a System

The FOAF ontology (Friend of a Friend [[BM10](#)]) offers a basic representation of user profiles – persons that know other persons, belong to groups, create content, hold accounts in online services, etc. The SIOC ontology (Semantically Interlinked Online Communities, cf. [[BBP+08](#); [BBD+10](#); [BB07](#)]) extends FOAF by a more elaborate model of user-generated content, covering web-based discussion areas, such as blogs and message boards: Holders of *sioc:UserAccounts* create *sioc:*

---

<sup>33</sup>This decision was subject to debate, as the name of a symbol only locally qualifies it within a CD, and for globally unique identification the CD base URI is additionally needed. However, *dct:identifier* has been specified as “*an unambiguous reference to the resource within a given context*” [[DCMo8](#)] (emphasis added by the author), and I argue that the CD provides a sufficient context.

<sup>34</sup>From the schema of the OpenMath CD language, it is, however, not clear where certain comments belong to. (The abstract information model for CDs does not consider them.) For example, the top-level *CD* element allows *CDDefinition* children to be interspersed with *CDComments*, which, presumably, document the definitions. However, in practice they have also been used for documenting a group of a few preceding or following definitions; additionally, *CDDefinition* can have its own *CDComment* children

Table 3.2: Metadata in OpenMath CDs

XML element(s)	Metadata property	Value range
<i>CDComment</i>	<i>rdfs:comment</i> <sup>a</sup>	<i>rdfs:Literal</i>
<i>CDDate</i>	<i>dct:date</i>	<i>xsd:date</i> <sup>b</sup>
<i>CDName, Name</i>	<i>dct:identifier</i>	<i>xsd:NCName</i> <sup>b</sup>
<i>CDReviewDate</i>	<i>omo:reviewDate</i>	<i>xsd:date</i>
<i>CDStatus</i>	<i>omo:status</i>	<i>omo:Status</i> = { <i>omo:Official</i> , <i>omo:Experimental</i> , <i>omo:Private</i> , <i>omo:Obsolete</i> }
<i>CDVersion, CDRevision</i>	<i>omo:version</i>	<i>omo:Version</i> (with fields <i>omo:major</i> and <i>omo:minor</i> of type <i>xsd:nonNegativeInteger</i> )
<i>Description</i>	<i>dct:description</i>	<i>rdfs:Literal</i>
<i>Role</i>	<i>omo:role</i>	<i>omo:Role</i> = { <i>omo:Binder</i> , <i>omo:Attribution</i> , <i>omo:SemanticAttribution</i> , <i>omo:Error</i> , <i>omo:Application</i> , <i>omo:Constant</i> }

<sup>a</sup> In practice, *CDComment* often contains information about the author, license and revision history of a CD, but only as unstructured text.

<sup>b</sup> OpenMath-specific restriction, cf. [section 3.2.3.3](#)

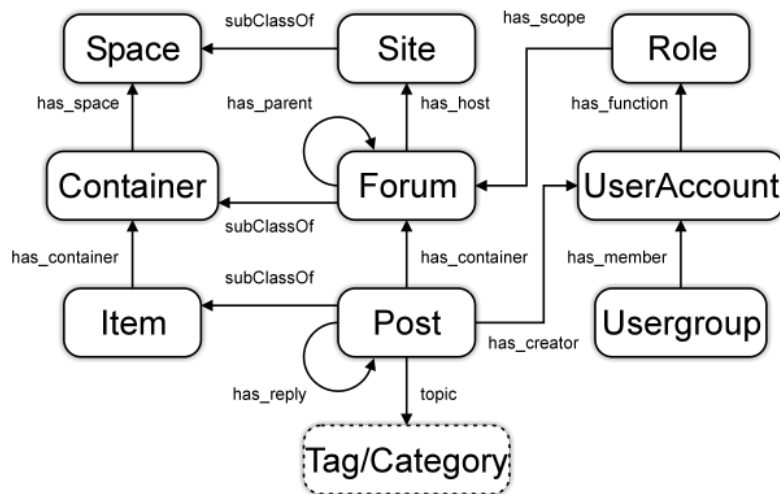


Figure 3.5: Main classes and properties of the SIOC Core ontology [BBD+10]

*Posts* in *sioc:Forums*, which are hosted on *sioc:Sites*. *sioc:Posts* can reply to other *sioc:Posts*. As SIOC evolved, these concepts were generalized to cover other kinds of user-generated content as well, as shown in [figure 3.5](#). Argumentation ontologies, which provide a more elaborate model of discussions, are covered in the following section. Finally, there are ontologies for user modeling, such as the General User Model Ontology (GUMO [[HSB+05](#)]).

#### 3.5.2 Application Domains of Mathematics

Notable examples of ontologies that model application areas of mathematics include SWEET (Semantic Web for Earth and Environmental Terminology [[Swe; RP05](#)]), GeoSkills, describing skills in interactive geometry, and the Statistical Core Vocabulary (SCOVO [[HHR+09](#)]), a lightweight vocabulary for statistical datasets. Brief overviews of SWEET and GeoSkills follow here, whereas [section 6.4.1.1](#) gives an example for integrating mathematical knowledge into statistical datasets.

The SWEET OWL ontology describes 4600 concepts in 150 modules from fields related to mathematics, such as physics, chemistry, biology, geology, and astronomy. These modules build on a foundation of general concepts of mathematics (e.g. functions), natural science (e.g. units), and space (e.g. coordinates). SWEET's model of mathematics does not intend to be as elaborate as ours and does not cover *structures* of mathematical knowledge, but SWEET provides a good showcase of how to integrate knowledge about mathematics with knowledge about its scientific application domains. In an application that focuses more on the mathematical model, the latter could be modeled using one of our ontologies for mathematics. One example of how SWEET integrates mathematics and science is the concept of a gravity field, defined as a vector field whose force is gravity. A vector field is a subconcept of a function whose result is a vector, and a vector is defined as an array of scalar elements.

GeoSkills is an OWL ontology describing topics, competencies and educational contexts related to interactive geometry [[Libo8](#)], albeit without a connection to a structural model.

### 3.6 Discussions about Knowledge Items

[Section 2.1.8](#) has reviewed argumentation models for discussing problems in knowledge and and particularly ontology engineering. The close relation of ontology engineering to mathematical knowledge engineering (clarified in [chapter 4](#)) suggests applying argumentation models to mathematical knowledge. Of the two argumentation models reviewed, I considered the DILIGENT model most appropriate for an application and specialization to MKM, due to its more rigid structure.<sup>35</sup> DILIGENT's core concepts served as extension points for mathematics-specific concepts.

This section describes the conceptualization, formalization, and implementation of a generic DILIGENT-inspired argumentation ontology based on SIOC ([section 3.6.1](#)), and its extension towards problems that occur in collaborative MKM ([section 3.6.2](#)). The conceptualization of the generic part alters the original DILIGENT model towards a higher flexibility in web [2.0] settings,

---

<sup>35</sup>The fact that DILIGENT had only been implemented in prototypical environments so far did not influence that decision, as my target environment for mathematical knowledge would not easily have permitted reuse of existing DILIGENT-based software anyway.



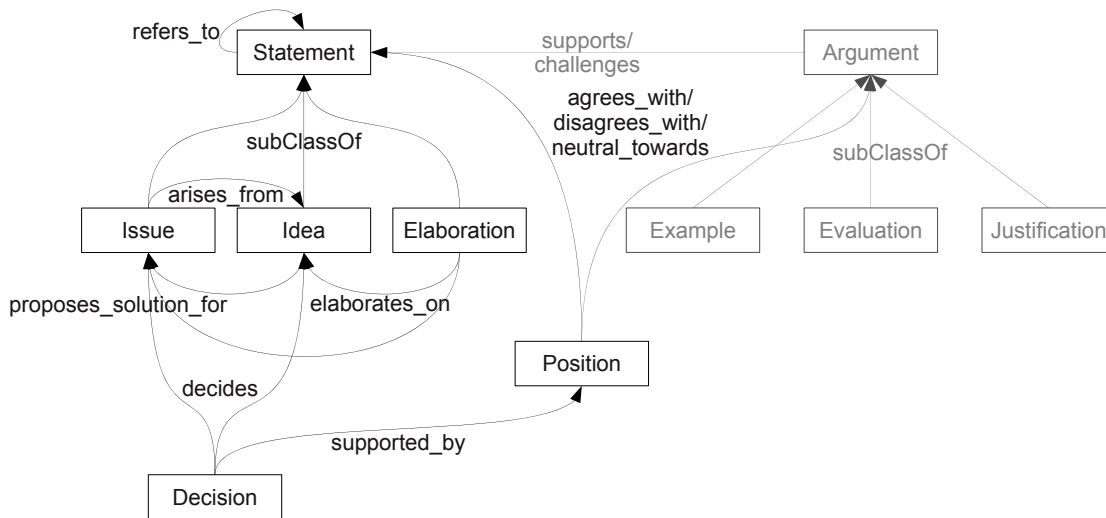


Figure 3.6: The SIOC argumentation module (optional parts in gray)

whereas I have conceptualized the mathematical extension from scratch due to the lack of previous results.

### 3.6.1 A Generic Argumentation Ontology

While the original DILIGENT model of argumentation has also been implemented as an ontology, we slightly diverged from its conceptualization and formalization and implemented our argumentation ontology – with my mathematics extension – as a module of SIOC. Collaboration with TUDOR GROZA on rhetorical structures in scientific documents (cf. SALT in [section 3.3](#)) and with ULDIS BOJĀRS on SIOC showed that argumentation can be used in a range of web [2.0] settings that is wider than DILIGENT’s application domain of ontology engineering (cf. [LBG+08]) and therefore needed a more flexible representation than the one given by the DILIGENT ontology. As SIOC was already widely accepted as a representation of web discussions (cf. [section 3.5.1](#)), we realized the core of our argumentation ontology, which is inspired by IBIS and DILIGENT, as a SIOC module. SIOC encourages the implementation of modules for specific applications (cf. [BBD+10]). The action module has already been mentioned in [section 3.4.4](#). Another module, SIOC Types, introduces subclasses of SIOC concepts in order to represent different kinds of social web objects more precisely. Among others, there is, a *sioc:Forum* subclass *sioc\_t:ArgumentativeDiscussion* representing “a discussion area where logical arguments can take place” [Sio]. Extending this towards a full model of argumentative discussions is the contribution that our argumentation module makes to SIOC.

The minimum requirement for modeling argumentation in a SIOC-compliant way is a class that can be assigned to any resource in addition to *sioc:Item* or *sioc:Post*, giving it the role of an argumentative *statement*. A post of type *sioc\_arg:Statement* is at the root of an argumentative dis-



cussion thread. Further *Statements*<sup>36</sup> can reply to it, connected by *sioc\_arg:refers\_to*, a subproperty of *sioc:has\_reply*. Starting from this, we specified additional classes and properties for arguments as subclasses of *Statement* or subproperties of *refers\_to*. The rationale for that design is to give developers and users a greater flexibility in identifying the argumentative types of their posts. This is different from the DILIGENT argumentation ontology, whose classes and properties do not have a common superclass or superproperty.

A description of the specific classes and properties in the order of an argumentative discussion follows: From the use cases of bug tracking (cf. [section 6.6.1](#)) and wiki discussion (cf. [section 9.1.4](#)) as well as forums and blogs (elaborated in [LBG+08]), we observed that discussions usually start with an issue or an idea. An *Issue* is a problem to be discussed, having a decision on a solution for the problem as an expected result. An *Idea* can take the role of a solution proposed for an issue or stand on its own. In the latter case, it can be a general idea, not proposing to solve any *particular* issue, or it can be a solution proposed for an implicit issue that has not been expressed as a discussion post. Conversely, *Issues* can also follow up on *Ideas* – particularly when a discussion has started with an *Idea* that turns out to be problematic. Most of our concepts (as depicted in [figure 3.6](#)) root in the DILIGENT argumentation ontology but have a slightly different semantics owed to our more general setting. A DILIGENT argumentation thread cannot start with an idea. Our more general understanding of issues is, however, still compatible with the common ancestor IBIS.<sup>37</sup> In our model, both *Issues* and *Ideas* can be followed up by *Elaborations*, which continue the line given by the parent statement.

Users can reply to *Issues*, *Ideas*, and *Elaborations* with *Arguments*, which can be justifications or challenges. An *Argument* tries to argue objectively; it is distinct from a *Position* (see below), which rather conveys the personal opinion of a user. *Arguments* may not be required in every use case; therefore, we treat them as optional. For example, in the Blogosphere, every post can be seen as a personal interpretation of the reality, while in a bug tracking system, comments are supported by real issues, thus having the circumstance of being considered objective. The role of an *Argument* can be resumed to: (i) an expression that states if an *Issue* is considered legitimate and worth discussing, and (ii) an expression that shows if an *Idea* can be considered a good solution. Subclasses of *Argument* comprise: *Example*, *Evaluation*, and *Justification*. The design of the *supports* and *challenges* properties was motivated by the DILIGENT variant used by the Cicero argumentation environment (cf. [section 9.5.7](#)). It allows for retrieving supporting or challenging arguments with one query step less than the original DILIGENT model with positive and negative argument classes and just one *arguesOn* property. Also, we opted for only a small set of *Argument* subclasses, as earlier studies in argumentation have shown that a restricted space of such types helps to keep a discussion more focused [PST04].

In a more subjective manner, users can express their *Positions* on a statement – agreeing, disagreeing, or neutral. Most argumentation ontologies do not support neutral positions, so as to force the argumentation towards solutions. Nevertheless, they are quite common in online discussions. In fact, a neutral position can be considered different from the absence of the position in that it expressed “I do care about this statement, I’m just not decided whether to support it or

<sup>36</sup>In the remainder of this section, I omit the *sioc\_arg* prefix when it is clear from the context that an entity belongs to that ontology module.

<sup>37</sup>... with the exception that IBIS assumes issues to be phrased as questions [KR70]. However, both DILIGENT and our variant are consistent with the IBIS requirement that “the origin of issues are controversial statements” [KR70].

not.” For a minimum working model, it is sufficient to support *Positions* on *Ideas*, but in a more elaborate model *Positions* on *Issues*, *Elaborations*, and even *Arguments* could make sense.

A decision taken at the end of an argumentative discussion can be documented by replying to the post that started the discussion – an *Issue* or an *Idea* – with a *Decision*. Decisions can also be taken on subtrees of a discussion, e.g. on one of the ideas for solving an issue, while leaving the overall issue open. In the case of deciding on an *Issue*, one should also link the *Decision* to the “winning” *Idea* and back it by links to the positions that were in favor of the action decided. If the idea was to create or modify a knowledge item, a *resolves\_into* link to [the current version of] the latter should be created, so that the community can transparently retrace the discussion that led to its creation or modification.

### 3.6.2 Mathematics-specific Extensions: Problems with Mathematical Knowledge and their Solution

Aiming at effectively supporting collaborative MKM by semantic services, I have developed a model that captures discussions about problems that occur in acquiring, conceptualizing, formalizing, organizing, and publishing mathematical knowledge. This model, grounded in observations of mathematical practice, refines a DILIGENT-style argumentation model (here: the one introduced in the previous section) by mathematics-specific *Issue* and *Idea* types.

#### 3.6.2.1 Conceptualization

To get an understanding of common issue and solution patterns in MKM, I conducted a survey among domain experts, initially focusing on the logical/functional and presentational dimensions of mathematical knowledge. The participants were asked for the types of knowledge items they had experienced issues with, what kinds of issues these were, and how these issues were solved; detailed questions and results are listed in [appendix D.1](#).

A majority of 30 out of the 52 participants was experienced in contributing to libraries of software tools like proof assistants; contributions to websites or open knowledge bases ranked second and third (25 and 22 participants, respectively). The most commonly experienced granularity of knowledge items was either a course unit, a mathematical theory (i.e. a few related definitions and axioms), or a mathematical statement. The participants reported few cases of automated issue tracking and solving being supported by knowledge bases.

The prevalent type of knowledge item that the participants had ever found affected by issues was the definition of a new mathematical symbol or concept.<sup>38</sup> About half of the 26 participants who answered that question had experienced issues with examples, theorems, proofs, theories, notations defined for symbols, and axioms. The most common issue was that a knowledge item was simply wrong or incomprehensible, followed by its truth being uncertain, being underspecified, or redundant. Further cases included knowledge items of which it was not clear whether they were useful, and knowledge items expressed in an uncommon style. Issues were mostly solved by directly improving the affected knowledge item (as opposed, e.g., to creating another one), by splitting it into more than one, or by deleting it altogether.

<sup>38</sup>The survey did not distinguish between symbol declarations and definitions.

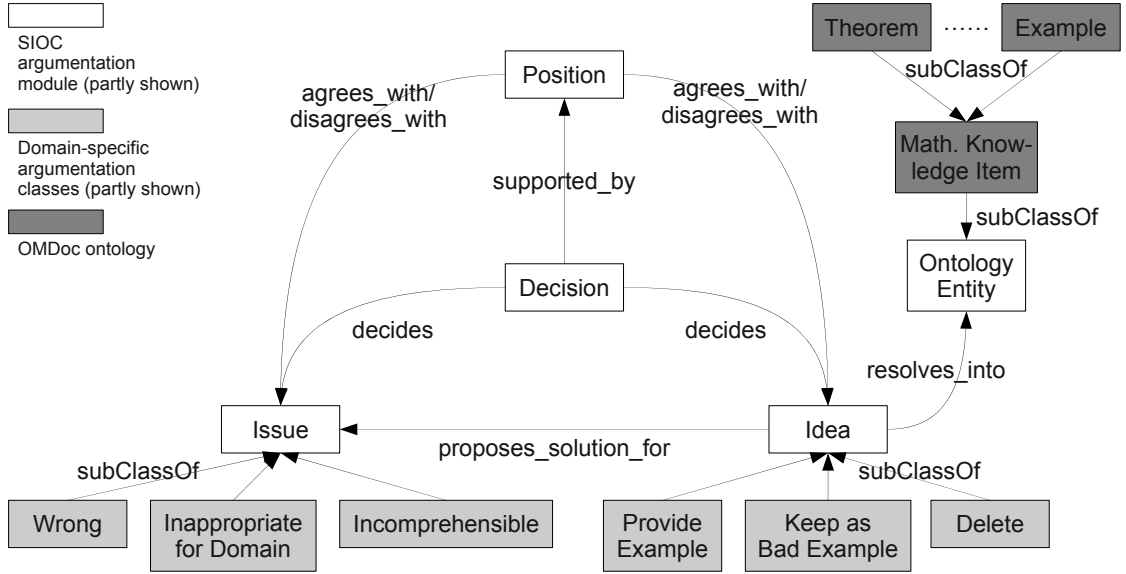


Figure 3.7: The SIOC/OMDoc argumentation ontology

The replies the participants gave about issues and ideas they had experienced influenced the further development of the model. The model also reflects that knowledge items, issues, and ideas cannot be combined arbitrarily. For example assertions, proofs, and examples can be wrong, whereas a notation can rather be inappropriate, misleading, or hard to read and write. Then, if some knowledge item is wrong, it could be deleted, or fixed in place, or kept as an instructive bad example, whereas splitting it into two parts would not solve that problem.

Finally, some participants had experienced issues being unresolved and mostly attributed this to an insufficient tool support for restructuring knowledge items. Other common reasons were insufficient awareness of the users that there is actually an issue, insufficient social interaction among users, as well as insufficient tool support for editing knowledge items.

### 3.6.2.2 Formalization and Implementation

The mathematics-specific *Issue* and *Idea* types have been formalized and implemented as subclasses of the former. Using the *matharg:appliesToKnowledgeItemType* meta-property, each *Issue* type is linked to a set of mathematical knowledge item types to which, including its subclasses, it can be applied. Similarly, the *matharg:appliesToIssueType* meta-property links an *Idea* type to a set of issue types – plus, optionally, a set of mathematical knowledge item types, – to which it can be applied. Neither the applicability of these meta-properties nor their propagation to subclasses of knowledge item or issue types, i.e. along *rdfs:subClassOf*<sup>-1</sup>, can be formalized in *SR**O**I**Q*, as these axioms involve quantification over both individuals and classes (i.e. unary predicates). In second-order logic, the semantics of *matharg:appliesToKnowledgeItemType* could be formalized as shown below; *matharg:appliesToIssueType* works analogously. The notation  $x$  is used for instances,  $t_x$  for their types.

$$\begin{aligned}
& \forall i \in \text{matharg:Issue} \forall t_i \in \text{owl:Class} \forall k \in \text{oo:MathKnowledgeItem}. \\
& i \in t_i \wedge \langle i, k \rangle \in \text{sioc\_arg:reported\_for} \\
& \rightarrow \exists t_k \in \text{owl:Class}. k \in t_k \wedge \langle t_i, t_k \rangle \text{mathArg:appliesToKnowledgeItemType} \\
& \forall t_i \in \text{owl:Class} \forall t_k \in \text{owl:Class} \forall t_{k'} \in \text{owl:Class}. \\
& \langle t_i, t_k \rangle \in \text{matharg:appliesToKnowledgeItemType} \wedge t_{k'} \sqsubseteq t_k \\
& \rightarrow \langle t_i, t_{k'} \rangle \in \text{matharg:appliesToKnowledgeItemType}
\end{aligned}$$

Table B.5 lists all issue and idea types implemented so far. While I have focused on issue and idea types that occur with mathematical knowledge items from the logical/functional and presentational dimensions, the schema of introducing new types is by no means limited to these dimensions. It is intended that communities adapt the ontology, e.g., to their application scenario, by adding further types.

### 3.6.3 Related Work

#### 3.6.3.1 Ontologies for Bug Tracking

In software engineering, discussions about problems with software artifacts, i.e. bugs, requirements, or feature requests, are well known. Several ontologies cover software engineering, but the particular aspect of bug tracking is not well represented. EvoOnt is a set of ontologies for modeling the whole process of software engineering, including, as one aspect, issue tracking [Evo; KBT07]. There are several subtypes of issues, such as defects (bugs) and enhancements (feature requests). An issue runs through several states, including “new”, “verified”, and “closed” and is eventually addressed with a resolution, such as “fixed”, “invalid”, “works for me”, or “won’t fix”. The BAETLE ontology (Bug And Enhancement Tracking Language) [BTS+] aims more specifically at improving bug retrieval across systems and projects by a unified model for all contemporary bug tracking systems but is still in a very early state of development.

#### 3.6.3.2 Classifying Posts with Multiple Argumentative Roles

COLIN FRASER et al. have developed an argumentation ontology for e-mails [FHT06], which covers the common case that one post agrees with some statements of the post it replies to, whereas it disagrees with others. To keep annotations easy to create for users, e-mails have shallow argumentative type annotations in their subject header. That means, however, that an e-mail that partly agrees and partly disagrees with another e-mail is classified both as an agreement and as a disagreement, which I do not consider quite useful for implementing sophisticated services on top of that model. In future, we intend to address this issue with a representation of fine-grained structures within posts.

#### 3.6.3.3 Scientific Discourse as Primary Knowledge

This section has primarily considered (external) discussions that collaborators hold *about* mathematical knowledge items in a knowledge base, but not the alternative perspective of scientific

discourse embedded into a knowledge base or a document (cf. [section 2.1.8](#)). Argumentation ontologies have also been used for the latter. For example, the SWAN ontology (Semantic Web Applications for Neuromedicine [[GHC+09](#)]) models scientific discourse in a distributed knowledge base by pointers to bibliographic records and entities from domain ontologies. Its primary target is neuromedicine, but, as SALT, it supports arbitrary domain ontologies in principle. SWAN is currently being aligned with SIOC [[PCB+09](#)], which may pave a path to an integration with the mathematical argumentation ontology presented here. Discourse inside documents is supported by the SALT rhetorical ontology, which allows to annotate the argumentative roles of rhetorical elements in documents using a variant of the vocabulary described in [section 3.6.1](#).

### 3.6.4 Conclusion and Future Work

With the SIOC argumentation module, we have developed a general-purpose argumentation ontology that is inspired by DILIGENT but more flexible. I have customized it to the mathematical domain by adding issue and idea subclasses for the most common problem and solution types that domain experts have encountered in their working practice and linking these issue and idea classes to those mathematical knowledge item types – from the primary structural dimensions – to which they apply. I believe that this method can be carried out analogously in any other domain where structured collaborative problem solution is of interest and where an ontology for the structures of primary interest exists.

[Section 6.6](#) deals with collaboration services that operationalize this argumentation ontology; this section concludes with future research directions from a knowledge representation point of view. In [section 2.1.8.2](#), the value of past discussions as a source of knowledge has been mentioned. With a machine-comprehensible argumentation ontology, such a corpus could be exploited in two ways:

**Identifying Problem, Solution, and Contribution Patterns:** One could try to automatically detect common problem and solution patterns. So far, we have focused on argumentation within a single, centralized site, but public discussion archives can also be utilized on the Web of Data; consider, for example, argumentative discussions spread across multiple blogs. As a concrete vision of how one community *C* can benefit from another community *D* exposing its argumentative discussions as linked data, consider a permission propagation mechanism based on an analysis of contribution patterns. For a specific example, suppose the administrators of *C* generally trust the users of *D*, and suppose the user *U* (where *U* refers to, e.g., a FOAF profile) is a member of both communities. By analyzing how many of *U*'s ideas posted in community *D* received positive feedback (in terms of *Arguments* or *Positions*) and finally got accepted (by *Decisions*), the administrators of *C* could assess whether *U* qualifies to receive extended permissions, such as the right to take decisions.

**Materializing Discussions into Primary Knowledge** – combining both perspectives on scientific discourse that have been mentioned initially in [section 2.1.8](#) – would allow for capturing further important mathematical practices. IMRE LAKATOS has studied how discussions about mathematical knowledge items materialize into new mathematical knowledge (cf. [section 2.1.1.3](#)). Consider, once more, a discussion thread in which a collaborator points out that a proof only covers a specific case and should be generalized. This discussion provides

the rationale for a later, generalized restatement of the respective theorem and its new proof and therefore could be integrated into the text that encloses the theorem and its proof. Future research could, building on the first steps reviewed in [section 3.6.3.3](#), investigate how to effectively support a semi-automated materialization of the argumentative structure of a discussion thread into rhetorical or logical structures, i.e. turning secondary into primary knowledge.

### 3.7 Requirements for Extracting Structures from Semantic Markup to RDF

The preceding sections have described a selection of ontologies that cover all relevant structural dimensions on mathematical knowledge. With the OMDoc and OpenMath CD ontologies, I have particularly provided a new ontology-based formalization and implementation of the conceptual models of those two semantic markup languages for logical/functional structures of mathematical knowledge. Thanks to these ontologies, an RDF representation of mathematical knowledge now satisfies those requirements that RDF did not meet in the assessment in [section 2.5.1](#). Therefore, we could now represent almost all mathematical knowledge in RDF, using these ontologies. However, the OMDoc and OpenMath CD ontologies assume that full mathematical objects are still represented in documents, as they can more naturally be represented, edited, and published that way – partly due to inherent shortcomings of RDF, but mostly due to better tool support for documents. For similar reasons, the SALT ontology does not represent the full text of a document in RDF, but instances of its document ontology point to those spans of the document that have annotations. Moreover, there a huge amount of mathematical knowledge exists in the form of documents. In order to make mathematical knowledge contained in documents comprehensible to a wider range of services, it has to be *translated* to RDF. That is, all of its relevant structures have to be identified in the document markup and extracted into an RDF outline.

The two main steps in extracting structures from markup to RDF are giving them identifiers, and representing them in terms of the appropriate ontologies. The general requirements for any implementation of these extraction steps, independently from a particular choice of markup language or ontology, are:

**Identifiers:** Unless the XML language supports RDFa – a case covered in [section 5.2.3](#) –, all structural entities that correspond to concepts covered by the given ontologies **MUST** be given an identifier by applying the first of the following rules that matches:

1. If the XML language specifies how to generate a URI for an entity represented by an XML fragment, that URI **MUST** be used. Consider, for example, the URIs of OpenMath symbols (cf. [section 2.4.3](#)) or the MMT URIs of OMDoc 1.6 (cf. [section 2.4.4.1](#)).
2. If the XML language specifies how to generate an ID for an entity, e.g. via XML ID [[MVWo5](#)], that ID **MUST** be used as a fragment ID (cf. [section 2.3.2.1](#)) if possible w.r.t. the syntax of URIs [[BLFMo5](#)]; appending it to the document's URI yields the URI.<sup>39</sup>

<sup>39</sup>In practice, most semantic XML markup languages support IDs on all elements, but authors only use them when an element is a target of an explicit link in the markup. Many RDF properties, such as whole→part relations, are,



3. If the XML language specifies how to generate an ID for an entity, which does not qualify as a fragment ID, the extractor **MUST** generate a fragment ID, which **SHOULD** reflect the original ID.
4. In any case, the extractor **MUST** generate a resource. It **SHOULD** be identified by a minted<sup>40</sup> URI, but it **MAY** also be a blank node with an ID, or an anonymous blank node. Minted URIs **MUST NOT** conflict with URIs generated for other entities in the XML document.

For authors and developers, reusing the identifiers from the XML markup in the RDF representation emphasizes the correspondence of both representations. For services, it improves retrievability, e.g., of RDF standoff markup for an XML representation: If a structural entity always has the same identifier, regardless of the representation format – semantic markup, RDF, or even a human-comprehensible presentation –, and if its different representations are published according to the “cool URI” best practices (cf. [section 2.3.1](#) and [SCo8]), all of them can be made available under the same URI (cf. [section 6.4.1.3](#)).

**Structures:** The extractor **MUST** map all structures that are supported by the given markup languages and the given ontologies from their markup representation to their RDF representation in terms of the given ontologies. Detailed requirements for this mapping arise from the schemata of the individual markup languages and the formalizations of the individual ontologies. The extraction **SHOULD**, however, not be hard-coded in a way that only works with the current state of the markup languages and ontologies, but it **SHOULD** be easy to add, adapt, and customize mappings.

The particular commitment to the OMDoc and MathML/OpenMath markup languages (without loss of generality, as argued in [section 2.5.2](#)) and the ontologies presented in this chapter entails a number of challenges to the mapping of structures – for example:

**URI Format Differences:** OpenMath symbols that occur in mathematical objects in OpenMath CDs always have URIs of the form *cdbase/cd#name* (cf. [section 2.4.3](#)). In OMDoc 1.2, the URI of a symbol is the URI of the fragment that declares a symbol named *name* (i.e. `<symbol xml:id="id" name="name">`) in the imported theory that has the name *cd* (cf. [section 2.4.4.1](#)).

**Mapping Elements to Classes:** Generally, OMDoc XML elements correspond to classes from the OMDoc ontology. However, the ontology has been designed with its utility for RDF-based applications in mind, not necessarily to represent the OMDoc XML markup literally. Therefore, instances of some subclasses are represented by the same element, only differing in the value of a certain attribute (e.g. the *@type* attribute; cf. [section 3.2.2.1](#)), or even by elements with different names (e.g. in cases of different degrees of formality; cf. [section 3.2.2.3](#)).

**Different Elements Representing Different Resources:** In most cases, different markup elements represent different resources. If a DL reasoner, which does not make a unique name

---

however, not represented by explicit XML links but by a parent-child relation, but triples using these properties require identifiable subjects and objects.

<sup>40</sup>The term “minting” URIs is common in linked data practice and refers to “the process to assign a unique global identifier [...] to a thing” [Hea+].



assumption, should take advantage of that, resources must explicitly be declared different<sup>41</sup>, at least in cases where their difference cannot be inferred by other means, such as them being instances of two disjoint classes.

**Markup Choices for Representing Relations:** Relations between two entities can be represented as a parent-child relation in the XML markup, as a sibling relation, e.g. *CMP-verbalizes-FMP* (cf. [section 3.2.1.2](#)), or by URI-valued attributes, or, in OpenMath CDs, child elements pointing from the element representing one entity to the other one. Some relations, such as *oo:homeTheory*, can be stated in both ways (cf. [section 3.2.2.4](#)). In OMDoc 1.2, definitions and axioms reference symbol declarations by their theory-local name, not by their URI. The *ID/IDREF* mechanism with document-local IDs, which many XML schema languages support, is, however, not employed here.

**Refining Relation Types:** As stated for classes above, the exact type of a relation is sometimes influenced by additional attributes on the same element; for example, *example[@type='for']/@for* translates to *oo:corroborates*, whereas *example[@type='against']/@for* translates to *oo:refutes*, and *example[not(@type)]/@for* translates to their common superproperty *oo:exemplifies*.

**Markup Choices for Representing Literal-valued Properties:** Literal-valued properties can be represented by text-valued immediate child elements (e.g. *CDDefinition/Name* for the name of an OpenMath symbol), by descendant elements nested more deeply (e.g. OMDoc 1.2 metadata), or by attributes (e.g. CD metadata of OMDoc theories).

**Whitespace in Literals:** MathML requires whitespace inside token elements to be normalized (e.g. `<mo>( </mo> ~ <mo>( </mo>)` [[ABC+10](#), chapter 2.1.7]. OpenMath does not specify whitespace normalization, but authors nevertheless commonly assume it.

**Implicit Structures:** The target ontologies reify certain concepts that do not have an explicit representation in the semantic markup, e.g. in the following cases:

- Equivalent informal and formal mathematical properties of a symbol in an OpenMath CD are not explicitly grouped, but it is assumed that the informal part (*CMP*) precedes the formal part (*FMP*; cf. [section 3.2.3.2](#)).
- SALT's document units, annotations, and rhetorical relations (cf. [sections 3.3.2](#) and [3.3.3](#)) are not always made explicit in OMDoc.
- The linked revision list structure of the versioning ontologies discussed in [section 3.4.4.2](#) is not explicit in OMDoc's revision metadata; they have to be constructed by ordering the *dc:date* metadata fields by date.
- The action that led to a revision can be represented at least in three different ways, depending on the ontology chosen: With DCMI Terms, it leads to a specific choice of *dct:date* subproperty – compare the *example/@type* case above! –, with SIOC Actions, actions are reified into resources, and the OMV and ModelDriven.org versioning ontologies reify them into change sets that again form a linked list.

<sup>41</sup>Note that the OWL 2 **DifferentIndividuals** (`i1 ... iN`) construct supports that in linear space [[MPSP09](#)].

**Alternative Representations of Classification Schemes:** Where a classification scheme has been implemented as an ontology, its categories are represented as classes or individuals. Otherwise, they are represented as RDF literals (cf. [section 3.4.3](#)). Similarly, some OpenMath CD metadata are represented as RDF literals, others (e.g. the status of a CD) as instances of an enumerated class (cf. [section 3.4.5](#)).

These observations require any language for specifying an XML→RDF mapping and its implementation to be highly flexible. [Section 8.1](#) presents and discusses my approach.

## 3.8 Related Work

Work related to detailed aspects of my approach has been discussed in the specific sections. Two more general aspects discussed in this section are specifying a semantics for markup languages, and representing mathematical domain knowledge using ontologies.

### 3.8.1 Markup Language Semantics

I have presented a way of making (not only mathematical) knowledge that is primarily represented in semantic markup languages more machine-comprehensible by translating this markup to RDF, whose semantics is then formalized in ontologies. I have focused on developing such target ontologies or reusing suitable existing ones, whereas I have specified the XML→RDF *mapping* informally so far.

In the BECHAMEL project, a generic high-level semantics for XML-based document markup languages has been developed, allowing to state, for example, that a particular XML child element represents a property of its parent or something that is part of the parent; however, it is not technically compatible with semantic web technology [[RDSM+02](#)].

The Yin/Yang model provides a unified model theory for both XML and RDF [[PSS03](#)]. However, its benefit is rather theoretical – SHENGPING LIU et al. have criticized its impractically restrictive assumptions about the XML schema, e.g. that an XML element always corresponds to an instance of a class with the same name [[LMY+04](#)].

In ROBERTO GARCÍA’s methodology of “XML semantics reuse” [[Gon05](#); [GCo5](#)], OWL ontologies are obtained by automatic translation from XSD. This translation preserves the semantics of certain constructs of the XSD schema language, such as substitution groups, subtyping of complex types, union and intersection types, and cardinality restrictions. Subsequently, instances of the XML Schemata, i.e. XML documents, are translated to instances of the OWL ontologies, i.e. RDF graphs. Here, again, it is assumed that XML elements always correspond to ontology classes, and XML attributes to properties. Gloze [[Bato6](#)] is a similar approach; it bypasses the schema/ontology level but also takes the XSD semantics into account and preserves schema information as RDF annotations. In addition to GARCÍA’s approach, it takes into account ID-type attributes and order. In the interest of compatibility with OWL reasoners, which cannot handle RDF collections, the order of those XML elements for which the schema declares order relevant is preserved in an optional overlay graph that arranges the RDF reifications of the actual RDF triples extracted from XML in an RDF collection. Thus, the *full* structure of the XML markup is preserved in the resulting RDF, which allows for translating the latter back to XML.

The OMDoc and OpenMath CD ontologies have been (manually) derived from an XML schema but in a less strict structural correspondence, instead aiming at a higher level of abstraction by making more use of the expressivity of *SRITQ*, which is higher than that of typical XML schema languages. *Bootstrapping* these ontologies could have been facilitated by having an initial OWL ontology auto-generated from an XML Schema and subsequently refining its formalization. For the instance-level XML→RDF translation, however, as well as the structural differences between the OMDoc and OpenMath CD markup languages and these newly developed ontologies and, more seriously, existing ontologies suitable for reuse, pose higher flexibility requirements – pointed out in [section 3.7](#) – than the general approaches of GARCÍA’s methodology, Gloze, or the Yin/Yang model could satisfy. Moreover, the existing approaches have been developed before the mainstream adoption of linked data and therefore do not ascribe adequate importance to a customizable minting of URIs in the RDF output. I have therefore satisfied the XML→RDF translation requirements from [section 3.7](#) by a language that allows for defining translation rules specific to the input markup language, target ontology, and desired URI format. [Section 8.1](#) presents that and discusses further work related w.r.t. the aspect of schema-/ontology-specific translation.

### 3.8.2 Mathematical Domain Ontologies

While the ontologies presented here model *structures* of mathematical knowledge, ontologies have also been used for modeling knowledge about the mathematical *domain* – i.e. instances of the structures, such as concrete definitions (“A monoid is a set with a binary operation that is associative and has an identity element”) or concrete theorems (“all differentiable functions are continuous”). Languages for representing mathematical knowledge have structural similarities with ontology languages in any case – as elaborated in [chapter 4](#) –, and, in fact, both have been used for the purpose of automated reasoning, but mathematical knowledge has also been represented in the “classical” ontology languages mentioned in [section 2.3.4](#). That usually demands focusing on selected aspects of mathematical concepts, as a full formalization would require more expressive logics. The facts that a differentiable function is a function that has a derivative function and that differentiable functions are continuous functions can be represented, e.g., in DL, as exemplified by MATTHIAS BRÖCHELER [Brö07] – but the fact that a differentiable function satisfies a certain  $\varepsilon/\delta$  criterion cannot, as it would require higher order logic. THOMAS R. GRUBER and GREGORY R. OLSEN have modeled mathematical concepts relevant for engineering in KIF [GO94]. FRÉDÉRIC FÜRST et al. have modeled concepts from projective geometry as conceptual graphs [FLT03]. These ontologies covered pure domain knowledge, no structural knowledge. BRÖCHELER, who implemented his DL formalizations of selected mathematical concepts in OMDoc, i.e. as OMDoc theories using a DL meta-theory, suggested an approach to unite structural and domain semantics [Brö07]. Working around the DL limitation of separating classes and instances, he took two perspectives on mathematical concepts – viewing them as classes in the domain ontology (e.g. *DifferentiableFunction*  $\sqsubseteq$  *ContinuousFunction*), but as instances in our structural ontology (*DifferentiableFunction*  $\in$  *oo:Symbol*, *DifferentiableDefinition*  $\in$  *oo:Definition*,  $\langle$  *DifferentiableDefinition*, *DifferentiableFunction*  $\rangle \in$  *oo:defines*) – and translated between both perspectives. Use cases were finding – via the structural ontology – examples for, e.g., groups – instances of a mathematical concept, determined via the domain ontology –, or finding applicable theorems or definitions about a mathematical concepts and – again via the domain ontology – all

related concepts. With the greater coverage of structures that the OMDoc ontology offers now (such as occurrences of symbols in mathematical objects), and with the proposed extensions (such as an in-depth treatment of the functional structure of mathematical objects, or a notion of truth), it seems feasible to automatically reduce a complex domain model, implemented as an OMDoc theory, to a simplified ontology.

As a concrete example, let  $D$  and  $C$  be OMDoc symbols for the sets of differentiable and continuous functions, let there be definitions for these symbols, an assertion (in higher-order logic) that “for all  $f$ , if  $f \in D$ , then  $f \in C$ ”, and a proof for that assertion. With a view from that logic to, e.g., DL, one could then leverage the structural information – i.e. that there is an assertion about two symbols  $D$  and  $C$ , which stating, translated to DL, that  $D$  is subsumed by  $C$  ( $D \sqsubseteq C$ ), and that that assertion has a proof – in order to obtain the axiom  $D \sqsubseteq C$  for the simplified domain model. Note that such domain relations, insofar as they are helpful for knowledge management, can also be treated like structural relations. For example, the ontology of the GeoText geometry textbook management system treats a subsumption-like “inheritance” relation between two definitions (“ $d$  is a special case of  $d'$ ”) as a structural one [Cheio]<sup>42</sup>. Thus, besides exporting an axiom like  $D \sqsubseteq C$  to a simplified domain model, we could add a structural relation “ $D$  inherits from  $C$ ” between two symbols to our knowledge base. A more pragmatic solution for knowledge management, requiring much less formalization effort, would, however, consist in adding an informally specified “inheritance” vocabulary term to our structural ontology and simply asserting such a relation between two symbols. The RDFa extension of OMDoc introduced in [chapter 5](#) enables such annotations.

### 3.9 Conclusion and Future Work

This chapter has presented ontologies for the structures of mathematical knowledge. While I have developed new ones for adequately representing logical and functional structures, I have been able to reuse existing ones for the other structural dimensions, while clarifying their reuse in a mathematical context. Uniformly representing mathematical knowledge in all of its structural dimensions in RDF using these ontologies makes its *structure* – not necessarily the *full* knowledge! – comprehensible to a wide range of services. Compared to the state before the introduction of these ontologies, services no longer have to understand a specific XML schema, and they get certain inferences (e.g. of dependencies) for free, given the availability of a suitable reasoning engine. Authors can still continue to use the existing OMDoc and OpenMath markup languages and the respective tools, as I have specified how to translate their semantic markup to RDF and provided an implementation of this translation (covered in [section 8.1](#)); thus, the entry barrier into contributing mathematical knowledge to the Web of Data remains low.

A welcome side-effect of specifying a translation from a semantic markup language to RDF is that it can help to uncover cases of underspecification of the former that would otherwise go unnoticed. Translating semantic markup to RDF forces one to make the identity of things explicit as URIs

---

<sup>42</sup>GeoText does not currently use an exchange-oriented representation language but a custom relational database schema. Its set of built-in types and relations – which users can extend – is similar to the OMDoc ontology but has only been specified very informally so far, so that the intended meaning of, e.g., “inheritance”, is not completely clear. For these reasons, its knowledge representation aspects have not been covered in further detail in this thesis.

and to make relations explicit as triples. Moreover, the conceptualization and formalization steps of designing an ontology that is more than a mere vocabulary of terms enforce an in-depth study of the schema of the respective markup language, and, where it exists, its abstract specification. The cases of OMDoc's underspecified *dc:date/@action* and *omtext/@type* attributes have been mentioned in sections 2.4.4.1 and 3.3.2, respectively, and the case of OpenMath's *CDComment* in section 3.4.5. Another case not mentioned so far even occurs in the abstract specification of a signature dictionary, where the relation to the CD, for whose symbols signatures are provided, is underspecified.<sup>43</sup>

When an author prefers markup languages different from those covered here, or when the application environment demands them, they can be used – as long as they support annotating documents with terms from ontologies, which several existing languages do (see, for example, section 2.4.8.6 and section 2.4.10.3). With this chapter, I have contributed a vocabulary for annotating mathematical structures in documents in such languages, or for representing mathematical knowledge as standalone RDF graphs. With the OMDoc and OpenMath CD ontologies introduced in section 3.2, I have created ontologies that cover the logical, functional, and presentational structures of formal and informal mathematical knowledge to an extent much larger than previously existing ontologies. With the mathematics-specific extension of the SIOC argumentation module, I have developed a novel ontology for discussing problems with mathematical knowledge items and their solution. Sections 3.3 to 3.5 do not introduce new ontologies but provide guidelines on how to reuse existing ontologies in conjunction with mathematical knowledge. Now, for example, a DocBook or XHTML+RDFa document annotated using these ontologies, becomes semantically equivalent to an OMDoc document. Similarly, a translation from mathematical markup languages, such as MathLang or CNXML to the ontologies introduced here could be implemented, following the methodology of section 3.2.1. That would turn these ontologies into a device for integrating heterogeneous data and enable high-level queries across multiple data sources (see [CX05; CX09] for an in-depth treatment of that topic). Even more generally, my methodology can be applied in *any* domain where a semantic markup language already exists, thus saving the effort of starting the ontology development from scratch with the conceptualization phase.

With the ontologies for representing all dimensions of mathematical knowledge and the translation from OMDoc to RDF that uses these ontologies, three problems have not yet been fully addressed:

**Ontology Implementation:** While the OpenMath CD ontology could be implemented in OWL, the *SR**OIQ* logic is not sufficiently expressive to restrict the possible interpretations of a vocabulary term (cf. section 3.2.1.3) in a way conforming with the intended semantics of OMDoc. This becomes apparent in a number of situations in the OMDoc ontology and in the interaction of certain ontologies with each other, e.g. in the case of metadata propagation. However, one goal of developing or reusing ontologies for representing mathematical knowledge was to give OMDoc 1.2<sup>44</sup> and the OpenMath 2 CD language a formal semantics in order to improve, for example, the precision of queries (cf. section 6.5.2) and the coverage of validators (cf. section 6.3). Chapter 4 introduces OMDoc as an expressive language

<sup>43</sup>The *CDSignatures* element merely refers to the *name* of a CD but not to a CD base URI [BCC+04].

<sup>44</sup>The logical/functional core of OMDoc 1.6 has a formal semantics; see section 3.2.1.3.

for logically heterogeneous and modular ontologies with comprehensive documentation facilities and then explain how the OMDoc ontology has been implemented.

**Multi-dimensional Structures and Metadata:** OMDoc 1.2 has a comprehensive coverage of logical/functional structures, presentational structures, rhetorical structures, and document structures of mathematical knowledge. But there are also relevant metadata vocabularies beyond the two ones natively supported by OMDoc 1.2 (DCMES and ccREL), for example more complex versioning ontologies (cf. [section 3.4.4.2](#)), there is application-related knowledge that can be represented using domain ontologies (cf. [section 3.5.2](#)), and there is structural and domain knowledge not yet covered by our ontologies, as discussed in [section 3.8](#). Currently, such knowledge would have to be maintained outside of an OMDoc document. The integration of RDFa into OMDoc presented in [chapter 5](#) enables integrated maintenance of all knowledge dimensions in the same document.

**Accessibility for Services:** Making knowledge comprehensible to services by representing it in RDF is the first step, but in a second step it has to be made *accessible* to services in the right application context. [Section 6.4](#) discusses how to deploy mathematical knowledge via two complementary distribution channels for knowledge on the Semantic Web: publishing linked data, which services can easily access and crawl, and embedding annotations into human-comprehensible documents, where interactive services such as those covered in [chapter 7](#) can utilize them in order to adapt the document's presentation or to look up and display further information in place.

Finally, any knowledge representation format, regardless of its coverage, is only as good as the services that support it – browsing documents and collections, discussing about knowledge items, editing and authoring, validation, searching, querying, and reasoning, and translation from and to other existing formats. [Part III](#) covers these services and their integration into documents, knowledge bases, and collaboration environments.

## Acknowledgments

ANDREI IONIȚĂ, NORMEN MÜLLER, KRZYSZTOF RETEL, GORDAN RISTOVSKI, and NIKITA ZHILTSOV (in alphabetical order) have contributed to the OMDoc ontology (cf. [section 3.2.2](#)). The initial mapping of OMDoc's rhetorical markup to the SALT ontology (cf. [section 3.3.2](#)) has been developed by JANA GIČEVA under joint supervision of TUDOR GROZA and me [[Gico8](#)] and further advice from SIEGFRIED HANDSCHUH and MICHAEL KOHLHASE. The SIOC argumentation module presented in [section 3.6.1](#) has been developed jointly with ULDIS BOJĀRS, TUDOR GROZA [[LBG+o8](#)], with further input from JOHN BRESLIN, SIEGFRIED HANDSCHUH, TUUKKA HASTRUP, STÉPHANE CORLOSQUET, THOMAS SCHANDL, CHRISTOPH TEMPICH, MAX VÖLKELE, and STEFAN DECKER. The brief description of the SIOC Core ontology in [section 3.5](#) is also partly based on that joint publication [[LBG+o8](#)].



# Using Mathematical Markup for Implementing and Documenting Expressive Ontologies

In the previous chapter we have seen a number of ontologies that can be used to represent mathematical knowledge in all of its structural dimensions. However, the formalization of certain structural aspects turned out to require a higher expressivity than the *SRIOQ* logic underlying the OWL ontology language could offer. Secondly, modular reuse and integration of ontologies is required to capture the interaction of some of the structural dimensions, such as the propagation of metadata along document or logical/functional structures, as well as structural similarities among alternative ontologies for the same task. Finally, such a complex formalization should have a comprehensive and comprehensible documentation.

This chapter argues that OMDoc is a suitable language for implementing and documenting such ontologies. Starting with a refined problem statement, [section 4.1](#) establishes requirements for an ontology language, and [section 4.2](#) reviews the state of the art. [Section 4.3](#) then points out the correspondences between OMDoc and [semantic web] ontology languages and leverages them for (re)implementing ontologies as OMDoc theories. We point out how OMDoc allows for fully capturing any given conceptual model while retaining compatibility with existing reasoners and semantic web tools as far as possible, for making interactions with imported ontologies explicit, and for embedding comprehensive documentation into ontologies in a way that supports services in making it comprehensible to users. [Section 4.4](#) briefly explains how the OMDoc ontology has been implemented in OMDoc. Finally, [section 4.5](#) argues that OMDoc does not just meet the special needs that we have encountered in *our* research, but that any existing semantic web ontology can benefit from enhancing its formalization, modularity, and documentation in OMDoc. This claim has been validated by reimplementing FOAF, a typical representative of a semantic web ontology.



## 4.1 Problem and Requirements Statement

In [chapter 3](#), we encountered several structures whose semantics could not be sufficiently formalized in *SRIQ*:

**Multilingual/Multi-Logic Groups of Mathematical Properties:** An OMDoc statement may have at most one informal (*CMP*) and one formal property (*FMP*). However, each of them may be given in a number of different languages or logics, respectively. This can be formalized in FOL, as demonstrated in cf. [section 3.2.2.3](#).

**Inheritance of Arbitrary Metadata from Referenced Entities** requires quantification over properties and therefore at least second-order logic, as pointed out in [section 3.4.2](#).

**Applicability of Problem and Solution Types** to types of mathematical knowledge items requires quantification over classes and therefore once more second-order logic, as pointed out in [section 3.6.2](#).

Further cases that have not yet been formalized include strong static dependencies on notation definitions (cf. [section 3.2.2.6](#)) and inheritance of missing metadata fields (cf. [section 3.4.2](#)). Additionally, we have encountered cases of aligning ontologies with great structural differences – the versioning ontologies discussed in [section 3.4.4.2](#) – that also requires a logic more expressive than *SRIQ*.

The mere fact that the formalization of *some* relevant structural aspects requires first- or even second-order logic does, of course, not imply that the complete OMDoc ontology should be formalized in such a logic. Limited expressivity was a deliberate design goal for description logics, as decidability is a prerequisite for web scalability. Web-scalable reasoning tasks, performed by RDFS or OWL reasoners, will remain an important application of the OMDoc ontology. Therefore, the desired ontology language should allow for formalizing everything that can be formalized in an OWL-compatible way. It should just, *additionally*, allow for formalizing the rest as well, for the following reasons: (i) Reasoners for more expressive logics exist, and complex knowledge management tasks might require applying them to a collection of mathematical knowledge. (ii) Developers of specialized MKM applications, which implement certain reasoning tasks without employing a general-purpose reasoner, need a complete and unambiguous documentation of those structural aspects they are interested in. A natural language developer's manual is more prone to incompleteness and ambiguity than a formal ontology, as can be seen, e.g., from the OMDoc 1.2 specification (cited in [section 2.4.4.1](#)).

Formalizing the interaction of structural dimensions and aligning ontologies in a comprehensive and comprehensible way not only requires a certain expressivity but also the possibility to express modularity and reuse. Again, this may not only serve automated reasoning purposes, but also provide documentation for developers.

Finally, there is proper, natural language documentation. While a formalization might be less ambiguous than a natural language explanation, the latter helps human users of an ontology: Developers of ontology-based applications need to understand the intentions the domain experts and ontology engineers had when conceptualizing a domain and formalizing it in an ontology. They need to understand what sections of the domain an ontology covers, and what the rationales

for certain design decisions were. Authors who want to annotate documents with concepts from an ontology, or ontology engineers who want to reuse concepts from an ontology or participate in the development of a large, modular ontology<sup>1</sup> have similar requirements on comprehensibility. We opt for embedding rich documentation into ontologies, most desirably even for literate programming (cf. [section 1.4.1](#)), as that makes the formalization and its documentation one unit of knowledge management – as argued for metadata in [section 2.1.1.3](#) –, and as it gives developers working on the formalization quicker access to the documentation.

Embedded documentation not only targets developers; it may also target end users, by the following argument: Publishing mathematical knowledge as linked data, as explained in [section 6.4](#), makes it reusable for mashups. Where such mashups want to provide their user with background knowledge about the mathematical structures in the linked dataset they use, for example for explaining what it means for a theory to have a meta-theory, their first stop for possibly finding such information is, according to the “follow your nose” policy common for linked data clients, the ontology. Thus, the ontology should have embedded documentation or at least fine-grained links to external documentation.

Our requirements for an ontology language can be summarized as follows:

**Logical Heterogeneity** *MUST* be supported. Those subsets of an ontology that can be handled by a reasoner *MUST* be expressible in a way that allows for extracting them from the complete ontology and feeding them to a reasoner in a language that the latter understands.

**Modularity and Reuse** of [parts of] external ontologies *MUST* be expressible.

**Documentation** *MUST* be embeddable into ontologies. Support for documenting ontologies as a whole, axioms/rules, and entities is *REQUIRED*; support for documenting sub-ontologies and subterms and further literate programming facilities *SHOULD* be provided.

## 4.2 State of the Art

This section reviews state-of-the-art ontology languages w.r.t. the requirements stated above, mostly focusing on RDFS and OWL, the languages most commonly used on the Semantic Web.

### 4.2.1 Expressivity

The common experience that the complexity of the domain of interest exceeds the expressivity of the ontology language chosen for implementation is addressed in different ways with non-heterogeneous languages. In some cases, the original ontology is formalized in an expressive logic, but simplified subsets in less expressive logics are provided. This is, e.g., the case with DOLCE, as mentioned in [section 2.3.4](#). For developers working with the OWL-based DOLCE Lite, the first-order KIF implementation serves as a more exhaustive reference. Neither the KIF implementation nor the natural language manual is, however, closely interlinked with OWL implementation. In other cases, the intended model is not fully formalized but only partly captured in the ontology,

<sup>1</sup>Larger ontologies are often compromised to the point of unusability because modeling decisions made in one part of the ontology do not correspond to modeling decisions made – possibly by different engineers – in other parts (see, e.g., [\[KPH+08\]](#)).

accompanied by a prose description of the actual axiom. This is, e.g., the case with FOAF (cf. [section 3.5.1](#)). FOAF has been implemented in OWL, but the semantics intended for one of its properties, *foaf:membershipClass*, cannot be expressed in OWL. The property is used to state that all instances of a given class *C* (where agents can be groups, persons, or organizations) automatically become *foaf:members* of a given *foaf:Group* *g* – which facilitates the maintenance of that *foaf:Group*, as one no longer has to explicitly add members to it but merely has to look at the properties of the potential members. The FOAF specification gives the example of making all people having the same workplace homepage members of that organization. This combination of ABox and TBox reasoning is not trivially supported in OWL. For example, one cannot formalize a rule such as  $\forall x \forall C \forall g. x \in C \wedge \langle g, C \rangle \in \text{foaf:membershipClass} \rightarrow \langle g, x \rangle \in \text{foaf:member}$ .<sup>2</sup> Therefore, *foaf:membershipClass* is not formally described in the OWL implementation of FOAF, but an informal text in the specification explains how application developers can implement hand-crafted support for the missing inference step. However, the discussion following up on [\[Alfo7\]](#) proves the ambiguity of that explanation.

### 4.2.2 Modularity

An RDFS ontology can reference arbitrary concepts from external ontologies by URI. These concepts may optionally point to the vocabulary they belong to by *rdfs:isDefinedBy* [\[BGo4\]](#). OWL improves on structured reuse by allowing explicit imports of ontologies via the *owl:imports* declaration [\[MPSP09\]](#). This always imports a whole ontology; there is, for example, no possibility for *information hiding*, as known from modular programming languages. Imports are not yet widely used in common semantic web ontologies, such as those discussed in [chapter 3](#), and tools usually do not enforce their usage. This is expected to change with OWL 2, which has made certain small improvements to the semantics of imports [\[CGHM+08\]](#).

### 4.2.3 Documentation

Documentation is crucial in engineering – in software engineering as much as in ontology engineering<sup>3</sup> – but the documentation capabilities of common ontology languages are still severely limited. Exhaustive documentation is usually maintained separately from a formal ontology, only pointing to entities of the ontology. Some ontology languages, such as F-Logic [\[KLW95\]](#), only support completely unstructured comments, like most programming languages do. RDFS and OWL 1 support annotating all entities of an ontology (classes, properties, individuals), as well as the ontology as a whole, with metadata [\[BGo4; MH04\]](#). OWL 2 adds this possibility to annotate axioms to the core of the language [\[MPSP09\]](#), following a pattern that is very similar to RDF

<sup>2</sup>It has been pointed out that there is a way of realizing the functionality of *foaf:membershipClass* within the scope of OWL by stating  $C \sqsubseteq \text{foaf:member}^{-1}:g$  [\[Alfo7\]](#). This is, however, not quite intuitive for the average author using FOAF, and *foaf:membershipClass* still has not been deprecated. Therefore, we continue to use it as an example.

<sup>3</sup>An informal Google search made in July 2009 showed, however, that ontology engineering still has to catch up. Six of the ten top hits for “software documentation” (without quotes) dealt with the process of documenting software, mostly providing guidelines. Four results led to documentation of concrete software products, two of them to auto-generated API documentation. Nine of the ten top hits for “ontology documentation” led to documentations of concrete ontologies, three of which had been auto-generated from the ontology sources. Coverage of the process of documenting ontologies only started on the second result page, hit #11 being our own previous work.

reification, i.e. treating statements as resources of their own and giving them a URI. However, this enhancement is fairly new and not yet supported by tools. Secondly, most tools restrict the value space of annotations to strings.

Practical language and tool support for annotating other subjects, such as subsets or -sections of an ontology or subterms of axioms, and for complex annotation texts or annotations inter-linked with ontology entities is rare to non-existent. Some semantic wikis, such as IkeWiki (cf. [section 9.3.1](#)), support ontology editing in a way that every entity is described by a text document, in which the links to other entities (e.g. superclasses) are embedded and surrounded by informal documentation. Named graphs extend the RDF data model by the possibility to assign a URI to any RDF subgraph [[CHB+05](#)] – e.g. a group of related axioms in an OWL ontology –, thus enabling them to be documented. The Protégé ontology editor supports named graphs via a plugin [[CSH+07](#)]; however, named graphs have mainly been explored for providing trust and provenance information so far, less so for documentation. Provenance information ranges from simple Dublin Core metadata expressing “who said what and when” to justifications of how an inferred statement has been established, i.e. what existing statements and axioms/rules have been used [[DSS+09](#)].

The RDF data model also allows for XML literals [[Beco4b](#)], which could, in principle, be used for complex annotations linking to ontology entities, but no tool support is known for this to date. RDFa would be an alternative for embedding RDF-based ontologies into HTML, turning the latter into a semantic markup language similar to the above-mentioned approach of some semantic wikis. RDFa has mainly been used for ABox knowledge so far; we are only aware of one application for TBox knowledge: Ontology Online is a web site for browsing and querying OWL and RDFS ontologies [[Deco7](#)]. Every page visualizes one entity of an ontology, with the original OWL or RDFS embedded as RDFa annotations – however, XHTML+RDFa is not used as a format for *authoring* the ontologies and giving them a richer documentation than one could provide in OWL or RDFS.

Finally, there is partial tool support for generating manuals from documented RDFS/OWL ontologies. Both Protégé and the NeOn toolkit [[Neo](#)] use OWLDoc [[Owla](#)] to generate an HTML view of an OWL ontology within the limited documentation capabilities of OWL. The specgen tool, originally developed for the FOAF ontology (cf. [sections 3.5.1](#) and [4.5](#)), works around these limitations: Besides the OWL implementation of an ontology, it additionally processes one external HTML fragment file per ontology entity and one global HTML template [[FBS](#)].

### 4.3 Implementing and Documenting Heterogeneous Ontologies in OMDoc

In MKM, tensions between high expressivity desired by authors and decidability or even tractability required for web-scalable automated inference are well known. Mathematical knowledge has traditionally been recorded and communicated in documents, which serve as their own documentation. We argue that OMDoc offers the features desired for ontology languages: (i) It is not committed to a particular logical foundation but can integrate any desired logic, thus supporting heterogeneous formalization. Therefore, one can combine axioms formalized in logics with favorable computational properties, such as *SR<sub>OTQ</sub>*, with formalizations in more expressive logics. Reusable OMDoc implementations of a large number of logics are available, including

various variants of first-order, higher-order, modal, and description logic [KMR]. (ii) It makes modularity explicit by supporting theory imports and views, thus allowing for formalizing reuse and interdependency. (iii) Finally, its wide range of formality degrees supported and its literate programming facilities allow for closely interspersing formalizations with their documentation. This section demonstrates how semantic web ontologies can be (re)implemented in OMDoc in a way that takes advantage of OMDoc's capabilities while retaining compatibility with existing ontology languages.

#### 4.3.1 Correspondences between OMDoc and Semantic Web Ontology Languages

One can easily identify the following correspondences between semantic web ontology languages and OMDoc: **Classes**, **Properties**, and **Individuals** correspond to objects or symbols. **Axioms** and **Rules** correspond to statements, as they state properties of resources. Semantic web ontologies usually do not distinguish proper axioms from facts derived from them. OMDoc, following the little theories approach (cf. [section 2.1.2](#)), allows for modeling this distinction and thus reducing theories to their core, while still enabling authors to document selected logical consequences of this core within the same theory. **Ontologies** correspond to theories, whose meta-theory is then usually a decidable subset of FOL. Both are often designed modularly and import other ontologies or theories. Both entities of an ontology and symbols of an OMDoc theory are identified by URIs within the namespace defined by the whole theory/ontology.

OMDoc is, as the existing semantic web ontology languages, based on web standards such as URIs and XML and allows for formalizing any desired logical foundation. We can thus make use of the correspondences pointed out above and model ontologies in OMDoc – provided that we overcome certain obstacles, which the following subsections address: (i) Since OMDoc is uncommitted to a *particular* logical foundation, it does not have a built-in understanding of the RDF(S) and OWL syntax and semantics (cf. [sections 2.3.3.1](#) and [2.3.4](#)). Therefore, these foundations – at least their vocabularies – have to be modeled as OMDoc theories first, which will then form the meta-theories of concrete ontologies. (ii) OMDoc theories can import other theories for a modular design. However, due to different URI formats, they cannot directly reference symbols from reused existing semantic web ontologies. Therefore, we have to specify an import syntax and semantics. (iii) OMDoc itself is not supported by any RDFS or DL reasoner. Therefore, we need to provide a translation of OMDoc theories into the standard encodings of semantic web ontologies, as has been done earlier for formalized mathematics languages (cf. [section 2.4.4.1](#)).

As we wanted to support RDFS and OWL at the same time and already had a running OMDoc→RDF translation at hand (cf. [sections 3.7](#) and [8.1](#)), we decided to accomplish [step \(i\)](#) via the RDF serialization of OWL [PSMo9], so that [step \(iii\)](#) could be accomplished by reusing our existing translation. However, this does not preclude us from implementing another theory, whose symbols correspond to the OWL functional-style syntax<sup>4</sup> [MPSPo9], and modeling views between both implementations, i.e. formalizing the mapping of OWL ontologies to RDF graphs [PSMo9] in OMDoc.

---

<sup>4</sup>In fact, part of that work has recently been done by FIGEN FÜSUN HOROZAL, who has implemented views from various description logics to OWL and from OWL to FOL in OMDoc [KMR].

### 4.3.2 Knowledge Representation

As a foundation for expressing semantic web ontologies in OMDoc, we wrote theories for RDF, RDFS, and OWL – covering the vocabulary compatible to OWL 1 so far –, which declare as symbols all classes, properties, and individuals of these languages. We have not yet formalized the full *semantics* of RDF, RDFS, and OWL in OMDoc.

An ontology is then written as follows: Classes, properties, and individuals are declared as *symbols* with a *type*<sup>5</sup>. Property types are modeled as compound types, e.g. `@(owl#ObjectProperty, oo#ProvenAssertion, oo#Proof)` constructed from the actual property type, plus domain and range. Class definitions like `oo:ProvenAssertion = oo:Assertion  $\sqcap$   $\exists$ provedBy.Proof`<sup>6</sup> (cf. [section 3.2.2.2](#)) are given as OMDoc *definitions*, as shown in [listing 4.1](#)<sup>7</sup>. This is a machine-oriented representation that a user would not usually see, but which would render as three lines in the human-comprehensible documentation (compare [figure 4.1](#) on page 156) and be edited using a dedicated formula editor (cf. [figure 6.1](#) on page 190). We make use of OpenMath’s support for *n*-ary structures, instead of breaking all class and property expressions down into triples. Furthermore, the sample definition showcases a maximum amount of literate programming: *term* annotates references to symbols (“technical terms”) in natural language, whereas *phrases* of natural language can be linked to corresponding subterms of mathematical objects.

All other statements, for which we have not introduced specific syntactic sugar, can be expressed as OMDoc *axioms* in such a way that a property is applied to two arguments: a subject and an object. This is the most direct way of representing RDF in OMDoc but does not take advantage of the higher expressivity of OMDoc. Note the possibility to annotate redundant axioms – as introduced in [section 4.3.1](#) – as *theorems*, which can then be proven on the OMDoc level, using other axioms of the same ontology plus the inference rules of the respective ontology language, as represented in the RDF, RDFS, and OWL theories.

### 4.3.3 Connecting OMDoc and Semantic Web URIs

OMDoc and RDF – and hence RDFS and OWL – have different ways of giving URIs to symbols. RDF-based ontologies have a namespace URI, which is usually considered to be the URI of the ontology, and all entities within the ontologies have local names (cf. [section 2.3.3.1](#)). An absolute URI is formed by concatenating the namespace URI and a local name. OMDoc, on the other hand, addresses symbols by a triple of *cdbase* (theory graph), *cd* (theory) and [local] *name* (cf. [section 2.4.4.1](#)).

Two situations where this difference needs to be overcome are (i) rewriting an existing semantic web ontology in OMDoc, e.g. for the purpose of documenting it or making its modular structure

<sup>5</sup>OMDoc has a foundationally unconstrained infrastructure for type systems: Symbols, actually all mathematical objects, can be associated with types that are objects themselves. The particular choice of types is only governed by the available theories. Here we define types as part of the RDF, RDFS, and OWL theories. Note that we mainly explored this approach as a *syntactic* possibility for writing down the common axiom pattern “individual is instance of class” more concisely. Type theory discerns terms and types and therefore is actually not adequate for formalizing the RDFS theory, in which `rdfs:Class rdfs:type rdfs:Class` is a theorem. The direct semantics of OWL, with its separation of individuals and classes, does not exhibit such problems.

<sup>6</sup>The actual axiom declares `oo:ProvenAssertion` as a subclass. Here, we assume equality for the sake of a nicer example.

<sup>7</sup>This listing and the following one make use of the RDFa extension of OMDoc’s syntax introduced in [chapter 5](#).



Listing 4.1: Simplified excerpt from the OMDoc implementation of the OMDoc ontology: class definition and documentation

```

<theory xml:id="oo">
  <link rel="oo:vocab" href="http://omdoc.org/ontology#"/>    <!-- (explained below) -->
  <!-- meta-theories -->
  <imports xml:id="owl" from="owl.omdoc#owl"/>                <!-- OWL -->
  <imports xml:id="plleq" from="plleq.omdoc#plleq"/>           <!-- FOL (with equality) -->
  <omtext type="introduction">
    <CMP>Most of <phrase verbalizes="#oo">the OMDoc ontology</phrase> can be
    <phrase verbalizes="#owl">formalized in the logic of OWL</phrase>,
    but <phrase verbalizes="#plleq">we also need FOL with
    <term cd="indeq" name="eq">equality of individuals</term></phrase>.
    We formalized the concept of <term cd="oo" name="ProvenAssertion">
    a proven assertion</term>, and many others.</CMP>
  </omtext>
  <symbol name="ProvenAssertion" xml:id="ProvenAssertion.sym">
    <meta property="rdfs:comment">an assertion that has been proven</meta>
    <type>
      <OMOBJ xmlns="http://www.openmath.org/OpenMath">
        <OMS cd="owl" name="Class"/>
      </OMOBJ>
    </type>
  </symbol>
  <!-- similar declaration of provedBy omitted -->
  <definition for="ProvenAssertion" type="simple">
    <CMP><term cd="oo" name="ProvenAssertion" role="definiendum">A proven assertion
    </term> is <term cd="oo" name="Assertion" role="definiens">
    <phrase verbalizes="#A"><phrase verbalizes="#A2">an assertion</phrase>
    <phrase verbalizes="#A1">that</phrase><phrase verbalizes="#A3">
    <phrase verbalizes="#A3B">is proved by</phrase>
    <phrase verbalizes="#A3C"><phrase verbalizes="#A3C1">a</phrase>
    <phrase verbalizes="#A3C2">proof</phrase></phrase></phrase></phrase>
    </term>.</CMP>
    <OMOBJ xmlns="http://www.openmath.org/OpenMath">
      <OMA id="A">
        <OMS id="A1" cd="owl" name="intersectionOf"/>
        <OMS id="A2" cd="oo" name="Assertion"/>
        <OMA id="A3">
          <OMS id="A3A" cd="owl" name="Restriction"/>
          <OMS id="A3B" cd="owl" name="provedBy"/>
          <OMA id="A3C">
            <OMS id="A3C1" cd="owl" name="someValuesFrom"/>
            <OMS id="A3C2" cd="oo" name="Proof"/>
          </OMA>
        </OMA>
      </OMA>
    </OMOBJ>
  </definition>
</theory>

```



Listing 4.2: An OMDoc ontology with a semantic web namespace URI

```

<theory xml:id="foaf">
  <metadata>
    <link rel="oo:vocab" href="http://xmlns.com/foaf/0.1/" />
    <meta property="dc:title">Friend of a Friend (FOAF) vocabulary</meta>
  </metadata>
  <!-- imported theories and ontologies omitted -->
  <symbol name="Agent"><!-- declaration omitted --></symbol>
  <!-- ... -->
</theory>

```

more explicit, and (ii) reusing symbols from an existing semantic web ontology in OMDoc. In order to have OMDoc ontologies generate correct RDF-style URIs on translation, we allow for attaching the vocabulary URI of the original ontology to a theory via the special *oo:vocab* annotation (named in the style of RDFa; cf. [section 2.3.3.4](#)), which is recognized by the OMDoc→OWL translation, whose implementation is explained in [section 8.1.3](#). [Listing 4.2](#) shows how this would be done for FOAF. The annotation makes sure that the OMDoc→OWL translation gives the *Agent* class its correct URI, i.e. <http://xmlns.com/foaf/0.1/Agent>. We can create a basic OMDoc implementation of a semantic web ontology simply by providing a suitable *oo:vocab* metadata field and leaving its augmentation by symbol declarations, definitions, axioms, etc., to the future. This is a low-cost way for starting the OMDoc implementation of an ontology, which does not preclude making use of OMDoc's possibilities for documentation and expressive knowledge representation later. Thus, we have a suitable migration path from web ontologies to OMDoc.

A nice side-effect of importing ontologies as theories is that they can now also be used to document subterms of OpenMath expressions. [Section 2.5.1](#) gives an example for annotating a Content MathML expression, referencing the annotation property via *annotation/@definitionURL*. By default, in OpenMath, one can only use properties with a hash URI, as attribution keys are symbols, whose URIs have to match the *cdbase/cd#name* format, or otherwise introduce a non-standard string→URI constructor, as remarked in [section 2.5.1](#). Now that we can wrap ontologies into OMDoc theories, we can use the symbols from these theories, e.g. `<OMS cd="dct" name="description"/>`, as attribution keys for annotating subterms.

A related question, whose answer we leave to future work, is whether we should now also formally import ontologies used as vocabularies for RDFa annotations. Such imports would not affect the processing of these annotations according to the rules of RDFa, but they might facilitate an in-depth study of interrelations of those dimensions of mathematical knowledge that OMDoc can natively express with additional dimensions for which we need RDFa – particularly in cases where these additional dimensions of knowledge are conceived as (primary) data rather than metadata; compare the discussion in [section 2.1.7.7](#).

#### 4.3.4 Documentation and Presentation

OMDoc’s elaborate presentation framework, whose implementation is summarized in [appendix C.1.2](#), is well suited for generating human-comprehensible documentation from ontologies. For every mathematical symbol, one or more *notations* can be defined (cf. [section 2.4.5](#)) – compare, e.g., our initial OWL example in the German DL notation ( $\text{ProvenAssertion} = \text{oo:Assertion} \sqcap \exists \text{provedBy.Proof}$ ) vs. the Manchester syntax [[HPS09](#)]:

**Class:** ProvenAssertion

**EquivalentTo:** Assertion **that** provedBy **some** Proof

A default notation is usually provided by the author of a theory; we have done that for our RDF, RDFS, and OWL theories. But users can also author their own ones to customize the presentation to their preferences. The most suitable notation for presenting a document to a specific user is then selected in a context-sensitive way. The XHTML+MathML output is semantically annotated, as explained in [section 6.4.2.2](#), which allows for enriching it by interactive services that help to enhance comprehensibility, as explained in [chapter 7](#). For example, a reader who does not know the  $\sqcap$  symbol in our sample formula can click on it to read its definition in the *owl* OMDoc theory that declares (and documents!) the symbol *owl#intersectionOf*. Embedded documentation can be given as metadata (cf. [section 5.2](#)), which can be attached to any markup element. Textbook or literate programming style is also possible, as exemplified in [listing 4.1](#).

### 4.4 Implementation of the OMDoc Ontology

Using OMDoc as an ontology language allowed us to fully implement the OMDoc ontology for logical/functional structures of mathematical knowledge and prepares us for formalizing its interaction with ontologies for other structural dimensions as well as aligning complex ontologies.

Those aspects of OMDoc that could be formalized in OWL have initially been implemented as an OWL 2 ontology. The full expressivity of OWL 2 DL – or *SR<sub>Q</sub>IQ*, respectively, – has been used to capture the semantics of OMDoc as faithfully as possible. The implementation has then been continued in OMDoc, and the formalizations summarized in [section 4.1](#) have been added, using more expressive logics. The ontologies reused by the OMDoc ontology have not completely been reimplemented in OMDoc so far; instead, we have created wrappers as shown in [listing 4.2](#), leaving the explicit formalization to future work. [Appendix B.1](#) provides information about further technical details of the implementation.

For more efficient reasoning, “downgrading” the OWL subset of the OMDoc ontology to one of the OWL profiles [[MCGH+09](#)] or modeling it in multiple layers of complexity should be considered. For example, MICHAEL DUMONTIER and NATALIA VILLANUEVA-ROSALES suggested (i) a simple taxonomy of classes and properties, (ii) a layer of complex axioms and restrictions, and (iii) specific restrictions, e.g. for validity w.r.t. a particular application [[DVR07](#)]. None of these has been done yet. However, the reasoners that have so far been used with the OMDoc ontology only consider subsets of it in any case, as discussed in [section 6.5.2](#), which makes them degrade gracefully on the complete OWL 2 DL subset of the ontology. Fine-tuned downgradings to specific profiles are still desirable, as they potentially retain more of the intended semantics. The closest subprofile of OWL 2 that the OMDoc ontology matches is RL – provided that the union classes

that occur as domains or ranges of some properties (cf. table B.2) are replaced, most reasonably by their closest common superclass. For example, one could declare a class *Exemplifiable* instead of *Symbol*  $\sqcup$  *Definition*  $\sqcup$  *Axiom*  $\sqcup$  *Assertion* as the range of *exemplifies*, where each of the classes *Symbol*, *Definition*, *Axiom*, and *Assertion* would be declared a subclass of *Exemplifiable*. This change does not affect most typical queries over RDF extracted from OMDoc, such as “give me all examples for statements in this theory”. Suitable additional disjointness axioms assumed, it does not affect certain ontology-based validation tasks either (cf. section 6.3.3.1).

## 4.5 Case Study: Reimplementing FOAF in OMDoc

We have so far seen that OMDoc is a suitable language for implementing the OMDoc ontology. However, we argue that any semantic web ontology can benefit from OMDoc’s support for heterogeneity, modularity, and rich integrated documentation. This section reports on a small-scale case study that we have conducted in order to validate this claim.

We have explored the increased expressivity and documentation possibilities by reimplementing a relevant subset of the FOAF ontology (cf. section 3.5.1 and [BM10]) in OMDoc.<sup>8</sup> We chose FOAF for the following reasons: (i) It is widely used, (ii) we will also need it for representing users and organizations who produce or consume mathematical knowledge, (iii) it comes with a comprehensive HTML documentation, (iv) it makes use of more OWL constructs than other comparable ontologies – disjoint classes and inverse properties in particular –, and (v) it even tries to capture concepts that exceed DL, as pointed out in section 4.2.1. A human-comprehensible rendering of the OMDoc reimplementation of FOAF is shown in figure 4.1. From studying the OWL implementation and the specification of FOAF, we noticed the following problems, which we were able to solve using OMDoc:

1. FOAF references entities from other ontologies (DCMES, WordNet, Geo Positioning, etc.), but it does not *import* them. OMDoc tools, such as the MMT system [Rab; RK11] can identify imports missing in an OMDoc ontology, and the implementation of our OMDoc  $\rightarrow$  OWL translation adds them to the resulting OWL ontology.
2. The source code contains notes for developers as XML comments. In the OMDoc version of FOAF, we were instead able to create informal text sections (*omtext*) for them. Other XML comments divide the ontology into sections, such as “naming properties”. In OMDoc, we were able to model document sections without disrupting the logical structure of the ontology.
3. Some of these comments were attached to individual triples, e.g. *foaf:mbox\_sha1sum* *rdf:type* *owl:DatatypeProperty*. Thanks to literate programming in OMDoc, we could precisely add them as informal comments (CMPs) to the respective OMDoc statements.<sup>9</sup>
4. The following properties are inverses of each other: *foaf:maker* = *foaf:made*<sup>−</sup>, *foaf:depiction* = *foaf:depiction*<sup>−</sup>, *foaf:topic* = *foaf:page*<sup>−</sup>, and *foaf:primaryTopic* = *foaf:isPrimaryTopicOf*<sup>−</sup>.

<sup>8</sup>We have not actually reimplemented *all* of FOAF, but enough to cover all different kinds of axioms and documentation found in the OWL implementation; the rest would be done completely analogously.

<sup>9</sup>The same would have been possible with OWL 2 axiom annotations.

## Friend of a Friend (FOAF) vocabulary

imports from: [wordnet](#), [dc](#), [owl](#), [quantl](#), [logicl](#)

**AXIOM:**  
The foaf:Person class is a sub-class of the foaf:Agent class, since all people are considered 'agents' in FOAF.  
 $\text{Person} \sqsubseteq \text{Agent}$

**AXIOM:**  $\text{Person} \sqcap \text{Organization} = \perp$

**CONCEPT: made**  
The foaf:made property relates a foaf:Agent to something foaf:made by it.

**TYPE:**  
ObjectProperty

**TYPE:**  
(Agent, Thing)

**AXIOM:**  $\text{made} = \text{maker}^{-}$

**LEMMA:**  $\text{maker} = \text{made}^{-}$

**PROOF:** 1. We know that  $\text{made} = \text{maker}^{-}$ .

- Interpreted using the model-theoretic semantics, this means that  
 $\text{made}^I = (\text{maker}^{-})^I = (\text{maker}^I)^{-}$ .
- Now we apply the inverse on both sides, eliminate double inverses, and obtain  
 $(\text{made}^I)^{-} = ((\text{maker}^I)^{-})^{-} = \text{maker}^I$
- This is just the interpretation of  $\text{maker} = \text{made}^{-}$ , which we had to prove.

**CONCEPT: membershipClass**  
The foaf:membershipClass property relates a foaf:Group to an RDF class representing a sub-class of foaf:Agent whose instances are all the agents that are a foaf:member of the foaf:Group. See foaf:Group for details and examples.

**AXIOM:**  $\forall m, g, C. (m :_{\text{type}} C \wedge \text{membershipClass}(g, C) \Rightarrow g \exists_{\text{member}} m)$

## Friend of a Friend (FOAF) vocabulary

imports from: [wordnet](#), [dc](#), [owl](#), [quantl](#), [logicl](#)

**AXIOM:**  
The foaf:Person class is a sub-class of the foaf:Agent class, since all people are considered 'agents' in FOAF.  
 $\text{Person} \sqsubseteq \text{Agent}$

**AXIOM:**  $\text{Person} \sqcap \text{Organization} = \perp$

**CONCEPT: made**  
The foaf:made property relates a foaf:Agent to something foaf:made by it.

**TYPE:**  
ObjectProperty(Agent, Thing)

**AXIOM:**  $\text{made} = \text{maker}^{-}$

**LEMMA:**  $\text{maker} = \text{made}^{-}$

**PROOF:** collapse

- We know that  $\text{made} = \text{maker}^{-}$ .
- Interpreted using the model-theoretic semantics, this means that  
 $\text{made}^I = (\text{maker}^{-})^I = (\text{maker}^I)^{-}$ .
- Now we apply the inverse on both sides, eliminate double inverses, and obtain  
 $(\text{made}^I)^{-} = ((\text{maker}^I)^{-})^{-} = \text{maker}^I$
- This is just the interpretation of  $\text{maker} = \text{made}^{-}$ , which we had to prove.

**CONCEPT: membershipClass**  
The foaf:membershipClass property relates a foaf:Group to an RDF class representing a sub-class of foaf:Agent whose instances are all the agents that are a foaf:member of the foaf:Group. See foaf:Group for details and examples.

**AXIOM:**  $\forall m, g, C. (m :_{\text{type}} C \wedge \text{membershipClass}(g, C) \Rightarrow g \exists_{\text{member}} m)$

German DL notation  
German DL notation  
Functional-style syntax  
Manchester syntax  
Hide formulæ

Definition (disjointWith)
This constructor guarantees that an individual that is a member of one class cannot simultaneously be an instance of a specified other class. An axiom  $A \sqcap B = \perp$  is equivalent to the following axiom:  $A \sqsubseteq \neg B$

collapse

Export OWL as RDF/XML

Figure 4.1: A subset of the FOAF ontology, reimplemented and documented in OMDoc.

Top: static version rendered by JOMDoc (cf. section 6.4.2.8)

Bottom: mockup of an interactively enriched rendering (cf. chapter 7)

While for each  $p = q^-$ , an OWL reasoner can infer  $q = p^-$  from its built-in DL axioms, FOAF redundantly declares each inverse relationship for both participating properties for the purpose of documentation. OMDoc allows for making the difference explicit: For any of the above  $p, q$  property pairs, we (arbitrarily) picked one  $p$  and stated  $p = q^-$  as an axiom, but  $q = p^-$  as an *assertion* that can (provably) be derived from the axiom and the semantics of *owl:inverseOf*<sup>10</sup>, as shown in figure 4.1. Domain and range of inverse properties can be handled similarly.

5. We were able to express the non-OWL semantics of *foaf:membershipClass* mentioned above. We chose the second-order logic representation shown in figure 4.1.
6. The correspondence of *foaf:maker* to *dc:creator* is only defined in prose. The specification suggests using *foaf:maker* whenever the agent who created something is known by URI, and to use the less semantic *dc:creator*, which neither has range nor domain declared, when the creator is only known by a string. Then, it also informally states a rule that the *foaf:name* or *rdfs:label* of the *foaf:maker* of something is the same as the *dc:creator* of that thing. The rule can be captured by a FOL expression in OMDoc, or alternatively by an OWL 2 property chain inclusion [MPSP09]. The notion that *foaf:maker* is similar to *dc:creator* but has a stronger semantics can be captured by having the FOAF theory import the DCMES theory and defining a *view* on DCMES, namely a morphism that maps *dc:creator* to *foaf:maker*.
7. Finally, we were able to include the informal sections and descriptions of the FOAF specification [BM10] right into the ontology document. This allows for a unified management of the formal specification and its informal explanation, including the introductory chapters and the change log, in a single, coherent document, of which both OWL and the XHTML shown in figure 4.1 can be generated. The original FOAF specification is generated with the specgen script mentioned in section 4.2.3.

The enhanced expressivity of the OMDoc reimplementing of FOAF comes at the expense of a much higher verbosity. While in RDF one can easily attach another axiom to a class (stating, e.g., a subclass relationship or disjointness), most of these triples have to be represented as individual axioms in OMDoc – as is the case in the functional-style syntax of OWL 2 –, unless there is an intuitive way of capturing their semantics in a type declaration syntax. Better annotation tools and shorthand input syntaxes could help (cf. section 6.2), but there is also a mathematical approach to improving this<sup>11</sup>: One could add additional axioms to the OMDoc theory for OWL, which introduce operators for shorthand notations (such as pairwise disjointness of a whole set of classes) that imply multiple atomic statements<sup>12</sup> – but then all these axioms would have to be *applied* before

<sup>10</sup> A proof is only *required* in OMDoc if one wants to do automated theorem proving.

<sup>11</sup> By technical coincidence, another way is possible: embedding OWL statements as RDFa metadata into OMDoc. We discourage this, however, as abuses *metadata* (= data about data) for information that should actually be treated as proper data, and as no other OMDoc-aware tool except our own OMDoc→OWL translator would be able to process such annotations.

<sup>12</sup> A shorthand syntax for pairwise disjointness has been introduced in OWL 2 [MPSP09], but, in other such cases, an ontology engineer does not enjoy the freedom of introducing additional shorthands as needed.

generating OWL from OMDoc. This can be done by supporting lambda calculus at the meta level and  $\beta$ -reducing all OMDoc axioms before generating OWL.

## 4.6 Related Work

### 4.6.1 Implementing OWL Ontologies in OMDoc

MATTHIAS BRÖCHELER has pursued an approach to implementing OWL ontologies in OMDoc, which is similar to the one described in sections 4.3.1 and 4.3.2 [Brö07]. The goal was different from ours: formalizing mathematical knowledge in a language commonly used for that task (i.e. OMDoc), while using a logic with favorable computational properties and good integration into the Semantic Web (*SHIN*, a DL expressible in OWL). BRÖCHELER's goal was to enable mathematicians to formalize mathematical concepts in DL as intuitively as possible (cf. the review of his work in section 3.8); therefore, he introduced a new, non-standard vocabulary for the constructs of the *SHIN* logic. Our goal is to enable ontology engineers to formalize and document their existing ontologies in OMDoc; therefore, to keep the migration effort as low as possible, we reused the vocabulary of the RDF serialization of OWL and made OMDoc and RDF URIs compatible with each other. Where BRÖCHELER initially limited his vocabulary to the *SHIN* logic, arguing that it is sufficient to capture mathematical knowledge (when capturing it in DL at all), we aim at supporting all constructs of OWL<sup>13</sup>, as our target audience are authors of OWL ontologies, and, in particular, our own ontologies make heavy use of OWL.

### 4.6.2 Heterogeneous Formalization

OMDoc supports the formalization of heterogeneous ontologies. Section 2.4.4.1 has briefly mentioned CASL, which has been extended to support heterogeneous formalization (including OWL) in the Hets environment [KLM+08]. However, in contrast to OMDoc, CASL is a purely formal language and does not allow for documenting heterogeneous ontologies.

### 4.6.3 Integrated Ontology Documentation

An alternative to maintaining an ontology and its documentation in the same document is provided by the PDFTab Protégé extension, which closely integrates a semantic annotation plugin to the Adobe Acrobat PDF editor into the Protégé ontology editor and therefore supports concurrent evolution of an ontology and annotations in a document, which can provide documentation for the ontology [Erio7]. The document itself, however, cannot be evolved concurrently, as the content of a PDF document is not intended to be changed.

## 4.7 Conclusion and Future Work

We have presented OMDoc as an expressive language for the formalization and fine-grained documentation of heterogeneous and modular ontologies. OMDoc not only allowed us to implement

---

<sup>13</sup>OWL 1 DL, corresponding to the *SHOIN* logic, and parts of OWL 2 are supported so far; full OWL 2 DL support, corresponding to the *SROIQ* logic, remains to be implemented.



the OMDoc ontology from [section 3.2.2](#) in its full expressivity, but it has also proven a suitable language for enhancing the formalization, modularity, and documentation of a typical semantic web ontology. We have demonstrated that in a case study with FOAF, where OMDoc allowed for formally representing those concepts that cannot be modeled in OWL, and for integrating all available documentation into the ontology. Having addressed the shortcomings of contemporary ontology languages w.r.t. heterogeneity, modularity, and, most importantly, integrated documentation in an integrated fashion, we have made a contribution to ontology engineering in general.

The possibility to generate both a machine-processable OWL ontology (cf. [section 8.1.3](#) for how that is done) and a human-readable manual from the same OMDoc source further facilitates the integrated maintenance of an ontology and its documentation. By the ability to integrate existing and envisaged interactive services into the rendered manual, as the mockup in the lower half of [figure 4.1](#) demonstrates, the manual can be further tailored to meet the requirements of its readers.

Besides proving that OMDoc can express all information required for an ontology, the *practical* benefits of doing so need to be assessed. A logical next step would be publishing the FOAF documentation generated from OMDoc online and gathering feedback from the developers and users of FOAF. The documentation should be enriched with all interactive services that are easily available. (At the moment, these would be definition lookup and folding, as shown in the mockup in [figure 4.1](#)). The following questions should be considered:

- Do the authors consider a migration from OWL (in the case of FOAF serialized as RDF/XML) and HTML sources for the ontology and its documentation to OMDoc feasible, using e.g. our OWL→OMDoc translator (cf. [section 8.2](#))?
- Do the authors find it easier to generate the documentation from the OMDoc source, compared to the old workflow?
- Do users (e.g. developers of ontologies reusing FOAF, developers of FOAF software, or authors who annotate documents with FOAF) find the desired information more easily in the documentation generated from OMDoc?
- Do users understand the documentation more easily? In particular, does it help that axioms can now optionally be displayed as mathematical objects – instead of being omitted from the documentation –, and that facts exceeding the complexity of OWL can now also be expressed formally?

While we started exploring OMDoc-based ontology engineering with OWL and the FOAF ontology, the approach is not limited to OWL. OMDoc can be used for representing and documenting ontologies in any other language as well, given that OMDoc theories and notation definitions for the symbol vocabulary of the respective underlying logics are implemented. In the KWARC research group, this has most recently been pursued for the SUO-KIF language (implementing a variant of FOL), in which the SUMO upper ontology [[PNLo2](#)] is represented [[Mis10](#)].

By representing ontologies in OMDoc, all services that we have developed for OMDoc and that [chapter 6](#) presents – particularly those for browsing, discussing, editing, validating, querying – become applicable to ontologies. We expect further benefits from specifying the process of designing “literate ontologies” as one strand of an ontology engineering methodology that covers



the full ontology lifecycle. In order to support literate design technically, relevant services would have to be integrated into an ontology development environment. One possibility is providing OMDoc plugins for ontology development environments such as Protégé [[Proc](#)] or the NeOn Toolkit [[Neo](#)]. [Section 9.6.4.1](#) discusses how our OMDoc-based SWiM wiki can be used for ontology development.

## Acknowledgments

The argumentation in favor of ontology documentation has been developed in collaboration with JOHN BATEMAN and MICHAEL KOHLHASE. The usage of OMDoc as a language for implementing and documenting ontologies was explored in collaboration with MICHAEL KOHLHASE [[LK09](#)], with advice from FLORIAN RABE and further inspiration from JOHN BATEMAN. SIARHEI KURLA helped to complete the OMDoc theories for RDF, RDFS, and OWL.

## Multi-Dimensional Metadata Markup

[Chapter 3](#) has presented ontologies for all structural dimensions of mathematical knowledge, and [chapter 4](#) has given us a language for extending and refining our supply of ontologies in a self-contained way. The vocabulary built into the OMDoc 1.2 markup language, however, is still restricted to logical/functional, presentational, rhetorical, and document structures, and few metadata vocabularies. Certain application scenarios demand representing additional information in a mathematical document, as we have, e.g., observed from a software engineering scenario, where information about the software process, the status of certification, and the project organization structure was maintained in documents in addition to the formal specification (cf. [section 2.1.7.7](#) and [\[KKL10a; KKL10b\]](#)). Similar requirements have arisen from authoring lecture notes and exercises in OMDoc (cf. [section 2.4.7.2](#)).<sup>1</sup>

With RDFa, a mechanism for representing knowledge in terms of arbitrary ontologies inside XML documents exists. This chapter presents an extension of OMDoc by RDFa. Following the considerations from [section 2.1.1.3](#), I generally subsume all dimensions of knowledge beyond the primary ones, which can be expressed natively in OMDoc, under the term “metadata”. After a detailed review of the metadata support of OMDoc 1.2 and pointing out its deficiencies, this chapter describes the integration of RDFa into OMDoc, particularly addressing the following issues: how RDFa annotations interact with the semantics of the surrounding OMDoc, and how the same expressivity as the OMDoc 1.2 metadata had can be achieved with RDFa and appropriate metadata vocabularies.

### 5.1 The Metadata Syntax of OMDoc 1.2 (State of the Art)

OMDoc supports metadata on almost every element on the document, theory, and statement levels [[Koho6b](#), chapter 12]. OMDoc 1.2 has hard-coded support for two metadata vocabularies, as exemplified in [listing 5.1](#): The OMDoc module DC comprises the DCMES with MARC relators and idiosyncratic versioning extensions, and the CC module covers ccREL. Each vocabulary resides in its own XML namespace separate from the OMDoc namespace. The DCMES has been

---

<sup>1</sup>Personal communication with MICHAEL KOHLHASE, July to September 2008. The original e-mail thread, not mentioning use cases, has been archived [[Koho8c](#)].

Listing 5.1: OMDoc 1.2 proof of Fermat's theorem, with a revision history of historical attempts

```

<proof xml:id="fermat-proof" for="#fermats-last-theorem"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:cc="http://creativecommons.org/ns">
  <metadata>
    <dc:title>Proof of Fermat's Last Theorem</dc:title>
    <dc:creator xml:id="fermat">Pierre de Fermat</dc:creator>
    <dc:contributor role="aut" xml:id="wiles">Andrew Wiles</dc:contributor>
    <dc:publisher role="edt" xml:id="kohlhase">Michael Kohlhase</dc:publisher>
    <dc:date who="#fermat" action="created">1637-06-13T00:00:00</dc:date>
    <!-- hundreds of other (incorrect) proofs omitted here -->
    <dc:date who="#wiles" action="updated">1995-05-01T00:00:00</dc:date>
    <dc:date who="#kohlhase" action="imported">2006-08-28T00:00:00</dc:date>
    <cc:license jurisdiction="de">
      <cc:permissions reproduction="permitted" distribution="permitted"
        derivative_works="permitted"/>
      <cc:requirements notice="required" attribution="required"/>
    </cc:license>
  </metadata>
  <derive xml:id="..."><!-- the first step of the proof --></derive>
  <!-- ... -->
</proof>

```

integrated according to the guidelines for implementing Dublin Core in XML [PJ03]; for the other vocabularies, comparable guidelines do not exist. Many other existing metadata vocabularies have been implemented in one or multiple XML schema languages. Such a schema could be added as a module to the RELAX NG schema of OMDoc 1.2. Its elements would have to reside in their own namespace and would have to be added to the content model of OMDoc's *metadata* element. For the ActiveMath e-learning system, OMDoc has been extended by additional metadata vocabularies (cf. section 2.4.4.5) – however, in a less scalable way, by adding all of their terms to the main OMDoc namespace [MAF+03; GUM+04]. Another notable extension of OMDoc by another structural dimension, the PhysML language mentioned in section 2.1.7.7, has been implemented similarly [HKS06].

The OMDoc 1.2 specification mentions the possibility of extracting metadata from OMDoc to RDF without going into details how that should be done [Koho6b, chapter 11.2]. In section 3.4.4, I have provided such a mapping but encountered two problems:

- Certain uses of MARC relators and ccREL licenses, which are not valid in these models or in their implementations as ontologies, are expressible in schema-valid OMDoc markup.
- OMDoc's idiosyncratic versioning vocabulary is underspecified and hard to map to any available versioning ontology.

These problems make it hard to implement full application support for OMDoc 1.2 metadata. ActiveMath is the only application that has supported a significant share of the vocabulary so far.

## 5.2 The new OMDoc+RDFa Metadata Framework

Given the need to represent additional dimensions of knowledge in OMDoc, and considering the particular deficiencies of the metadata syntax of OMDoc 1.2, we developed a new framework. General requirements for a language for representing multiple dimensions of mathematical knowledge have been established in [section 2.2](#); specific requirements in face of our particular commitment to the OMDoc language were as follows:

1. The set of metadata vocabularies **MUST** be extensible for future applications; there **SHOULD NOT** be a fixed set of vocabularies hard-coded into OMDoc’s XML schema.
2. The formal semantics of metadata vocabularies **MUST** be exposed to applications.
3. The new metadata **MUST** remain semantically backwards-compatible with OMDoc 1.2, i.e. it **MUST** support the DCMES and ccREL vocabularies and OMDoc’s extensions to DCMES.
4. In particular, it **MUST** support revision histories.

The following subsections explain how we have satisfied these requirements by (i) integrating RDFa into OMDoc ([section 5.2.1](#)) in a way that remains syntactically close to OMDoc 1.2, as not to disrupt existing authoring practices ([section 5.2.2](#)), while conforming to the processing rules of RDFa, in order to be immediately comprehensible to RDFa-aware software ([section 5.2.3](#)), and (ii) recommending a set of metadata vocabularies that provide authors at least with the same expressivity that OMDoc 1.2 had ([section 5.2.4](#)).

### 5.2.1 Integrating RDFa into OMDoc

A high-level justification for integrating RDFa into OMDoc has been given in [section 2.5.2](#): RDFa gives access to an unrestricted range of metadata vocabularies, and it gives all metadata an RDF semantics. Combined with the RDF semantics that we have already given to the semantic XML markup of OMDoc in [sections 3.2, 3.3 and 3.7](#), that means that all dimensions of knowledge can now be uniformly represented – and then queried or processed – in RDF. Thus, the RDFa integration satisfies the requirements [1](#) and [2](#) stated above. The decision for RDFa was further supported by the circumstance that the two metadata vocabularies that OMDoc had used so far had an RDF-compatible semantics. [Section 5.2.4](#) specifically addresses how the extension of OMDoc by RDFa satisfies requirements [3](#) and [4](#).

In view of the potential complexity of representations of additional dimensions of knowledge, we decided to give authors the freedom to use the full expressivity of RDFa. That means that not just the OMDoc-1.2-style *metadata* records but *any* OMDoc element may have RDFa attributes – including, for example, informal text structured using “rich text” markup. The second reason for fully integrating RDFa is compatibility to RDFa tools. When publishing the sources of OMDoc documents on the Web, linked data crawlers such as Sindice [[TDO07](#)] may find them. While they would not be able to make any sense of OMDoc’s own XML vocabulary – e.g. understanding that, by our OMDoc→RDF translation, a *proof* element represents an instance of the *oo:Proof* class –, they would at least be able to understand the annotations made in RDFa, and thus enable users to search for, e.g., resources having the *dc:creator* MICHAEL KOHLHASE.

Table 5.1: Elements of the recommended RDFa syntax for OMDoc metadata

Element	Attributes	Children
<i>meta</i>	<i>@property</i> , <i>@content</i> , <i>@datatype</i>	literal text or XML (optional)
<i>link</i>	<i>@rel</i> , <i>@rev</i> , <i>@resource</i>	( <i>resource meta link</i> ) <sup>*</sup>
<i>resource</i>	<i>@about</i> , <i>@typeof</i>	( <i>meta link</i> ) <sup>*</sup>

We have integrated all attributes of RDFa listed in [table 2.2](#), with the exception that the optional *@src* and *@href* attributes, which originate from RDFa's XHTML heritage, are only allowed on those elements of OMDoc's RT module that have also supported them in OMDoc 1.2. We define the OMDoc ontology both as default vocabulary and bind it to the default prefix, but we leave the introduction of new terms specifically for usage with RDFa to future work.

A full integration of RDFa into OMDoc has two consequences, which the following subsections discuss: In documents with few, relatively simple metadata, the unrestricted RDFa syntax may give authors too much flexibility and too little orientation in where to put the metadata. For such cases, we recommend a metadata syntax that resembles the one of OMDoc 1.2. Secondly, and more importantly, RDFa annotations may interfere with OMDoc's native markup, as OMDoc itself is already a semantic markup language. [Section 5.2.3](#) discusses how that can be avoided.

### 5.2.2 Recommended RDFa Syntax for Metadata Records

For a syntax that gives access to arbitrary metadata vocabularies but otherwise corresponds to the OMDoc 1.2 metadata syntax as closely as possible, we introduce the elements *meta* and *link*<sup>2</sup> as children of any *metadata* record.<sup>3</sup> Their syntax and semantics is roughly inspired by the namesake elements that can occur in the *head* of an XHTML document: *meta* is a literal-valued metadata field, whereas *link* points to another resource by referring to its URI. Blank nodes can be created using the *resource* element. The elements are shown in [table 5.1](#); examples for their usage are given in the following sections.

### 5.2.3 Semantic Interaction of RDFa and OMDoc Markup

RDFa has originally been designed for usage with XHTML and mostly been integrated into presentation markup languages so far (except for a proposed integration into DocBook and DITA reviewed in [section 5.3](#)). In presentation markup, annotations usually describe entities different from those represented by the XML elements that carry the annotations. An RDFa annotation in XHTML, unless at the top level of a document, would rather describe the *topic* discussed in a paragraph than the paragraph itself. Even if the paragraph has a fragment ID and thus a URI, this URI largely serves presentational purposes, such as enabling navigation from a table of contents to the paragraph, but less so as a means of identifying the paragraph as a semantic

<sup>2</sup> Actually, the *link* element has existed before, as a part of OMDoc's RT module [[Koho6b](#), section 14.6]. However, its usage in the RT module does not conflict with its usage as a *metadata* element.

<sup>3</sup> Note that RDFa processors do not need the *metadata* container element, as it does not carry any RDFa attributes. It is merely a means of structuring the OMDoc syntax. Therefore, we have now made it optional.

entity. If the paragraph itself is annotated, e.g. with information about its author, these annotations have a subject URI different from the URI of the *topic* discussed in the paragraph. — In contrast, annotations in OMDoc have always been conceived as annotations for the entities represented precisely by the respective XML elements, such as theories or statements. Therefore, we have to make ensure that the subject URI of annotations to an element of an OMDoc document is identical to the URI of that element.

The following amendment to the requirements for extracting RDF from OMDoc stated in [section 3.7](#) caters for that and make sure that an implementation of the OMDoc→RDF translation processes RDFa annotations in OMDoc in compliance with the RDFa specification:

**Identifiers:** The identifier – URI, blank node ID, or anonymous blank node – of a structural entity **MUST** be determined according to the RDFa processing rules for identifying a *new subject* [[ABM+11](#), section 7.5]. Wherever an RDF triple is generated according to the RDFa processing rules and its *new subject differs* from the OMDoc fragment or MMT URI of the current OMDoc element or the nearest of its ancestors that can have a fragment ID or MMT name (i.e. the URI that would be used for translating OMDoc markup to RDF), the extractor **SHOULD** issue a warning.

**Structures:** All RDFa annotations **MUST** be translated to RDF by applying the RDFa processing rules [[ABM+11](#), section 7.5].<sup>4</sup>

In the following, I will discuss the consequences of these requirements for authors.

Listing 5.2: Problematic usage of RDFa in OMDoc

```
<omdoc>
  <metadata>
    <meta property="dct:title">Famous Proofs</meta>
  </metadata>
  <proof xml:id="fermat-proof" for="#fermats-last-theorem">
    <metadata>
      <meta property="dct:title">Proof of Fermat's Last Theorem</meta>
      ...
    </metadata>
  </proof>
  ...
</omdoc>
```

As an example for the identifier generation requirement, consider [listing 5.2](#). When processing the innermost *meta* element, there is no attribute such as *@about*, which would establish a *new subject* according to processing rules 5 and 6 [[ABM+11](#), section 7.5]. Therefore, the value of the *parent object*, which is, at that point, the base URI *U* of the document, is used for the *new subject*, yielding a triple `<U> dct:title "Proof of Fermat's Last Theorem"`, i.e. giving the document a second title besides “Famous Proofs”. More seriously, the subject of the triple generated from the *proof* element is now

<sup>4</sup>This requirement entails the first sentence of the “identifiers” requirements. That sentence is therefore merely informative and emphasizes what the developer of an OMDoc +RDFa→RDF translation needs to pay particular attention to, in contrast to developing a translation from semantic markup without RDFa to RDF.

also determined according to the RDFa processing rules, resulting in `<U> rdf:type oo:Proof`. But from processing the *omdoc* root element, we already have `<U> rdf:type sdo:Document`. However, the OMDoc ontology, which reuses the SALT document ontology (cf. [section 3.3.1](#)), declares *oo:Proof* disjoint with *sdo:Document*. Thus, the document either makes a locally running reasoner fail, or, even worse, pollutes the Web of Data with inconsistent RDF triples.<sup>5</sup>

Thus, the OMDoc markup needs to be rewritten to `<proof xml:id="fermat-proof" about="#fermat-proof" for="...">`. Note, however, that by the changed rules for processing OMDoc+RDFa, the fragment ID is no longer required for determining an identifier of the RDF resource represented by the *proof* element. Thus, when authoring OMDoc with the primary purpose of obtaining RDF, one can omit the fragment ID in favor of an *@about* attribute. That said, our markup can be shortened to the more intuitive form `<proof about="#fermat-proof">`.

In contrast to RDFa, OMDoc 1.2 assumed that metadata contained in an XML element *E* always referred to the concept denoted by *E*, e.g., that the *dc:title* in [listing 5.1](#) is the title of the proof with the URI `#fermat-proof`. In the face of this fact, it might seem tempting to specify the following OMDoc-specific parsing rule: For elements that are allowed to have a metadata record according to the OMDoc 1.2 specification and which already have a fragment ID or MMT name, the *new subject* of the metadata annotations is implicitly set to the OMDoc URI of the respective element.<sup>6</sup> One could even specify that, if an element that can have a *metadata* record does not have a fragment ID or MMT name, a blank node should be generated as a *new subject*. However, as long as the RDFa specification does not provide a way for host languages to redefine the RDFa parsing rules – in a machine-comprehensible way! –, RDFa-aware software will not be able to handle OMDoc-specific exceptions. Therefore, we do not introduce any such custom rules for parsing RDFa in OMDoc.

### 5.2.4 Rewriting OMDoc 1.2 Metadata in RDFa

Due to the inherent flexibility of RDFa, any metadata vocabulary can now be used in OMDoc. However, the vocabularies discussed in [section 3.4.4.2](#) are particularly recommended due to their good coverage of the metadata that have already been supported by OMDoc 1.2. [Listing 5.3](#) shows the proof of Fermat’s last theorem once more, now redone using RDFa metadata, and using DCMI Terms for the revision history. Comparing this to [listing 5.1](#), particularly note the following features:

- We are able to link to resources, such as FOAF profiles (cf. [section 3.5.1](#)), that describe people (creators, contributors, etc.) in further detail.
- More than one predicate can be given per subject and object. This makes it convenient to say that a person is both an editor and a publisher of a document.<sup>7</sup>
- The complete revision history can be embedded into the document.
- Versions (or persons, or licenses) can also be described (as blank nodes) if they are only known in this document, i.e. are not globally identifiable by a URI.

<sup>5</sup>See the homepage of the “Pedantic Web” initiative [[HC09](#)] on the importance of consistency on the Web of Data.

<sup>6</sup>The RDFa-related microdata syntax of HTML 5 (cf. [section 2.3.3.4](#)) defines an empty attribute *@itemscope*, which conveniently creates an unnamed resource that can have annotations, like a blank node without any specific ID in RDF.

<sup>7</sup>*marcel:EDT* is only a subproperty of *dc:contributor* but not of *dc:publisher*.



Listing 5.3: Proof of Fermat's last theorem, with OMDoc's new RDFa metadata

```

<proof xml:id="fermat-proof" about="#fermat-proof" for="#fermats-last-theorem"
  prefix="dct:    http://purl.org/dc/terms/
    marcrel: http://www.loc.gov/loc.terms/relators/
    xsd:    http://www.w3.org/2001/XMLSchema#
    xhv:    http://www.w3.org/1999/xhtml/vocab#
    cc:    http://creativecommons.org/ns#">
  <metadata>
    <meta property="dct:title">Proof of Fermat's Last Theorem</meta>
    <link rel="dct:creator" resource="http://dbpedia.org/resource/Pierre_de_Fermat"/>
    <link rel="marcrel:AUT" resource="http://math.princeton.edu/~awiles/foaf.rdf#me"/>
    <link rel="marcrel:EDT dct:publisher" resource="http://kwarc.info/kohlhase/#me"/>
    <link rel="dct:hasVersion"><!-- Anonymous resource (bnode). We could also point -->
      <resource about="_:first">    <!-- to a repository URL of the previous version -->
        <link rel="dct:creator"
          resource="http://dbpedia.org/resource/Pierre_de_Fermat"/>
        <meta property="dct:created" datatype="xsd:date">1637-06-13T00:00:00</meta>
      </resource>
      <resource about="_:correct">
        <link rel="dct:replaces" resource="_:initial"/>
        <link rel="dct:creator"
          resource="http://math.princeton.edu/~awiles/foaf.rdf#me"/>
        <meta property="dct:modified" datatype="xsd:date">1995-05-01T00:00:00</meta>
      </resource>
      <resource about="_:digitalized">
        <link rel="dct:requires dct:source" resource="_:correct"/>
        <link rel="dct:creator" resource="http://kwarc.info/kohlhase/#me"/>
        <meta property="dct:issued" datatype="xsd:date">2006-08-28T00:00:00</meta>
      </resource>
    </link>
    <link rel="xhv:license"><!-- actually recommended: directly using
      the pre-defined license http://creativecommons.org/licenses/by/3.0/de/,
      which is the same as what we are constructing here -->
      <meta property="cc:jurisdiction" content="de"/>
      <link rel="cc:permits">
        <resource about="cc:Reproduction"/>
        <resource about="cc:Distribution"/>
        <resource about="cc:DerivativeWorks"/>
      </link>
      <link rel="cc:requires">
        <resource about="cc:Notice"/>
        <resource about="cc:Attribution"/>
      </link>
    </link>
  </metadata>
  <!-- The actual body of the proof -->
</proof>

```

- The DCMI Terms vocabulary allows for modeling the history of revisions more faithfully than the idiosyncratic Dublin Core extensions of OMDoc 1.2:
  - Successive revisions can be modeled as a linked list via *dct:replaces*, in addition to referring to them by *dct:hasVersion*. We did not model MICHAEL KOHLHASE’s digitalization of ANDREW WILES’s proof as such a replacement, but as a resource that is based on WILES’s proof via the *dct:requires* and *dct:source* properties.
  - We can distinguish a resource from its revisions, which has been done here, but DCMI Terms does not enforce that.
  - Specific subproperties of *dct:date* roughly allow for capturing the OMDoc 1.2 actions; more such subproperties could easily be added.
  - One shortcoming is that dates have to be made explicit to automated parsers by declaring a datatype for them. In contrast to the OMDoc XML schema, DCMI Terms does not enforce dates to be given in the ISO 8601 encoding of the *xsd:date* [BMo4b] but only recommends such a syntax. An OMDoc-specific extension of DCMI Terms could, however, enforce it.
- The license of this proof is a ready-to-use Creative Commons license that can simply be referenced using the *xhv:license* property<sup>8</sup> and its URI. Alternatively, we can construct it in place from elementary permissions and restrictions.

### 5.3 Related Work

BOB DUCHARME has demonstrated an integration of RDFa into the XML schemata of DocBook and DITA [DuCo9]. This approach, however, only considers the RDF semantics of the RDFa annotations, whereas the semantics of DocBook’s or DITA’s native markup is not taken into account. We have analyzed the semantic interaction, and potential interferences, of RDFa with OMDoc’s native markup. This is a novel aspect of the integration of RDFa into semantic markup languages.

### 5.4 Conclusion

By extending the OMDoc language to allow for embedding RDFa, we have given the metadata of OMDoc an RDF semantics, thus completing the work towards a uniform representation of all structural dimensions of mathematical knowledge started in chapter 3. Section 6.5.2 explains how that can be exploited for information retrieval. We have also made them syntactically compatible with semantic web standards. Existing RDFa tools (cf. [Rdfa]) can now be used with OMDoc, as far as they are not limited to XHTML+RDFa.

The integration of RDFa helps to extend the expressivity of OMDoc beyond logical/functional, presentational, rhetorical, and document structures. Given appropriate RDF vocabularies, it is

---

<sup>8</sup>This property from the XHTML vocabulary supersedes the former *cc:license* property [AAL+o8]. By the implementation of the ccREL ontology, this property is also a subproperty of *dc:license*, which in turn is a subproperty of *dc:rights* (cf. section 2.1.7.3).

now possible to maintain arbitrarily complex metadata and information about the application environment in OMDoc documents. It is also possible to extend the expressivity of OMDoc in those dimensions that it primarily intends to support – no longer necessarily by extending the XML schema, but simply by creating a new RDF vocabulary, for example an extension of the OMDoc ontology, and using it in RDFa annotations. The common web practice of “paving the cowpaths” can now be applied to the evolution of OMDoc. A new feature can first be tested inexpensively by creating basic RDF vocabulary for it and using that vocabulary in OMDoc documents via RDFa annotations. Those annotations that establish themselves in practice can then be promoted to native OMDoc features by creating XML markup for them.

Apropos native OMDoc markup: In very few cases, RDFa annotations are shorter than their OMDoc 1.2 counterparts; this is the case where RDFa enables a smarter choice of vocabulary, for example with the MARC relators and the predefined Creative Commons licenses. In most cases of simple key/value metadata fields (of type string), the RDFa annotations are about as long as the old OMDoc 1.2 markup. However, information that has to be represented as complex data structures in RDF, such as custom licenses or revision histories, RDFa causes a massive space blow-up, as can be seen from comparing [listing 5.3](#) to [listing 5.1](#). An author is forced to introduce a number of RDF resources, such as one per revision, and, depending on what he wants to express, does not necessarily benefit from the new possibility to attach additional information to these resources. In such situations, the authoring effort should be lowered – for example by a shorthand input syntax. Firstly, we will continue to support the “pragmatic” OMDoc 1.2 metadata syntax, whose strict counterpart will be RDFa, the pragmatic→strict translation being defined along the mapping of the OMDoc 1.2 syntax to RDF given in [section 3.4.4](#). This specification is conceptually easy for all cases of metadata except revision histories, where the right ontology yet has to be found, and it can be implemented, e.g., in XSLT in a straightforward way. As a compromise between the pragmatic and the strict syntax, we will also consider specifying a default RDFa *profile* for the OMDoc host language, which is allowed to define “*default terms, default prefix mappings, and a default vocabulary mapping*” [ABM+10] (see [section 2.3.3.4](#) for the terminology). For metadata beyond the OMDoc 1.2 vocabulary, we will also consider generic ways of facilitating input; [section 6.2.6](#) discusses possible approaches. Also note that the increased expressivity of RDFa, compared to OMDoc’s native XML syntax, makes it much easier to produce markup that is, even if syntactically valid w.r.t. an XML schema, semantically inconsistent. That makes semantic validation a crucial issue, which can partly be addressed by pragmatic syntax but also requires RDF-based approaches. [Section 6.3.3.1](#) discusses the possibilities.

Finally, the new possibilities to use unlimited metadata vocabularies in OMDoc documents and to refine their semantics in OMDoc will enable us to study and formalize the interactions between different dimensions of metadata in more detail.

## Acknowledgments

The initial version of the integration of RDFa into OMDoc has been developed in collaboration with MICHAEL KOHLHASE [[LK09](#)].



## **Part III**

# **Services and their Integration**



---

Effective collaborative management of mathematical knowledge requires a number of services to play together. [Chapter 6](#) introduces concrete mathematical collaboration scenarios that will be studied throughout this part. It analyzes the workflows that users have to accomplish in these settings, breaking them down into individual tasks. For many of these tasks in managing, understanding, and applying mathematical knowledge, primitive services and system components that support them already exist; further ones I had to develop. Thus, the chapter reviews the state of the art, but also presents new contributions, preparing the reader for the following chapters.

Chapters [7](#) and [8](#) introduce methods and techniques that enable designers of collaboration environments to integrate heterogeneous primitive services. That affects both the user-centered frontend – in mathematics most often a document-oriented interface – and the database backend of a system. [Chapter 7](#) introduces an approach to enable on-demand adaptation and information lookup in published documents by hooking interactive services into the fine-grained annotations of documents published from a semantic source. Different services often use different native knowledge representation formats – for technical but also conceptual reasons, which poses a challenge for integrating them. [Chapter 8](#) explains how transparent translations between different representations make knowledge accessible to heterogeneous services – to each of them in the granularity and language that it can handle best.

Finally, [chapter 9](#) presents SWiM, an integrated collaboration environment that combines the production of knowledge – creation, formalization, organization – with its consumption. This prototype serves the purpose of exploring the feasibility of integrating heterogeneous services in order to effectively support collaborative workflows. [Chapter 10](#) evaluates the usability of SWiM and of the services that it integrates in three of the collaboration scenarios presented initially. That permits conclusions on the usability of environments that integrate heterogeneous services in general and leads to recommendations on how to design them.





# Primitive Services for Managing Mathematical Knowledge

1. *Small is beautiful.*
2. *Make each program do one thing well.*
3. *Build a prototype as soon as possible.*
4. *Choose portability over efficiency.*
5. *Store data in flat text files.*
6. *Use software leverage to your advantage.*
7. *Use shell scripts to increase leverage and portability.*
8. *Avoid captive user interfaces.*
9. *Make every program a filter.*

—MIKE GANCARZ: The UNIX Philosophy [[Gan94](#)]

This chapter approaches collaborative management of mathematical knowledge from the primitives. [Section 6.1](#) introduces a number of realistic scenarios in which we have studied collaboration, and breaks the workflows that users have to accomplish in these settings down to atomic tasks. This whole chapter deals with the primitive services that support these tasks, grouped into the fields of editing, validation, publishing, information retrieval, and discussing about problems.

Services and system components for managing mathematical knowledge represented in the formats reviewed in [chapter 2](#) have existed for a long time, but not for all of the collaboration tasks we want to study. Therefore, this chapter combines reviews of state-of-the-art MKM services, presentations of our improvements to existing services towards a higher interoperability in a semantic web setting, presentations of entirely new services that I/we first had to develop in order to cover the scenarios of interest, and reviews of technology that is state of the art but has not been applied to MKM so far, with recommendations on how to apply it.

## 6.1 Tasks, Scenarios, and Required Primitive Services

The goal of this thesis (cf. [section 1.2](#)) is to provide a system architecture that supports human users in

1. creating new mathematical knowledge,
2. formalizing and organizing existing knowledge, and
3. expanding the “mental infrastructure” that the target audience of a collection of mathematical knowledge requires for a) understanding, b) reusing, and c) applying it.

Section 6.1.1 concretizes what these points mean in mathematical practice, section 6.1.2 describes realistic scenarios in which we have studied how to accomplish these tasks, and section 6.1.3 summarizes the primitive services that are required for that and that this chapter discusses in detail.

### 6.1.1 Tasks in Managing, Understanding, and Applying Mathematical Knowledge

Creating new knowledge means turning one’s mind state into a rough conceptualization, which is then refined during elaboration. New knowledge rarely comes out of nothing but is rather acquired from existing sources; often, it emerges in the last step of GEORGE PÓLYA’s problem-solving procedure mentioned in section 1.1, where the mathematician who has just solved one particular problem is asked to generalize the solution in order to make it applicable in related cases: “Try to make [the details of the solution] as simple as you can”, “make [more extensive parts of the solution] shorter”, “fit it into your formerly acquired knowledge as naturally as possible” (which means, in the setting of a collaborative knowledge base: integrate it with knowledge that already exists there), “scrutinize the method that led you to the solution, try to see its point, and try to make use of it for other problems” [Pól73], “scrutinize the result and try to make use of it for other problems”. Refining a sketch includes using a more rigorous style, as well as introducing suitable notation. Doing so not only helps oneself to understand the problem – and is therefore part of PÓLYA’s first step –, but it also helps others to verify the solution (cf. section 1.1 and [Heioo; DMLP79]).

Formalizing and organizing existing knowledge cannot be strictly separated from its creation. The original write-up of an idea already involves initial formalization and organization. Formalizing and organizing, particularly in cases where other collaborators enter the scene, also comprises uncovering errors, such as refuting an alleged proof and thereby stipulating improvement or recreation of that proof or even the conjecture according to IMRE LAKATOS’s method (cf. section 1.1 and [Lak76]).

Rigorous argumentation and suitable notation contribute to making mathematical knowledge comprehensible. Here, “suitable” particularly refers to the intended audience. That entails not only an adaptive presentation, e.g. using context-specific notations or a degree of formality adequate to the user’s information needs, but also a context-specific selection of knowledge items presented to the user, as pointed out in section 2.1.5. Reusability of knowledge is facilitated by the above-mentioned integration with existing knowledge, which makes potentially reusable knowledge items easier to be retrieved not only by their own content and their own structural properties, but also by their relations to other knowledge. Reusability is also facilitated by a comprehensible presentation, which enables the potential (re)user to judge whether a particular knowledge item is suitable for reuse in the given situation. More technically, reusability is facilitated if knowledge is exposed in standard representation formats, possibly multiple ones to choose from. Applicability

has similar prerequisites as reusability. Additionally, it is facilitated by integrating the mathematical knowledge with a model of the application environment, as pointed out in the works cited in [section 2.1.7.7](#).

### 6.1.2 Concrete Workflows and Usage Scenarios

This section introduces a number of concrete collaborative MKM scenarios that we have studied. For each case, I describe how tasks are performed at the moment, in the absence of specialized tools, and point out the problems resulting from that. Throughout this chapter, which introduces primitive services that address individual steps of such workflows, the scenarios serve as running examples. By integrating multiple primitive services, employing the techniques introduced in [chapters 7 and 8](#), systems will be able to address the workflows in their full complexity. A key design goal for any such system is to address the problems pointed out below. This holds in particular for the SWiM semantic wiki, an integrated collaboration environment presented in [chapter 9](#). In this environment, improved support for the first three of the following workflows, all of which are related to collaboratively maintaining OpenMath CDs, has been realized in coherence. [Chapter 10](#) summarizes an evaluation of whether that solves the problems mentioned.

#### 6.1.2.1 Quickly Fixing Minor Errors

In the course of formalizing and organizing existing knowledge, or making it more comprehensible, one eventually arrives at a stable state. In such a state, fundamental structural changes, which might require a LAKATOSian discussion, are no longer required<sup>1</sup>, but there may still be minor errors left. In collaborative authoring – here, concretely, in Wikipedia – minor errors have been characterized as follows: They have an obvious solution, “*the editor believes [the fix] requires no review*”, and they “*could never be the subject of a dispute*” [[Metc](#)]. Note that minor errors are not only presentational – in this category, Wikipedia names “*spelling*”, “*simple formatting*” and “*layout*” [[Metc](#)] –, but they can also affect the semantics – Wikipedia names “*obvious factual errors*”, “*adding and correcting [...] links*”, and “*removing vandalism*”. This thesis focuses on minor fixes that affect a single piece of content. This is based on the assumption that, in a web 2.0 environment, every reader is a potential collaborator and should be able to instantly fix an error once spotted. If the fix affects semantic markup that is not translated to presentation markup in a one-to-one way, it may be necessary to verify the result of republishing the knowledge item. [Section 6.1.2.2](#) covers such a workflow for the special case of fixing notation definitions.

Here is, for example, the concrete workflow of fixing a minor error spotted in the description of a symbol in the rendered presentation of an OpenMath CD:

1. Update the local working copy of the repository that contains the CDs.<sup>2</sup>
2. Open the CD file in question.
3. Navigate to the *Description* child of the symbol in question.

---

<sup>1</sup>... or the collaborators *think* that they are no longer required. With previously unrecognized errors uncovered, or other major reorganizations decided, the state of a knowledge item can jump back from “stable” to “unstable”.

<sup>2</sup>The official and contributed OpenMath CDs reside in a Subversion repository at <http://svn.openmath.org/OpenMath/>; exact URLs of subdirectories are given in [section 9.3.2.2](#).

4. Fix the error.
5. Commit the file to the repository.

Problems with these steps, which a system will have to address, are:

2. The location of a CD file may not be obvious from its rendered presentation, not to mention that a static XHTML+MathML presentation does not provide direct access to an editor.
3. In a text editor, the user would have to do a manual search. In an XML editor, depending on its configuration, there may be an easier way to access child elements.
4. Fixing the error is not a problem per se. Specialized editors for OpenMath CDs and OpenMath objects exist (cf. [section 6.2.1.4](#)).
5. The user can only point out what exactly he has changed (e.g. “fixed a typo in the description of the *sin* symbol”) by manually putting it into the log message. Due to the effort this involves, it is rarely done.<sup>3</sup>

I have not studied the scenario of creating new CDs – except for reviewing them, as discussed below in [section 6.1.2.3](#). A survey in the OpenMath community confirms that users more frequently edit existing CDs than creating new ones (cf. appendices [D.2.1.6](#) and [D.2.1.7](#)).

#### 6.1.2.2 Fixing and Verifying Notations

Notational errors form a special case of non-semantic minor errors. Like other minor errors, readers spot them in a published document, but they are not caused by the semantic markup of what the reader is looking at – unless a mathematical object uses a wrong symbol. Instead, they are caused by a notation definition for the respective symbol.<sup>4</sup>

As a concrete example, consider the workflow of fixing a notational error in an OpenMath setting:

1. Identify what symbol (in terms of *cdbase/cd#name*) is mis-rendered.
2. Find the file – notation dictionary or XSLT stylesheet – where the notation of that symbol<sup>5</sup> is defined. (If necessary, update the working copy of the repository.)
3. Try to fix the notation definition.

---

<sup>3</sup>Typical log messages of minor fixes made to CDs in the OpenMath Subversion repository do describe the kind of change that has been made, but they do not name the part of the CD, e.g. the symbol definition, that has been affected by the change. Other collaborators can only find that out from inspecting the diff.

<sup>4</sup>This thesis focuses on notational errors caused by the *rendering* part of a pattern-based notation definition, by the body of an XSLT notation definition, or, in case of a declarative notation definition, by any information except the symbol and the role. Notational errors caused by a wrong matching of notation definitions to symbols cannot generally be fixed that way; they would have to be fixed in those notation definitions whose prototype or symbol/role declaration matches the *content markup* of the (sub)formula in question.

<sup>5</sup>This workflow description assumes that each symbol has one notation definition. [Section 6.4.2.4](#) considers the case of multiple notation definitions.

4. Regenerate the document in which the rendered symbol was spotted originally. Ideally, regenerate the rendered presentations of *all* documents in the collection where the symbol occurs, so that others do not have to do it.
5. Open the regenerated document and check whether the symbol has now been rendered correctly. If not, repeat from [step 2](#).
6. Commit the notation dictionary or XSLT stylesheet file to the repository, giving a meaningful log message.

Problems with these steps, which a system will have to address, are:

1. In the worst case that the XHTML+MathML presentation is not annotated with parallel markup, it is not obvious what (semantic) symbol has been mis-rendered.<sup>6</sup>
2. Locating the source of the notation definition of that symbol is as much of a problem as pointed out above for minor fixes to the content of a CD.
3. Dedicated support for editing notation definitions is rare. So far, only one specialized editor for the ActiveMath pattern matching syntax (cf. [section 2.4.5.2](#) and [\[MLU+06\]](#)) is known; [section 6.2.7.4](#) briefly reviews it.
4. The process of republishing documents after a change to their sources has traditionally been controlled by makefiles. It is easy to implement the dependency of a published CD on its source file as a makefile rule, but implementing the dependency of a published CD on all notation dictionaries that contain notations for symbols used in that CD is much harder and therefore has not been done in practice. Hence, starting the publication process after a change to one notation definition practically republishes all CDs in the current repository directory.
6. For the log message, the same as what has been said above for minor fixes holds.

### 6.1.2.3 Peer Review and Preparing Major Revisions by Discussion

Unlike minor fixes, major revisions, particularly those that affect the semantics of a knowledge item and of dependent knowledge items, require discussion and mutual agreement. Similarly, new contributions to a knowledge base may be subject to a peer review before they are accepted. Communities have traditionally used mailing lists or issue trackers for such purposes.

This section discusses two concrete examples from OpenMath: peer reviewing a newly submitted CD, and discussing a problem with an existing CD. In practice, the peer review procedure required by [\[BCC+04, section 4.5\]](#) works as follows:

1. The author of the CD submits his initial version to the OpenMath issue tracker [\[Opec\]](#).

---

<sup>6</sup>The XSLT stylesheets that are employed for publishing at [openmath.org](#) do not output cross-linked parallel markup, which would directly link each presentation markup symbol to its semantic counterpart. Besides Presentation MathML, they do, however, output various semantic representations of an OpenMath object, as explained in [appendix C.1.2.1](#); therefore, an educated reader can identify the semantic symbol with some brain power.

2. The peer reviewers review it and point out remaining issues and suggest solutions.
3. The author addresses the reviewer's comments. Steps 1 to 3 iterate until there are no more objections.
4. The new CD is added to the repository.

Issues with existing CDs could in principle be resolved in the same issue tracker, but in practice they have been addressed differently. Suppose a person spots an issue, such as a symbol having a wrong *FMP*, but is unable to solve it himself.

1. He first makes the community aware of the issue, stating the problem as exactly as possible. In most cases, the [om@openmath.org](mailto:om@openmath.org) mailing list has been used for such purposes. An alternative installation of the Trac issue tracking system [Traa] has not been used consistently [Opea], as a survey in the OpenMath community confirmed (cf. [appendix D.2.1.8](#)).<sup>7</sup>
2. Other developers reply to this e-mail and propose solutions and discuss them.
3. The community agrees on one solution to be implemented in the repository.

The current state of reviewing CDs and discussing revisions exhibits the following problems:

- New CDs are usually only submitted for review as source files; reviewers who want to read them in their published appearance have to render them manually.
- A reader who spots an issue in the published XHTML+MathML version of an existing CD cannot immediately start a discussion about it, but has to compose an e-mail or go to the issue tracking site.
- When the author of a new e-mail or issue report wants to help others to inspect the problem, he has to manually paste a link to the repository URL of the CD in question into his post.
- On a mailing list, there is, in contrast to an issue tracking site, no possibility of retrieving a quick overview of discussions by their subject or state, such as: all discussions about a given CD, all discussions about notation definitions, all ongoing discussions about unsolved issues, all past discussions about issues that have been solved, etc.

#### 6.1.2.4 Serving Information Needs of Learners and Instructors

Students who want to understand mathematical lecture material may want to directly look up the meaning of a unknown symbols (e.g.  $\equiv$ ) in a mathematical object, or search for examples for a difficult concept (e.g. structural induction). Furthermore, they may want to use sample exercises covering the topics of last week's lecture to prepare for an upcoming exam.

---

<sup>7</sup>The "OpenMath 3" Trac has mainly been used for discussing issues related to the OpenMath specification, and additionally few issues with CDs. Most of these discussions had been seeded from e-mails, one per CD, in an attempt to shift the discussion into the Trac; they have also received a few new comments. Trac, as well as Jira, the system used for reviewing new CDs, would allow for integrating a Subversion repository, so that collaborators can link issue descriptions to repository resources, and commits to repository to the issues that they resolve (cf. [section 6.6.1](#) and [Trab]), but that has not been enabled.



Lecturers are rather concerned with reusing and applying knowledge in class than with understanding, but their information needs are similar. They may want to find a suitable topic for the next class, i.e. a topic that has minimal additional prerequisites beyond the material covered so far, or a set of exercises for the exam that covers the topics of the lecture well. They want to introduce new concepts via examples tailored to the audience: For example, a tree in a computer science lecture could be exemplified by the parse tree of a context-free language to students with a formal language background, or as a directory tree to students with an operating system background. There are also searches related to the organization of the lecture note collection, for example searches for didactic gaps, such as concepts without examples, or unjustified proof steps.

We have specifically studied these information needs on the collection of MICHAEL KOHLHASE's lecture notes introduced in [section 2.4.7.2](#), which comprises around 2000 OMDoc documents – obtained from an  $\text{\LaTeX}$  source – in a Subversion repository<sup>8</sup>. However, both groups of users may not want to restrict their searches to a single repository at their own university, but also find related material in other universities' online course notes, on mathematical web sites, or Wikipedia.

### 6.1.2.5 Managing a Project

Finding unresolved issues (cf. [section 6.1.2.3](#)) and gaps in a knowledge collection (cf. [section 6.1.2.4](#)) are aspects of project management. More specifically, we have studied the information needs of project managers in software engineering and knowledge formalization.

In the software engineering scenario mentioned in [section 2.1.7.7](#), we have encountered the following issues [[KKL10a](#)]:

**Software Process:** How much code has been implemented to satisfy a particular requirement from the contract? Has the formal code structure passed static analysis and verification? A project manager does not want to check that manually but needs high-level figures, such as a list of verified code modules.

**Certification:** What parts of the specification, e.g. requirements, have changed since the last certification? What other parts does that affect, and thus, what subset of the whole specification has to be re-certified?

**Human Capital:** Who is in charge of a requirements specification, a module of the mathematical model, an implementation or user's manual? How could an author be replaced if necessary, taking into account colleagues working on related topics?






















As a specific question occurring in a collaborative formalization effort – concretely: the Flyspeck project introduced in [section 1.1](#) –, we have identified a query for lemmas that are difficult to prove, in that many collaborators had already attempted them without success, or in that many collaborators has asked questions in the associated discussion forum [[LMRo8](#)].

Both scenarios involve knowledge in human-comprehensible as well as computerized representations, which are, in the current state, maintained independently from each other. In the software engineering scenario we have studied, the contracts, requirements specifications, and manuals had been written in  $\text{\LaTeX}$ , the formalization had been done in Isabelle, and the implementation in






---

<sup>8</sup>The repository is actually powered by the Subversion-compatible TNTBase database system now (cf. [section 6.5.2.1](#)).

Table 6.1: Matrix of primitive services vs. knowledge management tasks they help to accomplish<sup>a</sup>, and scenarios for which that has been investigated here<sup>b</sup>

▼ accomplishes ►	Creating	Formal./Organizing	Understanding	Reusing	Applying
Editing					
Validating					
Publishing					
Inform. Retrieval					
Arguing					

<sup>a</sup> The size of the circles represents the extent to which a service contributes to accomplishing a task.

<sup>b</sup> Scenarios:  quickly fixing minor errors (section 6.1.2.1),  fixing and verifying notations (section 6.1.2.2),  peer review and preparing major revisions by discussion (section 6.1.2.3),  serving information needs of learners and instructors (section 6.1.2.4),  managing a project (section 6.1.2.5)

C [KKL10a]. Similarly, Flyspeck comprises a book written in  $\text{\LaTeX}$  and formalizations done in Twelf and other languages.

### 6.1.3 Primitive Services and the Tasks they Accomplish

Neither the general tasks introduced in section 6.1.1 nor the specific workflows introduced in section 6.1.2 can be accomplished by a single service. Instead, they require the integration of several primitive services, as summarized in table 6.1. For example, organizing a collection of mathematical knowledge requires a combination of validation, information retrieval, and other primitive operations. Fixing the notation of a symbol requires a combination of editing and publishing services. Before we can practically integrate such primitive services in an environment that assists users in accomplishing complex, realistic tasks, we need to an overview of what services exist and what knowledge representations they operate on. This chapter reviews – or, where missing, introduces – services for the following primitive operations:

**Editing** services should support authors with knowledge acquisition and stepwise refinement – be it further formalization, or annotation with human-comprehensible explanations. The different structures of mathematical knowledge require different editing interfaces. Section 6.2 covers editors for logical and rhetorical structures of documents, for formulæ, for notation definitions, and for metadata separately, also considering different types of user interfaces: editing raw source code, optimized text input syntaxes, presentation-oriented visual interfaces, and input forms. In addition to reviewing the state of the art, I present the extension of an existing presentation markup editor into a versatile reusable editing component for all kinds of semantic representations of mathematical knowledge: logical and rhetorical structures, formulæ, notation definitions, and metadata.

**Validating** mathematical knowledge is not only of interest to a reader or (re)user who wants to make sure that, e.g., an assertion is true and therefore safe for teaching it to students or for

leveraging it in a software implementation. A certain degree of validation is also a prerequisite for automated services to operate on a machine-comprehensible representation of mathematical knowledge. Closely integrating such validation services into collaborative editing workflows supports authors in preventing mistakes and makes sure that the formalizations they produce are usable for other services and useful for other users. Conversely, an expressive representation of knowledge again enables a more in-depth validation of, e.g., whether it is being reused as intended. [Section 6.3](#) provides an overview of the consecutive stages of validating mathematical knowledge: First of all, the syntax has to be valid; then, several aspects of semantic validity can be checked – ranging from low-level checks for, e.g., the existence of link targets, to high-level checks for, e.g., the well-typedness of mathematical expressions.

**Publishing** targets humans as well as machines; [section 6.4](#) covers both. Publishing mathematical knowledge in a machine-comprehensible way enables reuse and application. Publishing for humans comprises rendering machine-comprehensible representations into human-comprehensible documents, and enabling navigation through them. Both are prerequisites for understanding knowledge, but navigating through rendered documents also helps users to uncover cases of misorganization and poor reusability. I first explain how to publish mathematical knowledge as linked data for machine usage. Publishing for humans, however, is not limited to printing on read-only paper either; therefore, I demonstrate a way of embedding linked data into published documents, where services can access them and assist users right in the setting in which they consume mathematical knowledge.

**Information Retrieval** comprises searching and querying and is particularly relevant in large collections that a user can no longer explore by interactive navigation. Users not only have to retrieve knowledge they want to apply, but also existing structures into which they want to integrate new knowledge. Retrieval is covered in [section 6.5](#), divided into formula search, querying structures above the object level, and reasoning with ontologies implemented as mathematical documents. Particular attention is devoted to a qualitative comparison of two alternative querying approaches, both of which can now be used on the same collection of mathematical knowledge, thanks to the contributions to mathematical knowledge representation made in chapters 3 and 5: querying XML vs. querying RDF.

**Arguing** about mathematical knowledge is covered in [section 6.6](#), with a particular focus on discussing – in the manner of LAKATOS – problems with the formalization, comprehensibility, reusability, or applicability of knowledge items and possible solutions of these problems. My approach is inspired by tools that support bug tracking in collaborative software development, powered by an argumentation ontology with mathematics-specific extensions (introduced in sections 2.1.8 and 3.6). I give general recommendations for enabling a focused discussion of problems and solutions using appropriate user interfaces, and how knowledge organization systems should assist users with solving problems based on the result of a discussion.

**Further Services for Managing and Applying Mathematical Knowledge** are not covered in detail in this thesis. The services mentioned so far comprise a *selection* of services relevant for

collaboration on mathematical knowledge. Beyond mere validation, change management has been investigated for mathematical and technical documents [Mül10b; AM10]. While [chapter 8](#) mentions versioned repositories for mathematical knowledge, archiving is not a focus of this thesis. It has, for example, been investigated in the T<sub>E</sub>XDocC project [T<sub>ex</sub>a]. Services that utilize mathematical knowledge for applications are only marginally touched in this thesis as well. These are, for example, services for mathematical education and for computation. Services for education have been developed for the ActiveMath [Act] and MathDox [Matb] e-learning systems. Services for symbolic computation have been integrated in the SCIENCE project [Sci], whereas services for numeric computation have been integrated in the MONET project [Mona].

## 6.2 Editing

In [chapters 3 to 5](#), OMDoc – with MathML, OpenMath, and RDFa – has been presented as a markup language for representing all aspects of mathematical knowledge. Throughout the history of complex representation languages, editing has always been supported by specialized user interfaces. After reviewing existing alternative approaches in [section 6.2.1](#), I present new ways of editing semantic representations of mathematical knowledge that we have developed. On the one hand, there is  $\text{\LaTeX}$  as a frontend input syntax that currently works one-way, an OMDoc $\rightarrow$  $\text{\LaTeX}$  translation being a feasible future extension ([section 6.2.2](#)). On the other hand, there are two-way visual interfaces: a document-oriented editor with annotation support for logical structures above the object level, as well as for rhetorical and document structures ([section 6.2.3](#)), a linear frontend input syntax combined with a visual interface for formulæ ([section 6.2.4](#)), a combination of the former two approaches for notation definitions ([section 6.2.5](#)), and forms for simple metadata ([section 6.2.6](#)).

### 6.2.1 State of the Art, by Type of Interface

This section provides an overview of the following state-of-the-art approaches to editing, with a focus on semantic markup and mathematical knowledge: (i) plugins for general-purpose text or XML editors that facilitate handling complex structures, (ii) a text-based frontend input syntax that is translated into the actual target language, (iii) visual interfaces following the WYSIWYG or WYSIWYM<sup>9</sup> paradigms, and (iv) forms. All of these fundamentally different approaches are extensible orthogonally by tool buttons, menus, or keyboard macros for inserting frequently-used content or performing other repetitive tasks; this is not covered here.

#### 6.2.1.1 Unrestricted Access to Markup with Support

Any general-purpose text or XML editor is capable of editing the semantic markup reviewed in [section 2.4](#). This is often facilitated by language-specific syntax highlighting, code indentation, section folding, and templates. This approach has been used most widely for mathematical markup languages so far, due to the availability of high-quality extensible text editors and the relative ease of implementing extensions. There is no conceptual difference between editing formulæ and other document structures.

---

<sup>9</sup>What you see is what you get/what you see is what you mean

For the Emacs editor, editing modes for OMDoc and  $\text{\LaTeX}$  have been developed. After an initial implementation from scratch (cf. [Jano6]), the OMDoc mode has been reimplemented as an extension of Emacs's `nxml` mode for editing XML [Pes07]. The  $\text{\LaTeX}$  mode [Ste] is an extension of the AUCTeX mode for  $\text{\LaTeX}$ . More recently,  $\text{\LaTeX}$ XIDE, an  $\text{\LaTeX}$  plugin for the Eclipse software development platform, has been developed [JK10].  $\text{\LaTeX}$ XIDE particularly benefits from Eclipse's support for handling collections of multiple files that belong to a project. jEditOQMath has been developed as a distribution of the jEdit editor, bundled with plugins for editing OMDoc for the ActiveMath e-learning system [Lib10]. jEditOQMath does not expose formulæ in their original markup but uses the QMath frontend input syntax.

### 6.2.1.2 Frontend Text Input Syntax

The complexity of semantic markup can often be reduced by a textual frontend input syntax that is easier to read and to write than the original representation. This is usually a one-dimensional syntax, as opposed to the two-dimensional layout of rendered mathematical objects; in the latter context, it is therefore also called *linear syntax*. Any pragmatic syntax that translates to a strict syntax, following the terminology of section 2.4.4.1, can also be considered a frontend input syntax.

There are languages that act exclusively as an *input* syntax: A preprocessor translates the frontend representation to the actual representation language, but the inverse translation has not been implemented. In integrated development environments, two-way translations are more common: On loading a document, it is translated to the frontend input syntax; on saving, it is translated back. For complex semantic markup languages, the challenge consists both in developing a frontend input syntax that gives access to all features of the original language, and in developing a lossless two-way translation – ideally one that respects any source code formatting, such as whitespace or comments, made in the text input. Once a frontend input syntax has been chosen, further support for editing it can be provided in any other way described here. The above-mentioned support for editing  $\text{\LaTeX}$  is, in fact, support for editing a frontend input syntax to OMDoc.

$\text{\LaTeX}$  is discussed separately in section 6.2.2, as it is the most widely used text input syntax for OMDoc, and as editing support for the OMDoc extensions introduced in chapter 5 have so far only been implemented in  $\text{\LaTeX}$ . Popcorn, a linear input syntax for OpenMath, and QMath, a linear input syntax for OpenMath objects and OMDoc documents, are discussed here. The N3 and Turtle serializations of RDF (cf. section 2.3.3.2) are further examples of a text input syntax but are not discussed in this section, as RDF has not been chosen as the primary representation for mathematical knowledge in section 2.5.

Popcorn [HR09b; HR09a] is a simple input syntax for OpenMath that focuses on CAS integration. Its supply of built-in infix and mixfix operators is not currently extensible, but there are some built-in notations for CAS-specific programming constructs from the *progl* CD. The Java library for SCSCP and OpenMath provides two-way translations between Popcorn, OpenMath 2 XML, and the efficient OpenMath 2 binary encoding [HR09b].

QMath [GP06a] has originally been created as an extensible linear input syntax for OpenMath objects. Special support for arbitrary mathematical symbols – besides the generic syntax that supports any OpenMath symbol – can be provided by so-called “contexts” consisting of declarative notation definitions (cf. section 2.4.5.3). Such contexts are bundled with QMath for the official OpenMath 2.0 CDs. In addition to QMath's own syntax, the QMath processor supports alternative

syntaxes resembling the syntaxes of various CAS and an experimental  $\text{\LaTeX}$ -like syntax. Above the object level, QMath has not yet been widely used. However, it ships with a context file for OMDoc that facilitates entering statement- and theory-level structures and OMDoc 1.2 style metadata, as long as they are not too deeply nested.<sup>10</sup> Similarly as for mathematical symbols, QMath syntax for any XML language, such as OpenMath CDs, can easily be defined. The QMath processor itself is a simple one-way preprocessor; however, the Sentido editor (cf. [section 6.2.4](#)) implements a translation from OpenMath to QMath, using the QMath contexts for emitting the desired QMath syntax for symbols. A translation of the statement, theory, and document-level syntax of OMDoc to QMath is not available but would be fairly straightforward to implement in XSLT.

### 6.2.1.3 Visual WYSIWYG/WYSIWYM Interfaces

A WYSIWYG editor shows a document as it will be rendered, or close to that. Semantic annotations are hidden in the editor's internal representation of the document. They are usually only partially exposed in the document-oriented user interface. Access to them is, if at all, usually given via dialog boxes. For example, an annotated text fragment could be highlighted with some color to indicate that an annotation exists. The annotation would be revealed on mouse-over, and it would be editable by selecting the highlighted text and opening a dialog box.

Word processors are a typical example of WYSIWYG editors. The probably most widely used widget for WYSIWYG HTML editing in web applications is TinyMCE [[Tin](#)], the default editor of many CMS and blogs, and for which a large number of plugins exists. For these reasons it is also the preferred base for several extensions using semantic web technologies, e.g. within the Semantic Reblog republishing tool [[WMo9](#)], or for the One Click Annotator of the loomp semantic CMS [[HLRO+10](#)]. It is also used by the semantic wikis IkeWiki (cf. [section 9.3.1](#)) and KiWi (cf. [section 9.6.4.4](#)); for the latter, it has also been extended by custom annotation plugins [[SSo9](#)]. WYMeditor is a similar but less widely used editor emphasizing the WYSIWYM paradigm, which means that it focuses less on an exact reproduction of the rendered appearance of the document, but rather on visualizing its structure and enabling structural editing, and on generating clean XHTML [[Hov+](#)]. The MathLang extension of the  $\text{\TeX}_{\text{MACS}}$ <sup>11</sup> scientific editor (cf. [section 2.4.6](#)) does a similar job.

WYSIWY[GM] editors have also been developed for formulæ. Due to the two-dimensional layout of formulæ, this is harder to realize than for text. There are numerous WYSIWYG editors for Presentation MathML [[Matd](#)], less for content markup. WYSIWYG editors for OpenMath objects – with a restricted set of supported CDs – have been developed by WIRIS [[Wir](#); [MEC+o6](#)] and in the course of the MathDox project [[Matb](#)]. The Connexions MathML editor supports the built-in symbol vocabulary of Content MathML 2, plus embedded Presentation MathML [[Cnxd](#)].

### 6.2.1.4 Forms for Metadata and OpenMath CDs

Forms are commonly used as an interface for editing uniformly structured data, such as relational database tables or key/value metadata records. Forms support efficient editing of such data

<sup>10</sup>Alternatively, one can use the QMath syntax for formulæ only and OMDoc/XML for the rest, or mix XML and QMath fragments.

<sup>11</sup>There, the editing paradigm is called “*What you see is what you want*” [[Texb](#)].



structures, as they allow the user to focus on the actual data, whereas the editor takes care of structuring them. A challenge in developing form interfaces is reconciling scalability w.r.t. changes or extensions to the underlying schema with ease of use.

Due to its fixed vocabulary, the OpenMath CD language has been targeted with several form-based editors. The OpenMath CD manager [Hea09] is a web application; the RIACA group and JÓNATHAN HERAS VICENTE have developed two form-based desktop editors independently from each other [Ria; HV]. The RIACA CD editor has not been maintained since 2006, whereas HERAS's editor is currently in use. OpenMath objects are treated as single text values in each of these editors, without dedicated editing support. The OpenMath CD manager is the only one of these editors that focuses on CD maintenance tasks; section 9.5.3 details that. The RIACA CD editor focuses on generating Java code for programs dealing with OpenMath objects from CDs. HERAS's editor is a general purpose one for CDs and type signatures.

OntoWiki is a form-based editor for RDF and ontologies with an extensible vocabulary. Some document-oriented editors provide forms as an alternative access to certain structured data embedded in the documents, for example – again with an extensible vocabulary – the KiWi semantic wiki (cf. section 9.6.4.4), and Semantic MediaWiki via the Semantic Forms extension [Kor+10].

## 6.2.2 $\S\TeX$ as a Frontend Input Syntax for OMDoc+OpenMath+RDFa

$\S\TeX$  as a  $\LaTeX$  frontend input syntax for OMDoc 1.2 has been introduced in section 2.4.7.2. Meanwhile,  $\S\TeX$  has partly caught up with OMDoc's extension by RDFa-compatible metadata, which I briefly review here. New to this section is my discussion of steps that still need to be accomplished for enabling a translation from OMDoc to  $\S\TeX$ , which would make  $\S\TeX$  applicable in a more versatile way.

### 6.2.2.1 Scalable Metadata and Vocabularies in $\S\TeX$

MICHAEL KOHLHASE has partly adopted OMDoc's new ability to define vocabularies for RDF metadata (cf. chapter 4) and the RDFa-extended metadata syntax presented in chapter 5 for  $\S\TeX$  [KKL10b]. In cases that  $\S\TeX$ 's *generic* RDFa support does not yet cover, or where a shorter input syntax is desired (cf. the discussion in section 5.4),  $\S\TeX$  advocates supplying *specific* vocabulary as “pragmatic” macros – implemented as  $\LaTeX$ XML bindings.<sup>12</sup>  $\S\TeX$  environments for certain complex metadata structures, such as requirements specifications, are currently in an experimental state [Koh10c]. Following the pattern introduced there, one could implement further  $\LaTeX$ XML bindings, which would enable expressing the revision history given in the OMDoc RDFa metadata example in listing 5.3 as follows<sup>13</sup>:

```
\begin{DCTversions}
  \DCTversion[id=initial,creator={Pierre de Fermat},created=1637-06-13]
  \DCTversion[id=correct,replaces=initial,creator=awiles,date=1995-05-01]
  \DCTversion[id=digitalized,source=correct,creator=kohlhase,issued=2006-08-28]
```

<sup>12</sup>For example, the  $\LaTeX$ XML binding for the *dc:title* metadata element of OMDoc 1.2 looks as follows [Koh10a]:  
`DefConstructor('DCMtitle{}', "<dc:title>#1</dc:title>");`

<sup>13</sup>We skip certain peculiarities, such as URI references to authors, for simplicity.



`\end{DCTversions}`

### 6.2.2.2 Translating OMDoc to $\S\TeX$

The setup in which  $\S\TeX$  is currently being used, most prominently the repository of MICHAEL KOHLHASE’s lecture notes (cf. sections 2.4.7.2 and 6.1.2.4), assumes that content is exclusively acquired or enhanced via editing  $\S\TeX$ , which is then translated to OMDoc for any application scenario other than publishing printable PDF (cf. figure 6.9 on page 221). It is assumed that (i) the generated OMDoc documents are not edited, and (ii) that no external content available in OMDoc or in a language that can be translated to OMDoc is imported into the repository and should as well be made available for editing.

In future, more general application scenarios, these assumptions can no longer be maintained. When there are OMDoc documents that do not originate from an  $\S\TeX$  source but authors still want to benefit from the tool support for  $\S\TeX$  (for OMDoc, fewer editing tools are currently available!), they need an OMDoc $\rightarrow\S\TeX$  translation. Such a translation does not currently exist; merely a translation of OMDoc 1.2 to presentational  $\LaTeX$  had once been implemented. The following challenges have to be taken into account for making the right design decisions when developing an OMDoc $\rightarrow\S\TeX$  translation:

**Macro expansion:** A fully bijective translation is not possible in principle, as  $\S\TeX$  authors can take advantage of  $\TeX$  macros, which the  $\S\TeX\rightarrow\text{OMDoc}$  translation expands. This is not a problem as long as one assumes that external OMDoc content is only translated to  $\S\TeX$  once, on import into the knowledge collection, and then continued to be edited in  $\S\TeX$  (and possibly enriched with macros).

**Syntactical differences:**  $\S\TeX$  and OMDoc currently have different syntaxes for declaring symbols and their notations. OMDoc treats both separately, whereas  $\S\TeX$  has a combined syntax for declaring a symbol (without a type, though) and its notation, which resembles the  $\LaTeX$  `\newcommand` macro.

**Different treatment of  $n$ -arity:** More seriously,  $\S\TeX$  represents the argument list of an  $n$ -ary operator as a single  $\TeX$  argument that is a comma-separated list, in order to circumvent  $\LaTeX$ ’s arbitrary restriction that a macro can only have up to nine arguments. Operators with a fixed arity of  $n < 10$  are modeled as commands with  $n$  arguments – like `\frac{num}{den}` in  $\LaTeX$  –, but truly  $n$ -ary operators are modeled as unary  $\LaTeX$  commands, where the actual arguments are comma-separated, e.g. `\nunion{A,B,C}` for the  $n$ -ary set union  $A \cup B \cup C$ . In OpenMath, either expression would be represented as an OMA element with  $n$  children. Type signatures for symbols in OMDoc content to be imported would ease the problem but cannot generally be assumed to be available. A one-time  $\S\TeX\rightarrow\text{OMDoc}$  import, as mentioned above w.r.t. macro expansion, could thus simply treat all imported symbols as  $n$ -ary, leaving the task of changing the declarations of fixed-arity or flexary infix symbols<sup>14</sup>

<sup>14</sup>As an example for the common  $\S\TeX$  practice of modeling these, consider the function declaration symbol  $f: S_1 \times \dots \times S_n \rightarrow R$ , which could be written as `\function{f}{S_1, \dots, S_n}{R}` in  $\S\TeX$ . The first and last arguments are fixed, but there is a flexible number of arguments in between. The more semantic alternative

into a more structured syntax to the  $\text{\LaTeX}$  author for manual adaptation, a task similar to introducing macros. In a round-trip editing scenario with a permanent two-way translation,  $n$ -ary symbols actually constitute less of a challenge, as the  $\text{\LaTeX} \rightarrow \text{OMDoc}$  translation puts fine-grained back-references to  $\text{\LaTeX}$  source locations into its OMDoc output, which allow for keeping track of arguments. That given, even flexarity is not harder to handle than  $n$ -arity.

### 6.2.3 Visually Annotating Logical/Functional, Rhetorical, Document, and Presentational Structures in Documents

This section describes a general way of adapting a WYSIWYG HTML editor for semantic markup by leveraging HTML as a means of exposing the structures of semantic markup to the author. [Algorithm 6.1](#) defines a translation  $t: \text{XML} \rightarrow \text{HTML}$  that, by default, makes each XML element of the semantic markup accessible in the editor as an HTML table of a special CSS *@class*. The head row of one such special HTML table consists of three cells: (i) the name of the XML element, e.g. *definition* for an OMDoc definition, (ii) an *@xml:id* (possibly empty), and (iii) a list of additional attributes as “key=value” lines, e.g. “for=symbolname” (possibly empty). The table body consists of one cell that, recursively, contains the analogous representation of the content of the XML element. The environment that integrates such an editor has to apply the translation  $t$  to any semantic markup on loading a document into the editor, and the inverse  $t^{-1}$  on saving.

---

**Algorithm 6.1** Translating XML to editable HTML tables:  $t(x)$

---

**Require:**  $x$  is an XML node *// ... of a well-formed XML document, of course*

**Ensure:**  $h$  is a well-formed HTML table element (first case), or a string (other cases)

**if**  $x$  is an element **then**

$h \leftarrow \langle \text{table class} = \text{"semantic"} \rangle$  *// compliant thead/tbody markup ignored for brevity, ...*

$\langle \text{tr} \rangle \langle \text{th} \rangle \text{name}(x) \langle \text{th} \rangle$  *// ... XML namespaces as well*

$\langle \text{th} \rangle \pi_N(\$x/\text{@xml:id}) \langle \text{th} \rangle$  *//  $\pi_B = \text{XPath evaluation function returning an XML node or nothing}$*

$\langle \text{th} \rangle$  for each attribute  $a$  except *@xml:id*, call  $t(a)$ ,

concatenate the result strings (separated by  $\leftrightarrow$ )  $\langle \text{th} \rangle \langle \text{tr} \rangle$

$\langle \text{tr} \rangle \langle \text{td} \rangle$

for each text or element child node  $c$ , call  $t(c)$ , concatenate the result strings/HTML trees

$\langle \text{td} \rangle \langle \text{tr} \rangle \langle \text{table} \rangle$

**else if**  $x$  is an attribute **then**

$h \leftarrow \text{name}(x) + \text{"="} + \text{value}(x)$

**else if**  $x$  is a text node **then**

$h \leftarrow x$

**end if**

**return**  $h$

---



---

$\backslash \text{function}\{f\}\{\text{\crossproduct}\{S_1, \dots, S_n\}\}\{R\}$  takes an author more time to write and would not fundamentally change the above-mentioned translation problem either, as the cross product is still  $n$ -ary.

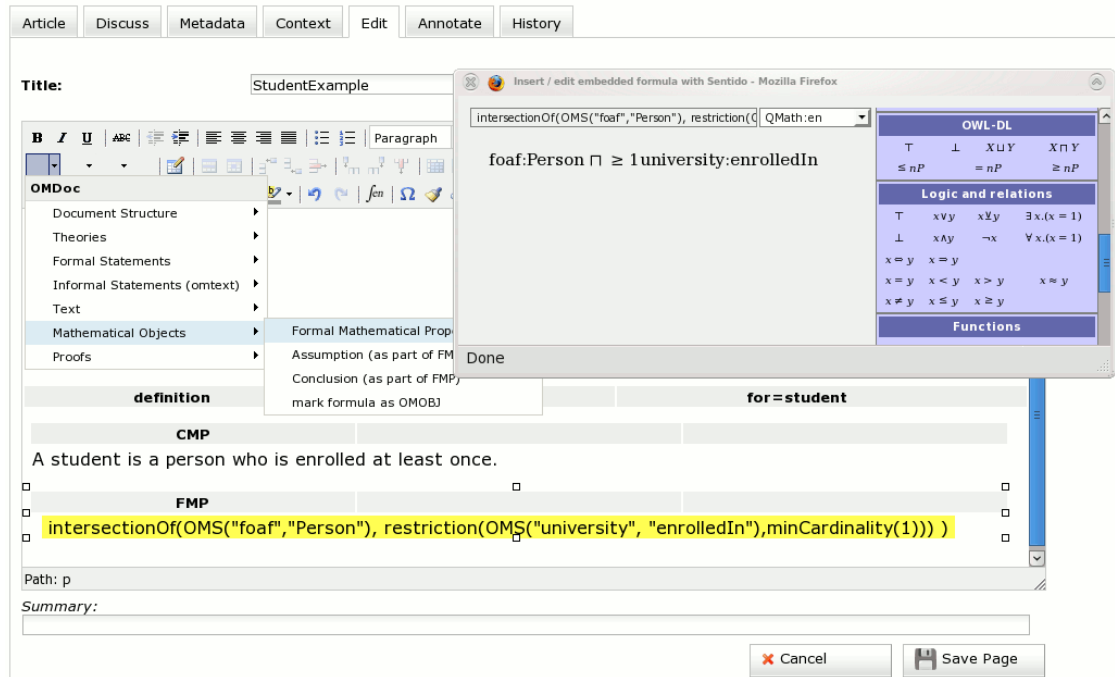


Figure 6.1: Editing an axiom of an ontology in an adapted WYSIWYG HTML editor with an integrated formula editor (TinyMCE+Sentido)

Depending on the use case, an application might additionally take advantage of the presentation-oriented WYSIWYG editing facilities, i.e. allow users to mix semantic markup tables with purely presentational HTML markup and extend  $t^{-1}$  to preserve the latter as a special kind of annotation – which might be reused for publishing the semantic markup.

While any XML markup – even mathematical formulæ – can be made editable via such HTML tables, this is not user-friendly for deeply nested structures, as the usability evaluation of the editor in the context of the SWiM semantic wiki confirmed (cf. [section 10.5.4.5](#)). Besides giving formulæ and metadata a special treatment (cf. [sections 6.2.4](#) and [6.2.6](#)), the HTML tables can also be applied more flexibly. I have realized a proof of concept for pattern-based notation definitions (cf. [section 2.4.5.2](#)) by inserting an additional case before “ $x$  is an element” into [algorithm 6.1](#), which handles *notation* elements in a special way. These notation definitions map a content markup pattern (“prototype”) to a presentation markup fragment (“rendering”), which the new case reflects by returning a side-by-side arrangement

notation	
prototype	rendering

instead of

notation
prototype
rendering

.

I have implemented such a translation for the integration of the TinyMCE editor ([section 6.2.1.3](#)) into the SWiM semantic wiki (cf. [chapter 9](#)); [appendix C.1.1.1](#) describes the technical details. [Figure 6.1](#) shows the editor in action, with additional user interface elements that allow for inserting elements of the supported semantic markup languages (shown here), as well as deleting the closest semantic annotation around the selection.

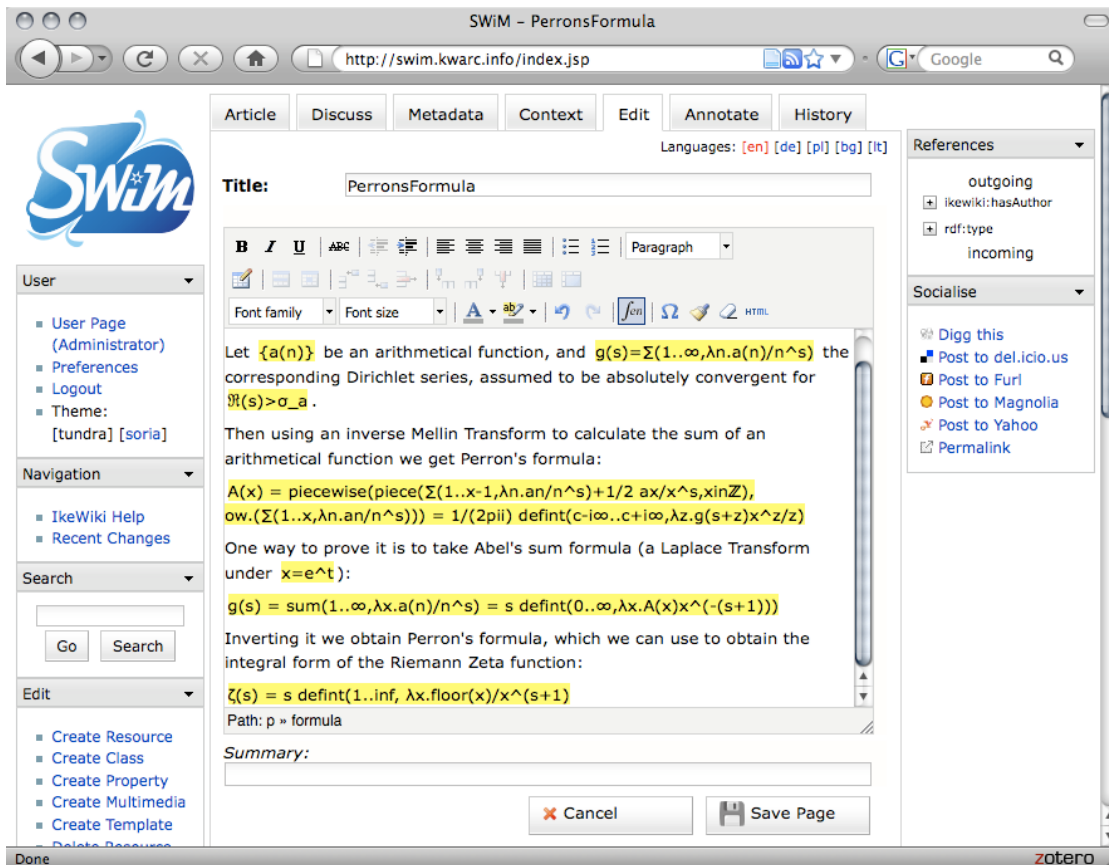


Figure 6.2: Editing a document in the extended TinyMCE, formulæ marked yellow [LGPo8]

### 6.2.4 Integrating a Formula Editor into a Visual Document Editor

In principle, formulæ could also be edited by making their semantic markup accessible as HTML tables in the manner described above. Due to the nested structure of formulæ, such a table can, however, easily fill the whole screen, where one line of linear syntax would suffice. Therefore, we have integrated Sentido [GPo6b], a dedicated OpenMath formula editor that both supports linear input and has a visual interface, as a plugin into the visual editor introduced above. Appendix C.1.1.2 describes the technical details of our implementation.

Figures 6.1 and 6.2 demonstrate formula editing with the Sentido TinyMCE plugin. In the text area of the editing widget, the formulæ are displayed in the QMath linear syntax (cf. section 6.2.1.2). When the cursor is inside one of them, a tool button is highlighted that draws the user's attention to the possibility of opening a window for visually editing the formula (cf. figure 6.2). Minor fixes, as introduced in section 6.1.2.1, can be made right in the linear syntax in the text area. The full visual editor, shown in figure 6.1, features a linear input field for the formula, a preview area where the two-dimensional Presentation MathML rendering of the formula is shown and updated in real time as the user types in the input field, and a set of collapsible/expandable palettes for inserting formula templates. Figure 6.1 demonstrates a simple linear syntax that we developed

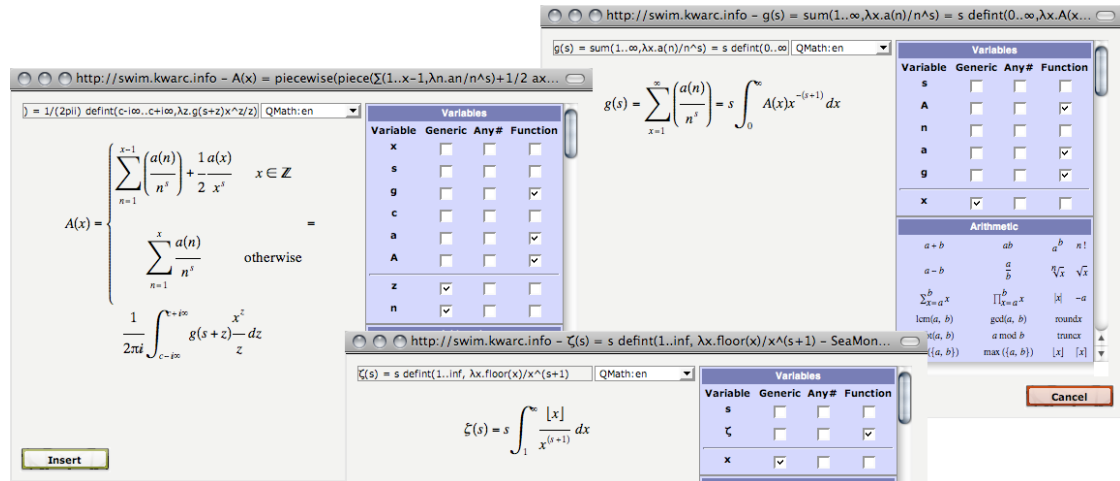


Figure 6.3: The formula editor window, when editing three different formulæ. Notice the declaration of function variables via the Variables palette and the usage of Unicode and ASCII symbols in the linear syntax [LGP08].

for OWL, simply reusing their symbol names as prefix operators, and generic linear syntax for symbols the editor does not know. Conversely, figures 6.2 and 6.3 demonstrate the use of Unicode characters defined in the linear syntax. Furthermore, figure 6.3 demonstrates an annotation facility to distinguish standalone variables from function variables applied to arguments.

Integrating such a formula editor into a visual HTML editor may require an adaptation of the above-mentioned translation between semantic markup and the markup recognized by the editor. In our case, the formula editor would have recognized formulæ in their original content markup, but we had to extend the XML→HTML translation to escape the latter in order to prevent interference with the HTML editor. I have enabled editing of content markup constructs that one formula editor does not support – e.g. the CDBase of a symbol in the case of Sentido – by adding another case to the XML→HTML translation. This case handles formulæ containing unsupported constructs and translates them to a representation different from what the first formula editor expects, so that another, complementary one can handle them.

### 6.2.5 Accessing and Editing Notation Definitions

*I do see the possibility of a fascinating future in which we don't only agree that a good notation helps, but in which we actually teach how to design notations that are geared to the manipulative data needs at hand.*

—EDSGER W. DIJKSTRA [Dij86]

This section discusses alternative interfaces for editing notation definitions as well as their maintenance in an integrated publishing environment. We have developed – but not yet integrated into the editor presented above – a linear syntax for pattern-based notation definitions, in which the notation definition from listing 2.6 would be written as follows:

Metadata Properties for cd:arith1

Property	Value	Language	Type	Action
Creator	Administrator	any	string	Delete
Date	2006-10-02	any	date	Delete
Description	This CD defines symbols for common arithmetic functions.	any	string	Delete
Resource Identifier	arith1	any	string	Delete
Language	en	en	string	Delete
Title	arith1	en	string	Delete
reviewDate	2006-03-30	any	date	Delete
revision	0	any	nonNegat	Delete
status	draft	any	string	Delete
version	3	any	nonNegat	Delete
comment	This document is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. The copyright holder grants you permission to redistribute this document freely as a verbatim copy. Furthermore, the copyright holder permits you to develop any derived work from this document provided that the following conditions are met: a) The derived work acknowledges the fact that it is derived from the work to the original source. b) The fact that the derived work is not the original OpenMath document is stated prominently in the derived work. Moreover if both this document and the derived work are Content Dictionaries then the derived work must include a different CDName element, chosen so that it cannot be confused with any works adopted by the OpenMath Society. In particular, if there is a Content Dictionary Group whose name is, for example, 'math' containing Content Dictionaries named 'math1', 'math2' etc., then you should not name a derived Content Dictionary 'mathN' where N is an integer. However you are free to name it 'private_mathN' or some such. This is because the names 'mathN' may be used by the OpenMath Society for future extensions. c) The derived work is distributed under terms that allow the compilation of derived works, but keep paragraphs a) and b) intact. The simplest way to do this is to distribute the derived work under the OpenMath license, but this is not a requirement. If you have questions about this license please contact the OpenMath society at http://www.openmath.org.	any	string	Delete

Figure 6.4: Editing OpenMath CD metadata via a form (left) and a document-like interface (right)

```
(?combinat1?binom $arg1 $arg2) |-
{lang:de,en} <mfenced><mtable><mtr><mt>$arg1</></><mtr><mt>$arg2</></></></>,
{lang:fr,ru} <msubsup><mi mathvariant="script">C</></>$arg1 $arg2</>
```

This linear syntax fully covers the OMDoc 1.3 notation definition language introduced in [section 2.4.5.2](#); its full specification is given in [\[KLM+09\]](#), [appendix C.1.1.3](#) provides technical details.

Note that editing pattern matching notation definitions in their raw XML representation also has one key advantage: Authors can copy fragments of content and presentation markup that already exist in documents and turn them into notation definitions with few editing steps – replacing concrete arguments by generic placeholders, linking corresponding matching and rendering constructs (here: *expr* and *render*), and adding further information, such as operator precedences or presentation contexts, as needed.

Declarative notation definitions (introduced in [section 2.4.5.3](#)) are usually more concise than pattern-based ones. A declarative syntax can be specified as a pragmatic syntax – and thus a frontend input syntax – that translates to a, then “strict”, pattern syntax, as has been outlined in [\[KLM+09\]](#) and is scheduled for implementation in OMDoc 1.6 (cf. [section 2.4.4.5](#)). We have not further dealt with editing declarative notations but remark that, due to their uniform structure (symbol, role, fixity, precedence, presentation markup symbol), the simpler ones among them are amenable to form-based editing.

## 6.2.6 Alternative Visual Editing Interfaces for Metadata: Documents vs. Forms

Pragmatic syntax does not only facilitate metadata editing, as discussed in [section 5.4](#), but can also be of advantage for validation, as discussed in [section 6.3.3.2](#). This section briefly introduces two visual interfaces for editing metadata in OpenMath CDs, both of which I have evaluated in the context of the “minor fixes” scenario introduced in [section 6.1.2.1](#): forms and a document-like interface.



The IkeWiki system underlying the SWiM semantic wiki, into which the document editor introduced in [section 6.2.3](#) has been integrated, ships with an RDF-based editing form for all key/value-like metadata – annotation properties and datatype properties of all ontologies known by the system – of the knowledge item currently being edited (cf. [section 9.3.1](#)). In the process of deploying SWiM to the OpenMath community, it turned out that they were not familiar with RDF’s unordered metadata model. Neither the abstract CD specification nor the XML reference encoding imposes any order on metadata, i.e. it is rather a database-like than a document-like XML language, but the CD authors preferred to treat metadata as ordered document elements.<sup>15</sup> This roots in their practice of authoring CDs as XML files with a text or document-oriented XML editor (cf. [appendix D.2.1.6](#)). I resolved the conflict by making the metadata accessible both in the form and in the document editor – not via the general-purpose table markup described in [section 6.2.3](#), but in a more lightweight way shown in [figure 6.4](#). Both editing interfaces access the metadata from the same central RDF triple store; [section 8.3.3](#) explains the translation that enables this.

## 6.2.7 Related Work

### 6.2.7.1 $\text{\LaTeX}$ Input

LaTeX2OQMath [[And](#)], which has been developed as a means of importing legacy  $\text{\LaTeX}$  content into the ActiveMath e-learning environment. It runs  $\text{\LaTeX}$ ML to obtain an intermediate XML representation of a presentation-oriented  $\text{\LaTeX}$  document, which is then post-processed by a set of XSLT stylesheets applying a number of heuristics and yielding OQMath, i.e. OMDoc XML markup with embedded formulæ in the QMath linear syntax. This is an alternative to manually annotating  $\text{\LaTeX}$  sources with  $\text{\gTeX}$ ’s semantic macros. So far, its heuristics have been optimized for algebraic geometry, and its capabilities are generally by the presentation-oriented  $\text{\LaTeX}$  syntax, which only allows for a reliable semantic interpretation of few mathematical symbols.

### 6.2.7.2 Structural Markup

The “in-place” editor of the Rhaptos system of Connexions (cf. [sections 1.4.2](#) and [2.4.8.5](#)) allows for editing the CNXML source code of each structural unit of a document in a local text area, while all other sections are shown in a rendered preview in a lighter color (cf. [figure 6.5](#)). A subset of the structural elements of the CNXML language is supported. Each supported element can be inserted before, after, or between existing structures. In contrast to the semantic TinyMCE extension presented in [section 6.2.3](#), the author first selects the kind of structure he wants to insert and then provides the content. The content of the supported elements is edited as CNXML source code, whereas an input form is offered for top-level properties.

---

<sup>15</sup>This was deemed particularly important for the *CDComment* element. In absence of a stronger metadata vocabulary and annotation syntax, it is used to hold unstructured general-purpose metadata without relevance for the abstract information model of a CD, which is why I have mapped it to *rdfs:comment* (cf. [section 3.4.5](#)). It is, however, also used in the manner of explanatory XML comments, then being placed next to the markup that is explained.



insert...

$$\binom{n}{k} = C_n^k$$

insert...

**Notational Differences**

The equation shown above demonstrates two notations for the binomial coefficient:

<list id="list1">

Type:
☐ Bulleted
Bullet [**•**]

☒ Enumerated
Arabic [1, 2, 3, ...]

☐ Stepwise
Arabic [Step 1, Step 2, Step 3, ...]

☐ Labeled Item

Title (optional):

<item>one that is common in German and English</item>
<item>and one that is common in French and Russian</item>

</list>

Save

Cancel

Delete

Help editing <list>

MathML Editor

insert...

Figure 6.5: Local CNXML source editing with Connexions’s “edit in place”

### 6.2.7.3 Integrated Document and Formula Editors

Some office word processors, as well as the  $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$  editor mentioned in [section 6.2.1.3](#), seamlessly integrate WYSIWY[GM] editing of documents and the formulæ they contain. These are usually presentation-oriented formulæ. However, the  $\text{Plat}\Omega$  extension of  $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$  allows for editing visually editing content-oriented formulæ, using symbols and notations defined earlier in the same document [[AFN+07](#)]. In addition to such standalone desktop applications, reusable document editors with integrated formula editing support have been developed as components for web environments. The Connexions MathML editor has been integrated seamlessly into the kupu widget for editing presentation-oriented HTML [[Cnxd](#)]. It has not yet been integrated directly into Connexions’s own in-place editor, but only via copy/paste.

Other web environments embed previews of mathematical formulæ into their document editors but employ special interfaces for editing the formulæ – as our TinyMCE+Sentido solution does. This is generally easier to realize than a seamless integration. Both WIRIS [[Eix](#)] and the editor of the ASsciencePad wiki<sup>16</sup> [[Jip](#)] follow this approach, using HTMLArea, another HTML editing widget. ASsciencePad displays the preview of a formula as Presentation MathML and makes its

<sup>16</sup>[Section 9.5.6](#) reviews the wiki aspects of ASsciencePad.

195

presentation-oriented source editable in a  $\text{\TeX}$ -like linear input syntax. Due to technical restrictions of TinyMCE and also of the IkeWiki environment, into which I have integrated the TinyMCE+Sentido editor, we were not able to display a Presentation MathML preview inside the TinyMCE text area. TinyMCE+Sentido is, however, similar to ASciencePad in that linear syntax can be edited directly in the text area. The WIRIS/HTMLArea integration merely displays preview images of the formulæ inside the text area. The formula itself is editable in a WYSIWYG Presentation MathML editing Java applet.

#### 6.2.7.4 Notation Definitions

OMPE (OpenMath Presentation Editor [[MLU+06](#)]), a plugin for the jEditOQMath editor mentioned in [section 6.2.1.1](#), is an editor for pattern-based notation definitions in the ActiveMath syntax (cf. [section 2.4.5.2](#)). The content markup pattern to be matched is entered in the QMath linear syntax. The presentation markup fragment can be edited in a  $\text{\LaTeX}$ -like syntax. There are a toolbar for frequently used presentational symbols and form fields for editing top-level options such as the precedence of the operator and the language context of the notation definition. With its linear input and toolbars, the OMPE interface resembles Sentido. In fact, we consider it feasible to extend Sentido for editing not just content markup formulæ but also presentation markup and then – just a bit more than a combination of both – also notation definitions. The general feasibility of such an extension path has already been proven by extending a linear syntax for OpenMath into a linear syntax for OpenMath/MathML notation definitions.

#### 6.2.8 Conclusion and Future Work

Editing the various structural types of mathematical knowledge – logical/functional, rhetorical, document, and presentational structures, formulæ, notation definitions, and metadata – in an integrated fashion has so far only been possible on the source code level. This section has demonstrated how we have extended an originally presentation-oriented, but customizable markup editor into a visual interface for editing generic semantic markup, and how we have added dedicated, specialized support for editing formulæ, notation definitions, and metadata to this editor. Thus, we have obtained an all-round editing component for integration into arbitrary web environments. As an alternative to the visual editor, the utility of  $\text{\S\TeX}$  as a general-purpose frontend input syntax for editing OMDoc – including the RDFa metadata extension from [chapter 5](#) – has been discussed.

One current shortcoming of the Sentido formula editor is that it only offers special support (i.e. linear syntax, Presentation MathML preview, and tool palettes) for a fixed set of symbols. The QMath contexts and palettes are extensible in principle, but there is no interface for dynamically extending them by, e.g., those symbols that are available in the environment that integrates the editor. RDFa also enables an extension of the annotation vocabulary, but an integrating environment has to support the import of additional RDF vocabularies. So far, our editor only supports RDFa annotations via the generic semantic markup tables. An RDFa annotation plugin for TinyMCE, which we could reuse in future, has been developed for the KiWi semantic wiki (cf. [[SS09](#)]). That plugin dynamically obtains any available RDF vocabulary from the RDF triple store of the integrating KiWi environment.

[Chapter 9](#) discusses the role of editors in integrated collaboration environments. Our TinyMCE +Sellido visual editor constitutes a core component of the SWiM semantic wiki; there, it is closely integrated into maintenance and publishing workflows. I have evaluated its usability for maintaining OpenMath CDs, focusing on the workflows for fixing minor errors in metadata and notation definitions introduced in sections [6.1.2.1](#) and [6.1.2.2](#). [Section 9.4](#) explains how SWiM supports the complete workflows, and [chapter 10](#) reports on the evaluation. Validation, an important service that is closely connected to editing and often tightly integrated into editors, is discussed in the following section.

## 6.3 Validating

Services that utilize mathematical knowledge require valid input, albeit with varying degrees of validity. Most services presented in this thesis, for example the document editor introduced in the previous section, strictly require syntactically well-formed XML input. The majority of applications presented in this thesis additionally require correct nesting (e.g. that statements are given as children of theories and not vice versa) and linking of structures (e.g. that the link from an assertion to its proof does not link to an example instead, or that the symbols used in a formula have been declared in an imported theory). Without that level of validity, it would be impossible to render mathematical objects and to publish documents, to edit formulæ, and to perform structural queries. Finally, applications dealing with highly formalized input, such as proof assistants, require a higher level of semantic well-formedness than could be guaranteed by structural checks; for example, a theorem used to justify a proof step must already have been proven. These applications are not subject of this thesis, but nevertheless we are interested in supporting the creation of content that is of a sufficient quality to be useful for them.

After the initial formal/technical justification of validation, [section 6.3.1](#) emphasizes its importance in collaborative workflows. [Section 6.3.2](#) reviews reusable state-of-the-art validation services for mathematical knowledge in more detail, from syntactic to logical validation. In [section 6.3.3.1](#), we contribute one new kind of validation to this stack by explaining how to leverage existing RDF-based metadata and link validation facilities with semantic markup for mathematical knowledge.

### 6.3.1 Validation and Division of Labor in Collaborative Workflows

Validation contributes to successful collaboration. A tight feedback loop with immediate reporting of validation errors not only helps a single author to get his edits right but also ensures that content meets minimum requirements when collaborators take over. That allows for dividing formalization or documentation responsibilities among multiple authors. One can easily imagine the following roles, which have been identified in (non-semantic) wiki communities, in a mathematical setting (cited from [[Wik](#); [Mado7](#)], comments added by the author): the regular “contributor”, who “starts small by correcting typos [recall the minor fixes scenario from [section 6.1.2.1](#)] and thrives to develop his writing skills”, the “page maintainer”, who “put[s] each page into context on the wiki by understanding and utilizing a [possibly semantic, e.g. theory] structure that will make sense and augment searches”, the “gnome” or “gardener”, who “performs small edits [such as ‘add[ing] or fix[ing] links’, or ‘improv[ing] the flow and clarity of content’] on a wiki to continually improve its overall quality”

(once more recall the minor fixes scenario!), and the “*Zen master*”, who “*makes format changes to make the wiki more visually appealing*” (think of fine-tuning mathematical notation). In [KKL10b], we have studied the task of subsequent formalization of an existing collection of documents written for a software engineering project. In that case, the formalization was done by one person different from the original document authors, but its multiple steps could as well have been distributed among further persons.

### 6.3.2 A Stack of Validation Services

In correspondence to a stack of increasingly expressive knowledge representation formalisms and languages, such as the semantic web architecture described in section 2.2, I arranged the following review of mostly state-of-the-art validation services for mathematical knowledge like a stack, from low to high complexity. (All except metadata and link validation, to which I contribute in the following section, are well established in MKM.) Usually, a knowledge item needs to satisfy one level of validity before it makes sense to validate it w.r.t. the next one.

**Syntactic validation:** The minimum requirement for semantic XML markup is that it is syntactically well-formed, i.e. parseable. Many XML parsers can be instructed to validate their input against a schema while parsing. The schemata of the languages considered here – MathML, OpenMath, and OMDoc – are specified in the RELAX NG schema language. This and other grammar-based XML schema languages cannot validate all aspects of XML syntax. An example from OMDoc are multilingual *CMP* groups: Wherever a *CMP* child is allowed, one can actually provide multiple ones, provided that they have different *@xml:lang* attribute values (cf. [Koho6b, chapter 14.1]). Validators for the rule-based Schematron language [Sch], which allows for specifying a series of XPath node tests on the input, are capable of doing that. Schematron rules can be specified independently from a grammar-oriented schema or embedded into the latter as annotations, if the schema language supports that (cf. section 2.3.2.1).

**Metadata validation:** The usage of metadata terms is usually restricted by domain (i.e. where they can be applied) and by range (i.e. what values can be given for them). Most general-purpose metadata terms can be used very liberally; for example, almost anything from a document collection down to a subterm of a mathematical object may have a *dc:description* with any string value. Some have a restricted range of values, for example *dc:date*. Metadata terms from specific vocabularies, such as classification schemes, educational metadata, or metadata about licensing and versioning, are usually restricted by range and by domain. The range restriction is obvious; for example, the Mathematics Subject Classification or the level of difficulty of a mathematical knowledge item is a value from a fixed set of strings or of class instances. The domain of these metadata terms is usually restricted by granularity. It is reasonable to provide educational metadata for small course modules (e.g. introducing one new concept) and for individual exercises, but less so for a mere symbol declaration, a single mathematical object, or for a complete book. On the other hand, licensing metadata, and versioning metadata similarly, are often provided for large units of knowledge, such as a book. If the knowledge base supports a higher modularity, they can also be given for smaller knowledge items, such as mathematical statements, but it does not make sense to provide

them for a single mathematical object, which is usually just a part of a formal mathematical statement or an informal paragraph of text.

When a markup language has a fixed vocabulary of metadata terms, their domains and ranges can be validated by an XML schema; this is, for example, the case in OMDoc 1.2. The possibility of using arbitrary metadata from RDF vocabularies, which is more recent versions of OMDoc give (cf. [section 5.2](#)), requires the RDF domains and ranges of metadata terms to be taken into account. This is similar to RDF-based link validation (see next item) and discussed in more detail in [section 6.3.3.1](#).

**Link validation:** In semantic XML markup, most links are represented by parent-child containments or URI references in attribute values (cf. [section 3.2.2.4](#)). The former kind of link can be validated against an XML schema; for the latter kind, XML schema validators can only check whether an attribute value is a syntactically valid URI, but they cannot dereference it in order to check whether the link target has the right type.

Except for a few link types with an intentionally weak semantics, such as *rdfs:seeAlso*<sup>17</sup>, the type of a link target is usually restricted in semantic markup. For example, the OMDoc markup `<proof for="#claim">` is only valid if the target, i.e. the element with the ID *claim*, is an assertion, which is usually denoted by the *assertion* element. Neither DTDs nor RELAX NG support this kind of validation. XSD can validate references to targets in the same document by its limited XPath support, but such references would have to be made by exactly reusing the ID of the target; they cannot be made as a URI reference to a target fragment. By virtue of its full XPath support, Schematron validation rules can dereference links, even if they are not directly encoded as URI references, but, e.g., as OpenMath symbol references (`<OMS cdbase="http://www.openmath.org/cd" cd="arith1" name="plus"/>`), from which URIs can be constructed by concatenation. Validating links with Schematron only works practically when the links point to fragments of documents written in the same language, but not when they point to documents in other languages, or to arbitrary resources on the Web.

An RDF-based validator handles the latter situation easily, given that all links are represented in RDF. As in-document links can equally easily be handled in an RDF representation obtained from XML by translation, RDF is preferable for link validation. There are, however, some pitfalls w.r.t. the mode of reasoning used for such validations, as discussed in [section 6.3.3.1](#).

**Structured well-formedness:** Additional well-formedness criteria apply to logical and functional structures of mathematical knowledge. The MMT system checks terms for “structural well-formedness”, which means that “*a term only uses symbols and variables that are in scope*” [[Rabo8](#)], where symbols in scope are symbols in the current theory or in imported theories. The notion of structural well-formedness is inductively extended to the statement and theory level. The MMT system automatically performs these checks on all OMDoc documents added to its knowledge base. Part of these validations have also been implemented in the JOMDoc library [[Jom](#)], as optional features.

<sup>17</sup>“used to indicate a resource that might provide additional information about the subject resource” [[BGo4](#)]

Listing 6.1: The STS type signature of the *arith1#plus* symbol

```

<OMOBJ xmlns="http://www.openmath.org/OpenMath">
  <OMA>
    <OMS name="mapsto" cd="sts"/>
    <OMA>
      <OMS name="nassoc" cd="sts"/>
      <OMV name="AbelianSemiGroup"/>
    </OMA>
    <OMV name="AbelianSemiGroup"/>
  </OMA>
</OMOBJ>

```

**Type checking:** For *terms*, there is the notion of a *type*, which can be uniquely determined in a given type system. The well-formedness of a term is defined by structural induction; for example, if there is a term  $t$  of type  $A$  and a function  $f$  of type  $A \rightarrow B$ , then the expression  $f(t)$  is well-formed and has type  $B$ . The Small Type System (STS [Dav99]), which is most commonly used with OpenMath CDs, is deliberately weak. Its two target applications are arity checking and documentation for humans implementing phrasebooks. Types that cannot be constructed with the STS are usually represented as variables with suggestive names. Listing 6.1 shows the type signature of the *arith1#plus* symbol. The addition is declared as an  $n$ -ary associative operator mapping  $n$  arguments of some type to one result of the same type. The argument and result types are not specified further. The naming of the respective variable merely *suggests* to the human reader that the intended type is an Abelian semigroup, but within the expressivity of STS there is no way of enforcing that, e.g. if there were some CD defining what an Abelian semigroup is. STS type checking has been implemented by compiling STS type signatures into RELAX NG schemata (cf. [Koho8b]), generating a small constant number of grammar rules for each type and symbol. More expressive type systems exist, usually built into functional programming languages or proof assistants based on type theory; I refer to [Rabo8] for an overview. For knowledge formalized in MathML, OpenMath, or OMDoc, such systems become accessible by translating into their native language (cf. [Koho6c]).

**Proof checking:** Proof checkers can check the correctness of a proof, given a sufficient level of formalization, making use of the logic and calculus underlying the respective system.<sup>18</sup> Checking the truth of an *assertion* is harder, as the system would first have to develop a proof. Depending on the logic, this problem is usually not decidable. Proof *assistants* follow a more pragmatic approach, where a human user supports the machine in finding a proof by giving hints. As with type checking, we have access to such systems by translation to their native language (cf. [Koho6c]).

<sup>18</sup>Note that, under the Curry-Howard isomorphism, proof checking reduces to type checking, but not all proof checker implementations follow this approach; see [Rabo8] for a detailed review.



**Human validation:** Finally, situations can occur where the desired level of automated validation is not available. This could be the case with informal content, such as natural language or presentation-oriented mathematical objects, when an automated formalization tool (cf. [section 2.1.2](#)) is not available or not powerful enough. It could also be the case with formal content, which is formalized in the “wrong” logic, i.e. when a type or proof checker for the given logic is not available and the content cannot practically be translated to a different logic. In these cases, one or more brains of human peer-reviewers can carry out the validation. Once they give a knowledge item sufficiently positive ratings, it may be tagged as “validated”.

The “flagged revisions” extension for MediaWiki [SB10], for example, supports five rating levels in the three categories “accuracy”, “depth”, and “readability” in its default configuration. This is similar to other web 2.0 environments (cf. [section 2.1.8.1](#)); more appropriate categories for mathematical knowledge could easily be added, possibly borrowed from the argumentation ontology introduced in [section 3.6.2](#) or from other guidelines for the quality of mathematical content [CS07]. An article with sufficiently high ratings in each category receives a quality marker, default possibilities being “reviewed”, “quality”, and “pristine”. In many language editions of Wikipedia, this extension is currently enabled to prevent vandalism, i.e. to maintain a minimum level of validity. A selected group of users has the right to mark a revision as “sighted” if there is no obvious vandalism; unregistered visitors see the latest sighted version by default [Metb]. Higher validation levels, such as having facts checked by domain experts, are not yet in use.

### 6.3.3 Validating Metadata and Links

Metadata from a fixed vocabulary can be validated against an XML schema. For metadata from arbitrary vocabularies and for links, or for validating semantically similar structures represented in different concrete representations (e.g. OpenMath CDs and OMDoc theories, assuming a common ontology, as discussed in [section 3.2.3.5](#)), an RDF-based validation scales better. Using RDF and OWL ontologies, one can validate whether the subject and object of each RDF triple have a type consistent with the predicate, but it is not necessarily obvious *how* to do that. In this section, I discuss both possibilities, particularly providing guidelines for applying RDF-based validation to mathematical knowledge, and finally consider their integration into an editor.

#### 6.3.3.1 Validating RDF Metadata and Links

Any OWL reasoner can validate literal objects of metadata fields w.r.t. the datatype that has been declared as the range of the respective predicate (then called *datatype property*) in the ontology, but the following catches have to be considered:

**Vocabulary Implemented in RDFS:** Many metadata vocabularies, such as those from Dublin Core, have only been implemented in RDFS; therefore, one would first have to add OWL datatype declarations as desired. Even ontologies that have been implemented in OWL do not necessarily make use of OWL datatypes. FOAF, for example, declares *rdfs:Literal* as the range of all of its datatype properties for compatibility reasons.



Listing 6.2: Inconsistent links in an OMDoc document

```

# TBox in the ontology
oo:proves rdfs:range oo:Assertion .          # (T1)
oo:Example owl:disjointWith oo:Assertion . # (T2)
# ABox extracted from document markup
# <example xml:id="e"/>
# <proof xml:id="p" for="#e"/>
<#p> a oo:Proof .                            # (A1)
<#e> a oo:Example .                          # (A2)
<#p> oo:proves <#e> .                        # (A3)

```

**No Datatype Declared:** When no explicit datatype is given for a literal, the datatype defaults to *rdf:PlainLiteral*, whose value can be any string [MPSP09]. That is, a literal whose lexical form happens to match that of a more specific datatype (e.g. the literal “2010-06-29T13:43:00” matching *xsd:dateTime*), is not automatically identified as an instance of the latter datatype.

**Poor Tool Support:** Practical support for datatype validation is poor. Tests with the OWL validator [Hor], powered by the OWL API [Owlb], a comprehensive implementation of OWL 2, showed that the examples for literal and datatype inconsistencies from the OWL specification [MPSP09] were not rejected.

**Fixed Set of Datatypes:** The OWL specification merely requires a reasoner to support the built-in datatypes of XML Schema [BM04b], but not custom datatypes.

Validating whether only admissible metadata fields are attached to a given subject, e.g. whether we are not inappropriately talking about educational properties of a mere symbol declaration, is possible in OWL. Subject validation, both for literal-valued metadata and for links, works in the same way as explained below for validating link targets, just that range is substituted by domain. However, with the mapping of OMDoc to the structural ontologies presented in chapter 3, subject validation for links is not necessary in many cases. The type of the subject is usually determined by the name of the XML element used to represent it, and the translation to RDF takes that into account when translating the attribute representing the link to RDF. For example, in a non-RDFa setting, the OMDoc markup `<proof xml:id="p" for="#claim">` translates to `<#p> oo:proves <#claim>`<sup>19</sup>, whereas `<example xml:id="e" for="#claim">` translates to `<#e> oo:exemplifies <#claim>`, i.e. using a different property. Moreover, the preceding syntactic XML validation makes sure that no two elements with the same XML ID can be declared, i.e. it would prevent the author from introducing an `<example xml:id="p"/>` in the same document as the proof. A working RDF-based validation is required nevertheless, as the introduction of RDFa into OMDoc (cf. section 5.2.1) makes the markup more flexible than XML schema validation could cope with; consider abusive annotations such as `<example about="#p"/>`.

Several approaches to checking the consistency of ontologies or their instance by increasing the expressivity of the ontology are known [Vra10, chapter 9.2]. Validating the types of link sources

<sup>19</sup> Additionally, the RDF translation generates `<#p> a oo:Proof`, but that is redundant, as it can be inferred from the ontology by `oo:proves rdfs:domain oo:Proof`.

and targets, i.e. subjects and objects of *object properties*, is one special case of this, which I have addressed with disjointness axioms. The OMDoc ontology already contains such axioms, but they could also be provided as an additional module, as discussed in [section 4.4](#).

[Listing 6.2](#) gives an example for validating link targets. To an RDFS reasoner, which does not understand (T2), this RDF graph is consistent. (A2) states that *e* is an example, and from (T1) and (A3), the reasoner infers that *e* is (also!) an assertion (cf. [BGo4]). That works differently than one might expect with an XML background, as, in contrast to an XML schema or a database schema, an RDFS reasoner assumes an open world. An OWL reasoner would make the same inference as an RDFS reasoner, but also take (T2) into account, and from that conclude an inconsistency, as *e* cannot be an instance of two disjoint classes. As, however, domain and range validation in the ABox is not one of the primary tasks an OWL reasoner has been designed for, an author cannot expect the reasoner to report that a wrong target was given for the (A3) link. From the reasoner's point of view, all of (T1), (T2), (A2), and (A3) contribute equally to the inconsistency, because removing any single one of these triples would remedy it. (However, see [section 6.3.3.3](#) for how an editing interface practically solves that problem.)

### 6.3.3.2 Closed-World Validation of Metadata with a Pragmatic XML Syntax

Validation of literal-valued metadata against an XML schema has a closed-world semantics. Such implementations exist for the fixed metadata vocabularies of OMDoc 1.2 and the OpenMath CD XML encoding. A similar service for extensible metadata vocabularies could be provided in analogy to the dynamic compilation of symbol type signatures into XML schema extensions mentioned under “type checking” in [section 6.3.2](#). Following that approach, one would compile all metadata properties from their declarations in the respective ontologies into elements of “pragmatic” XML markup, one per property, i.e. doing automatically what I have suggested as a possible manual extension in [section 5.4](#). That way, elements with missing metadata, undeclared metadata terms, terms not applicable to the current subject, or metadata with invalid values would be rejected.

### 6.3.3.3 Integrating Metadata and Link Validation into an Editor

Any inconsistencies detected when validating a document should be reported to its creator – ideally in a tight feedback loop, i.e. while the author is still editing the document. Note that instant feedback makes certain theoretically challenging validation problems easy to handle. Consider once more the OWL validation example from [listing 6.2](#): Knowledge items are usually identified before linking them, i.e. before an author creates the link from *#p* to *#e*, the proof *#p* and the example *#e* already exist, and thus the ABox facts (A1) and (A2). So far, the RDF graph is consistent w.r.t. the TBox. Now, when the author creates the offending link and receives a validation error, it can only have been caused by the link.

XML-based validation naturally integrates into editors. Depending on the quality of the error messages from the XML parser, the editor can directly point out the invalid markup. Any advanced XML editor does that, the in-place editor for CNXML mathematical markup mentioned in [section 6.2.7](#) as well. For reporting an inconsistency detected in RDF extracted from XML markup, an editor has to identify the markup location from which an error in the RDF originated. Depending on the complexity of the XML→RDF translation (cf. [sections 3.7](#) and [5.2.3](#)), this is not

always trivial. For example, the XML element with a given URI sometimes has to be located by decomposing the URI, as is the case with OpenMath symbols or MMT names. Or consider RDFa: With prefixes and terms, it supports various ways of abbreviating URIs, and annotations about one subject can be made in multiple places within a document.

More restrictive editing interfaces hold further potential for validation. When designing a frontend input syntax, such as  $\text{\LaTeX}$ , one can make a larger number of invalid constructs generate parsing errors. For example, the RDFa-compatible metadata extension being developed for  $\text{\LaTeX}$  (cf. [section 6.2.2.1](#)) reports metadata terms attached to resources out of their declared range as errors. Well-designed graphical user interfaces can even preclude the creation of invalid content completely. For example, the IkeWiki metadata editing form mentioned in [section 6.2.6](#) only offers the user to add properties whose domain matches the type of the resource being edited.

### 6.3.4 Related Work

This section discusses approaches related to the validation of metadata and links described in [section 6.3.3](#).

#### 6.3.4.1 Integrity Constraints for OWL

YUAN FANG LI et al. have suggested a method for automatically adding disjointness axioms and other restrictions to OWL ontologies for the purpose of efficient validation [[LSD+06](#)]. STEPHAN GRIMM and BORIS MOTIK have investigated an extension of OWL by autoepistemic operators for closed-world reasoning [[GM05](#)]. The latter would also allow for checking whether certain facts are asserted right in the local ontology or RDF graph, and not merely assumed to exist somewhere in the open world; see [sections 3.2.2.2](#) and [3.4.2](#) for a discussion of examples in the context of OMDoc. In an alternative approach, JIAO TAO et al. have developed an closed world and unique name semantics for OWL [[TSB+10](#)]. Further examples are mentioned in [[Vra10](#), chapter 9.2].

#### 6.3.4.2 DRa Validation in MathLang

In the context of MathLang, a validation of the DRa, i.e. statement-level logical structures and document structures, has been implemented (cf. [sections 2.4.6](#) and [2.4.10.2](#) and [[Ret09](#); [KMR+07](#)]). A document is validated in two steps: First, the DRa annotations made in terms of the DRa ontology are extracted to a “dependency graph” (DG); then, a graph of logical precedences (GoLP) is obtained from the DG by generalizing some of the relations.<sup>20</sup> The ontology is, however, only used for partly formalizing and implementing the DRa vocabulary. For example, the validation process has a notion of provable and unprovable statement types, which has not been formalized in the ontology. The validation process is not technically based on the ontology and therefore not easily extensible to validating other structural aspects of a MathLang document. The ontology has not been designed for extensibility either, as discussed in [section 2.4.10.2](#).

---

<sup>20</sup>In the terminology introduced in [section 2.1.6](#), the DG is not a dependency graph but rather a graph of various logical statement-level relations, whereas MathLang’s notion of “logical precedence” qualifies as a special kind of dependency.

### 6.3.4.3 Temporal Validation Along Narrative Paths

An alternative approach to checking the consistency of semi-structured documents, which is more flexible and more powerful than both MathLang and the OWL-based approach discussed in [section 6.3.3.1](#) in that it takes multiple narrative paths through a document into account, has been followed by FRANZ WEITL et al. [WJF09]: As in MathLang and my approach, structures from the document are extracted to a graph, using a DL ontology as a vocabulary. In addition to general background knowledge about the document model and the domain of discourse, users can define additional constraints. The validation is performed using a specialized temporal description logic, which is more expressive than OWL but still decidable.

### 6.3.5 Conclusion

In a collaborative knowledge base, validation is a prerequisite for dividing responsibilities among collaborators and for enabling other services and applications to use knowledge items. This section has provided an overview of a stack of validation services for mathematical knowledge. I have particularly discussed the validation of RDF graphs extracted from semantic markup. This will become applicable in integrated knowledge bases holding both XML and RDF representations of mathematical knowledge (cf. [chapter 8](#)).

## 6.4 Human- and Machine-Comprehensible Publishing

Publishing mathematical knowledge addresses the goal of making mathematical knowledge comprehensible, both to human and machine audiences, so that they understand how to reuse and apply it. [Section 6.4.1](#) discusses how to publish mathematical knowledge originally represented in OMDoc or OpenMath CDs as linked data in RDF or in their original representation. The pure publication of semantic representations as linked data primarily targets machines, but there are also human-oriented interfaces for browsing RDF linked data. Human-comprehensible XHTML+MathML documents can also be published in a way that preserves their complete semantic structure, as mathematical content markup and RDFa, which enables assistive services to hook into the documents. While [chapter 7](#) presents such services and their integration, [section 6.4.2](#) explains the process of publishing XHTML+MathML+RDFa documents.

### 6.4.1 Publishing Linked Data

Publishing knowledge on the Semantic Web in conformance with the linked data principles (cf. [section 2.3.1](#)) makes it accessible and interpretable by services in a straightforward way. The most common format for publishing linked data is RDF. When the original knowledge is represented differently – e.g. in a relational database or as XML markup – RDF has to be extracted from it in the two steps explained in [section 3.7](#): minting URIs for entities, and a structural translation to RDF graphs. With the ontologies and translations presented in [chapter 3](#), we are now able to do that with mathematical markup. This section explains a proof of concept by which we have shown how statistical datasets, one typical web of data application, benefit from the availability of mathematical knowledge as linked data. Then, I briefly review human-oriented state-of-the-art

Listing 6.3: Number of patients waiting for hospital operation from zero to one week in South Tyneside (from [OKP+10], abridged)

```

:ds1_1.2                                # just some ID for this data point
  a scv:Item ;
  scv:dimension :w0to1week ; # the type of items counted (patients waiting 0 to 1 week)
  scv:dimension :A_RE9 ;      # the "region" dimension (South Tyneside)
  scv:dimension :TP2008_09 ;  # the "time period" dimension
  rdf:value 185 ;            # the count
  scv:dataset :ds1 .         # back-reference to the whole dataset

```

interfaces for browsing linked data, and discuss problems that remain own work on deploying knowledge from OMDoc and OpenMath CDs as linked data.

#### 6.4.1.1 Mathematical Semantics of Statistical Datasets

Statistical datasets published as linked open data currently lack mathematical semantics. We have proposed grounding their semantics in OpenMath CDs. This section introduces this use case from the statistical dataset point of view and argues why the OpenMath CDs themselves also need to be published as linked data. The following section discusses the technical implications of that requirement on OpenMath.

Section 1.5.2 has introduced the linked open datasets generated from the statistical data collected by the US and UK governments. In their current state, these datasets contain a lot of data points without making their origin semantically explicit, as shown in listing 6.3.<sup>21</sup> We have proposed SCOVOLink, an extension of SCOVO (cf. section 3.5.2), which allows for semantically grounding data points, saying, e.g., “the items that we have counted here are patients – e.g. referencing <http://dbpedia.org/resource/Patient> – by waiting duration, region, and year” (cf. [VLH+10]).

Mathematical knowledge becomes relevant when modeling *derived values*, such as the hospital density of a region in a given year, defined as the number of hospitals divided by area.<sup>22</sup> At the moment, there are a lot of derived values in the datasets published, simply given as additional raw data points, without provenance information referring to derivation rules. For a client consuming these data, there is no way of verifying their correctness. Applying the same derivation rule to new or changed base values or to aggregated sets<sup>23</sup> is not possible either, because the derivation rule is not made explicit.

As RDF is not particularly suitable for fully representing mathematical objects (cf. section 2.4.10), and as there are OpenMath-aware computation services on the Web (cf. section 1.4.3.3), we have proposed a division of responsibilities between RDF and OpenMath. Standard statistical functions as well as custom derivation rules are assumed to be defined in OpenMath CDs. Using the SCOVOLink vocabulary, every data point can link to the function that has been applied to the

<sup>21</sup>One could expect such explicit information to be given for the dimensions that the data points link to, but this is not currently the case.

<sup>22</sup>Another example from published datasets is the average number of jobs per citizen in a particular region.

<sup>23</sup>For example, the combined unemployment rate for two regions cannot simply be computed by adding the two individual unemployment rates, as they are weighted by population.

Listing 6.4: Modeling the derivation of the population density of Germany in SCOVOLink

```
@prefix ex: <http://www.example.org/> .

# the population density is computed by ...
ex:population-density-germany-2010 sl:computedFrom [
  # ... applying arith1#divide
  sl:function <http://www.openmath.org/cd/arith1#divide> ;
  sl:arguments
    # ... to the population
    [ sl:argPosition "1"^^xsd:int ;
      sl:argValue ex:population-germany-2010 ] ,
    # ... and the area
    [ sl:argPosition "2"^^xsd:int ;
      sl:argValue ex:area-germany-2010 ] ] .
```

values of other data points, which are passed as arguments, in order to derive its value.<sup>24</sup> Listing 6.4 gives an example. For (re)computing the derived value using an OpenMath-aware service, a linked data client has to translate this RDF annotation to an OpenMath object. A call to an *sl:function* with *sl:arguments* translates to an OMA. For each data point referenced by *sl:argValue*, its *rdf:value* is obtained and represented as an OMI or OMF. Alternatively, SCOVOLink supports named arguments (*sl:argName*), which correspond to the names of variables used in the symbol definition of a function in a CD (see below), and unordered argument lists for functions on sets, such as *s\_data1#mean* (the statistical mean); see [VLH+10] for examples.

Translating SCOVOLink annotations using named arguments to OpenMath objects (which do not know named arguments!) requires access to the respective definition in a CD to get their order right. This is one reason for *also publishing the CDs as linked data*. The other reason is that OpenMath computation services and thus phrasebooks are developed independently from datasets being published, but that datasets may use non-standard derivation rules that existing phrasebooks do not support. While standard CDs, such as *arith1* or *s\_data1*, are usually supported by the built-in phrasebooks of OpenMath-aware services, OpenMath CDs give the freedom to introduce arbitrary new functions. Suppose a dataset contains the Human Development Index (HDI) of a country [Wikioa]. Assuming that the four required auxiliary data points have already been computed (*LE* = life expectancy index, *ALI* = adult literacy index, *GEI* = gross enrollment index, and *GDP* = an index computed from the gross domestic product per capita at purchasing power parity, all normalized to a scale between 0 and 1), the *HDI* is defined as  $\frac{1}{3}(LE + \frac{2}{3}ALI + \frac{1}{3}GEI + GDP)$ . When dataset publishers define this derivation rule in a CD accompanying the statistical dataset, e.g. <http://example.org/statistics>, derived data points can be annotated in analogy to listing 6.4. For (re)computing an HDI data point derived from four other data

<sup>24</sup>Providing such information not per data point but per dataset would avoid a lot of redundancy. Recall, however, that the semantics of linked data vocabularies is usually intentionally weak in order to keep reasoning scalable. Dataset-level derivation rules would involve universal quantification over data points, requiring more powerful clients and query engines, and might therefore not work as universally as semantically lightweight annotations of individual data points.



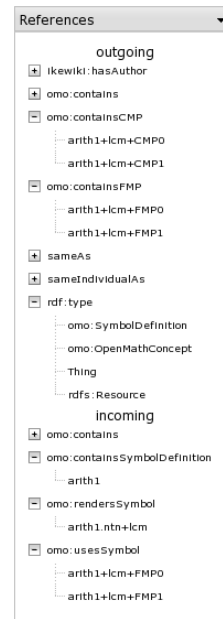


Figure 6.6: Navigation links for an OpenMath symbol definition

points containing the *LE*, *ALI*, *GEI*, and *GDP* values, a client would download the definition<sup>25</sup> of the <http://example.org/statistics#hdi> symbol from the CD, expand the mathematical expression using the definition, and then send that expanded expression, which only uses operators from the universally understood *arith1* CD, to the computation service.<sup>26</sup>

#### 6.4.1.2 Generic Interfaces for Browsing Linked Data

A user interface for browsing has to answer the following three questions to a user in every situation: “Where am I? What’s here? Where can I go?” [Vee01] User interfaces for browsing RDF try to answer these questions by displaying all local information about one resource (i.e. its literal-valued metadata) and allow for exploring the neighborhood by traversing outgoing or, as far as they are known, incoming links. This style of browsing is, for example, employed by the IkeWiki semantic wiki [Scho6]. Figure 6.6 shows the linked data style navigation tree for an OpenMath symbol, as it occurs in the IkeWiki-based SWiM system (cf. chapter 9). Most of the answers for the questions “Where am I?” and “What’s here?” are, at the same time, given in the main area of the browser window, where a rendering of the symbol definition is displayed. Figure 9.1 on page 291 shows a

<sup>25</sup>By saying “the definition”, I assume, in the absence of support for proper definitional *FMPs* (cf. section 2.4.3), that there is one *FMP* of the form  $\text{@}(\text{relation1}\#eq, \text{@}(HDI, LE, ALI, GEI, GDP), (\frac{1}{3}(LE + \frac{2}{3}ALI + \frac{1}{3}GEI + GDP)))$ . Overloading certain vocabulary terms with semantics that has not been specified for them – here: interpreting an *FMP* as a definition – is known from the practice of RDF linked data. For example, the *rdfs:seeAlso* property is semantically very weak (“used to indicate a resource that might provide additional information about the subject resource” [BG04]), but in linked data settings it is commonly assumed to point to a URI that provides further linked data compliant information. Conversely, *owl:sameAs* is commonly used to declare that two things, despite having different URIs, are the same (cf. [MPSP09]) – but hardly any linked data application makes use of OWL reasoning. Other practical applications of OpenMath CDs make similar assumptions of *FMPs*; cf. section 7.6.3.

<sup>26</sup>Here, we assume that those values, from which the HDI is computed, are either hard-coded in the dataset, or that they have been computed before, using the same method.



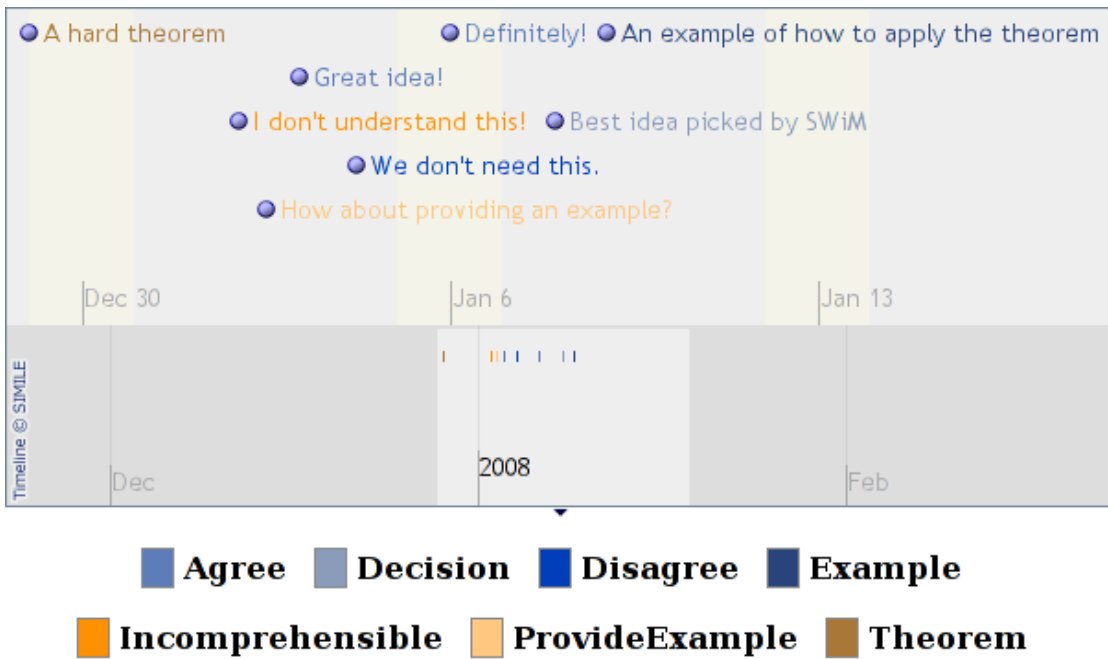


Figure 6.7: The Exhibit timeline view of [figure 9.7](#) on page 299, starting at a hard theorem and ending at a helpful example.

complete screenshot of SWiM, with the “references” navigation tree on the right side. This way of navigation is employed in SWiM’s realization of the “minor fixes” workflows introduced in sections 6.1.2.1 and 6.1.2.2, as explained in [section 9.4.1](#). ZUZANA NEVĚŘILOVÁ has applied a similar navigation interface, displaying the local neighborhood in a graph view, to a digital library of mathematical documents [[Nev10](#)].

Note that both of these implementations require local access to the RDF graph via custom interfaces. That is, they would not work on arbitrary linked data published on the web. The same *principle* of browsing can, however, also be employed for browsing arbitrary RDF that has been published as linked data. We have developed such a generic navigation service for linked data embedded into documents as RDFa (cf. [section 7.5.3](#)). Or consider, for example, the Tabulator browser [[BLCC+06](#)] and the widgets of the Exhibit framework [[HKM07](#)], which visualize linked data from arbitrary sources. [Figure 6.7](#) shows the RDF representation of a discussion thread in terms of the argumentation ontology introduced in [section 3.6.1](#), displayed by Exhibit’s timeline widget, and demonstrates a datatype-aware interface, which arranges multiple resources by their values of a property of type *xsd:date*.

#### 6.4.1.3 Remaining Challenges to Publishing OMDoc and OpenMath Linked Data

[Section 6.4.1.1](#) has introduced a use case for publishing mathematical knowledge as linked data. In my work on publishing OMDoc as linked data and extracting RDF from OpenMath CDs (without publishing it as linked data yet), I have identified three problems. They remain to be addressed

before large scale linked data publishing will work in practice: specifying MIME types for content represented in mathematical markup languages, stopping bad authoring practices, and working around restrictions in the URI formats of these languages.

**MIME Types:** The use case described above required a combination of OpenMath and RDF. OpenMath is suitable for representing derivation rules, RDF for interlinking statistical data; OpenMath is comprehensible by computational web services, RDF by linked data browsers. Therefore, mathematical knowledge should be published both in its original markup language and as RDF – and in a human-comprehensible XHTML+MathML rendering (cf. [section 6.4.2](#)). It is best practice to make all representations of a resource available from the same HTTP URI, as URIs should identify things and not concrete encodings of descriptions of things (cf. [section 2.3.1](#) and [\[BCH07\]](#)). The client indicates the desired encoding by *HTTP content negotiation*, i.e. requesting a particular MIME type in the *Accept* header of its HTTP request (cf. [\[SCo8\]](#)). *application/omdoc+xml* has been specified as a MIME type for OMDoc [\[Koho6b, chapter 12.1\]](#), and *application/mathml+xml* for MathML<sup>27</sup> [\[ABC+10, chapter 6.2.3\]](#). OpenMath does not yet have an official MIME type, but *application/openmath+xml* for OpenMath objects and *application/openmath-cd+xml* for OpenMath CDs have been proposed. There are several encoding choices for serving an OpenMath CD, depending on what the client understands best. As an alternative to using the OpenMath CD markup, it could be served as a pure OpenMath object, encoded using the “meta” CD [\[BCC+04, appendix A.1\]](#), or it could be served as RDF, where OpenMath objects are represented as XML literals, as discussed in [section 3.2.3.5](#).

**Bad Authoring Practices** – from a linked data point of view – result from authors of mathematical XML markup using URIs wrongly or not at all. Hardly any community-contributed OpenMath CD specifies a *CDBase* or references symbols by absolute URIs, which indicates a lack of awareness. The fallback base URI <http://www.openmath.org/> is, even independently from linked data considerations, not suitable for CDs that developers did not explicitly contribute to the [openmath.org](http://www.openmath.org) site. Finally, authors who are aware of CDs and symbols having URIs usually merely consider them globally unique *names* without relevance for retrieving information about these resources [\[Lan10\]](#).

**URI Format Restrictions:** The OpenMath 2 and OMDoc 1.2 specifications are older than the linked data idea. They use URIs, but mostly as a means of identifying things, not for retrieving them. Besides widespread bad authoring practices, the languages impose restrictions on URIs that complicate the integration of knowledge represented in these languages into the Web of Data. The remainder of this section discusses how to work around them.

OpenMath’s *cdbase/cd#name* URI format, which has also been adopted by Strict Content MathML, complies well with linked data practices – unless CDs grow large. Resolving fragments after the # (hash) in a URI is up to the client [\[BLFM05\]](#). Consequently using hash URIs for OpenMath symbols therefore forces clients to always download a complete CD, in which they could then locate the desired symbol. For a large CD, of which a client is only interested in few symbols, it would be more efficient to use “slash URIs”, such as *cdbase/cd/name* or the similar MMT URIs to

<sup>27</sup>or, more specifically, *application/mathml-presentation+xml* and *application/mathml-content+xml*

be introduced in OMDoc 1.6 (cf. [section 2.4.4.1](#)). With RDF linked data, this choice is up to the data publisher (see, e.g., [BPo8b]). When publishing OpenMath CDs as RDF, one could work around this restriction by setting up server-side redirects. On a client's initial request for *cdbase/cd#name* – i.e. actually *cdbase/cd* – the server could return an RDF graph that contains no information but *rdfs:seeAlso* links from each hash URI in the CD to a corresponding slash URI, from which the client would then be able to retrieve the desired fine-grained information.

A similar problem exists in OMDoc 1.2/1.3, whose symbol URIs are formed from the fragment IDs of the symbol declaration elements but referenced by *cdbase*, *cd*, and *name* from outside – for example from a web document with MathML objects annotated using Strict Content MathML<sup>28</sup> or OpenMath –, and any client processing such a reference would reasonably construct a *cdbase/cd#name* URI from it, according to the OpenMath 2 standard. A possible way to give access to the symbol declaration from the latter URI is:

- using the URL of the OMDoc document as *cdbase* – preferably without the “.omdoc” extension, as URIs of things should not reflect concrete encodings (“cool URIs”, cf. [SCo8]),
- using the name of the theory as *cd*, as that conforms to the practice of using OMDoc, and
- using the name of the symbol as *name*. The name has to be the same as the fragment ID of the symbol declaration, as the client, after retrieving a document from the server, has to resolve that part of the URI. For documents containing multiple theories – which is possible in OMDoc<sup>29</sup> –, this means that symbol names must be chosen uniquely within the document – as the same constraint applies to fragment IDs –, whereas the OMDoc semantics only requires them to be unique within a theory.

The server would then have to be instructed to redirect *cdbase/cd* to *cdbase*, i.e. the URL of the OMDoc document – or its RDF or XHTML or any other representation. In contrast to the “hash→slash” redirect mentioned above, this redirection approach works for any representation format. The MMT URIs to be introduced in OMDoc 1.6 do not cause problems with linked data deployment, as they are, like slash URIs, entirely resolved on the server. However, by the same argument as used for OMDoc 1.2, a redirect as described for OpenMath above should be set up for clients that are not aware of how to construct an MMT URI from a *cdbase*, *cd*, and *name* that they find in an annotation.

A final problem with OpenMath CDs is that only dictionaries and symbols can have URIs, whereas for other entities there is not even the possibility to give them URIs. An XML→RDF translator might generate some, as required in [section 3.7](#), but an agent interested in retrieving XML representations would also need them. As a use case for such fine-grained links, consider the DLMF mentioned in [section 1.4.1](#), which contains a large number of equations describing or defining mathematical functions. A linked data version of the DLMF could link these to the corresponding *FMPs* in OpenMath CDs.

<sup>28</sup>Note that the non-strict *csymbol/@definitionURL* attribute supports unrestricted URIs. However, a client processing such a link cannot expect it to point to a CD; it could as well point to a description that is merely comprehensible to humans. Therefore, strict links are preferable when linking to CDs.

<sup>29</sup>In August 2010, 40 out of the 1959 OMDoc 1.2/1.3 documents in MICHAEL KOHLHASE's lecture notes (cf. [sections 2.4.7.2](#) and [6.1.2.4](#)) contained more than one theory.

**Linked Data Publications in Progress:** As a first proof of concept, we have published MICHAEL KOHLHASE’s lecture notes as a linked dataset in OMDoc, XHTML+MathML+RDFa, and RDF (via a SPARQL endpoint) in order to serve the information needs of learners and instructors, as explained in [section 6.1.2.4](#); a demo is available at [\[Koh+a\]](#). [Section 6.4.2.8](#) explains the techniques employed for annotating the human-readable documents, [section 7.5](#) explains how the linked data annotations are utilized for interactive browsing, and [section 8.1](#) covers the OMDoc→RDF translation. [Section 11.3.4](#) summarizes our current progress towards publishing further collections of mathematical knowledge as linked *open* data.

## 6.4.2 Publishing Documents Comprehensible for Humans and Assistive Services

When publishing human-readable documents generated from semantic representations, our goal is not just a high printing quality. We conceive reading as a process of understanding a document by interacting with it. If that process is to be supported by interactive services, the mathematical knowledge contained in published documents has to be made available to assistive services in a representation they understand, so that they can adapt the presentation of the document to a user’s needs or provide him with further information without switching away from the document. The HELM project (cf. [section 1.4.3.1](#)) was probably the first to systematically investigate both possibilities of such “*semantics-aware transformations*” [\[Scho2\]](#), which I, more precisely, call *semantics-preserving transformations* here: creating “backwards pointers” to a machine-comprehensible knowledge representation stored externally of the document, and generating “*annotated documents containing the associated content-oriented information*”.

This section restates the requirements for such transformations in the face of the progress that mathematical knowledge representation and semantic web technology have made since then and discusses how to address these requirements. The semantic representation is assumed to be given as markup.<sup>30</sup> While focusing on human-comprehensible documents generated from a semantic representation, most of the annotation recommendations made in this section also hold for settings where the annotated human-comprehensible document is the only representation of the respective knowledge.

### 6.4.2.1 General Requirements for Semantics-Preserving Transformations

[Section 6.4.1.3](#) has mentioned the possibility of publishing human- and machine-comprehensible information at the same “cool URI”, so that a service can retrieve the latter by HTTP content negotiation. For interactive services to work in the human-comprehensible document, that scenario requires a minimum amount of machine-comprehensible information in a published document, so that client-side applications know what parts of the document they can anchor services to. That is, for all semantically relevant fragments of a document, they have to know at least where they can obtain the corresponding machine-comprehensible information from. In settings where

---

<sup>30</sup>One can also present RDF representations to humans, using the Fresnel display vocabulary [\[BBG+08; PBK+06\]](#), which is inspired by XPath and CSS. However, not all structures of mathematical knowledge can reasonably be represented in RDF, as explained in [section 2.4.10](#). Due to the complexity of mathematical structures and its relatively low expressivity – compared to, e.g., XSLT –, Fresnel is not used in this thesis.

too many round-trips to a server are not desirable, e.g. for performance or security reasons, or where publishers want semantics-enabled web crawlers to index the documents, or in archival scenarios, it may even be desirable to create compound documents that contain a full human- and machine-comprehensible representation of the same knowledge.

Thus, we can phrase requirements as follows:

1. In a human-comprehensible document generated from a semantic representation, any fragment that corresponds to an entity in the semantic representation **MUST** be linked to the latter.

When the semantic representation is available via content negotiation from the same URI as the human-comprehensible document, fragments in the human-comprehensible document **SHOULD** use the same IDs as their semantic counterparts. Even in the presence of alternative linking mechanisms, this is **RECOMMENDED**, as it also facilitates looking up the human-comprehensible presentation that corresponds to a given fragment of a semantic representation.

2. Additional semantic annotations **MAY** be embedded into the human-comprehensible document. They **MUST NOT** contradict the original semantic representation(s).

The following subsections discuss how these requirements can be addressed in the XHTML+MathML+RDFa output format and very briefly mention other output formats, such as PDF.

#### 6.4.2.2 Preserving Object-level Structures as Cross-Linked Parallel Markup

When mathematical objects originally represented in OpenMath or Content MathML are rendered to Presentation MathML, the result can be annotated with the original content markup (cf. [section 2.4.2](#)). Suppose we had a rendering algorithm  $\rho: C \rightarrow P$  creating presentation-only markup from content markup. This algorithm can easily be extended to an algorithm  $\rho'$  that produces parallel markup, as shown on the right. The utility of parallel markup can be greatly enhanced by adding fine-grained cross-links between both representations, covering the subterm structure down to individual symbols. MathML does not prescribe the direction of these links (cf. [\[ABC+10, chapter 5.4\]](#)), but as we are mainly interested in accessing the semantic structure of a mathematical expression from its human-comprehensible presentation, we require that they point from presentation to content markup. Besides being most intuitive in our case, it has the particular advantage that it works for  $n$ -ary operators. Consider the content markup  $\text{@}(\text{arith1}\#\text{plus}, a, b, c)$  and its rendering  $a + b + c$ . The *arith1#plus* operator occurs once in the content markup but  $n - 1$  times in the presentation markup. As XML links are injective – i.e. multiple links can have the same target, but from the same origin there can be at most one link – and established by giving potential link targets unique IDs (e.g. *expr1*) and having the link sources point there (e.g. via `xref="#expr1"`), the *arith1#plus* operator can only be cross-linked when giving its content representation an ID and letting the rendered presentations point there.<sup>31</sup>

$$\rho': c \mapsto \begin{array}{l} \text{<semantics>} \\ \quad \rho(c) \\ \quad \text{<annotation-xml>} \\ \quad \quad c \\ \quad \text{</annotation-xml>} \\ \text{</semantics>} \end{array}$$

<sup>31</sup>The MathML specification does not prescribe a direction for these cross-references, for good reasons. “In absence of other criteria [which we, however, have!], the first branch of the semantics element is a sensible choice to contain

Our further work with parallel markup relies on the rendering algorithm implemented by the JOMDoc library (cf. [appendix C.1.2](#)). As I did not develop that algorithm, I will not provide a full description of how it creates cross-linked parallel markup, but emphasize its two most challenging aspects: (i) For every content element to be cross-linked, a unique ID has to be generated. (ii) All sources and targets of cross-links have to be identified both in the presentational fragment and the content markup pattern of a pattern-based notation definition (cf. [section 2.4.5.2](#)).

A prerequisite for cross-linked parallel markup of subterms is that subterms be marked up in the rendered presentation. From a purely visual point of view, this is not required, as the experienced human reader knows how to read brackets and how strong operators bind when no explicit brackets are used; i.e. there is no need to group the  $2x$  in  $2x + 1$ . For a machine, however, that wants to translate a user's selection of  $2x$  back to the corresponding content markup object  $@(\text{arith1}\#\text{times}, 2, x)$ , this subterm needs to be marked up. Presentation MathML supports that by the invisible *mrow* element, which groups its content into an explicit subterm. Certain other Presentation MathML elements have this grouping property: the constructors for fractions, radicals, super-/sub-/under-/overscripts, table cells, and a few others; see [[ABC+10](#), table 3.1.3.2] for a complete list.

From a single presentation markup element, the corresponding content markup element can simply be looked up by traversing the *@xref* link. For a range of presentation markup selected by the user – e.g. using the mouse –, this is less trivial, unless the user interface restricts the possible selections the user can make to subterms. One solution is to locate the closest common ancestor of all selected presentation elements<sup>32</sup> that carries an *@xref* attribute and traverse that link ([algorithm 6.2](#)). [Figure 6.8](#) gives an example.

Since the JOMDoc rendering algorithm supports pattern-based and thus potentially non-compositional notation definitions (cf. [section 2.4.5.4](#)), not every content markup subterm corresponds to a presentation subterm. For example, in the presentation element corresponding to  $\sin^2 x$ , there will be no contiguous subexpression pointing to the content expression  $\sin x$ .

### 6.4.2.3 Alternatives to Parallel Markup: Attributes and Switchable Alternatives

All structural information about a mathematical expression *can* be preserved by consequently using cross-linked parallel markup, as described so far. However, certain types of information may not be conveniently or efficiently accessible to software. I am aware of the following two cases:

---

*the id attributes. Applications that add or remove annotations will then not have to re-assign these attributes as the annotations change.*” [[ABC+10](#), section 5.4] Moreover, *our* choice of direction has solely been influenced by the injectivity requirement mentioned above; the MathML specification emphasizes that “*the direction of the references should not be taken to imply that sub-expression selection is intended to be permitted only on one child of the semantics element. It is equally feasible to select a subtree in any branch and to recover the corresponding subtrees of the other branches.*” [[ABC+10](#), section 5.4]

<sup>32</sup>Due to the ordered tree structure of XML, this can be realized by looking up the closest common ancestor of the start and the end nodes  $s$  and  $e$  of the selection. Proof sketch by induction over the tree depth: For a tree of depth 0, i.e. just containing one node  $r$ , both root and leaf, we have  $s = e = r$ . For a tree of depth  $n + 1$ , whose root  $r$  has  $k$  children,  $c_1, \dots, c_k$ , we only have to consider a selection that contains at least two nodes  $n_1$  and  $n_2$  in two different branches; otherwise, one of the  $c_1, \dots, c_k$  would already be a common (not necessarily closest) ancestor of  $n_1$  and  $n_2$ , by the induction hypothesis. Due to the acyclicity of a tree,  $r$  is the closest common ancestor of  $n_1$  and  $n_2$ . Due to the definition of the document order,  $s$  and  $e$  in particular are in two different branches and therefore have the closest common ancestor  $r$   $\square$



---

**Algorithm 6.2** Determine the smallest content markup term that corresponds to a set of selected presentation markup elements (see [BFM+10] for the XML concepts used)

---

**Require:**  $S$  is a set of presentation markup elements in an XML document/fragment  $d$

**Ensure:**  $c$  is a content markup element, or  $\perp$  (nothing)

```

 $s_{\min} \leftarrow \min_d(S)$  // minimum w.r.t. the document order of  $d$ 
 $s_{\max} \leftarrow \max_d(S)$ 
 $e \leftarrow \text{cca}(s_{\min}, s_{\max})$  // closest common ancestor  $\text{cca}(n, m) := \min_{i,j} \{c \mid \langle n, c \rangle \in P^i \wedge \langle m, c \rangle \in P^j\}$ ,
where  $p$  is the XML parent accessor and  $P$  is the relation  $\{\langle n, p(m) \rangle \mid n \text{ is a node in } d\}$ 
while not ( $\pi_B(\$e/@xref)$  or  $e = r$ ) do //  $\pi_B$  = XPath evaluation function returning a boolean
result,  $r$  = root node of the XML document
     $e \leftarrow p(e)$ 
end while
 $t \leftarrow \perp$  // target of the presentation  $\rightarrow$  content link (@xref)
 $x \leftarrow \pi_S(\$e/@xref)$  //  $S$  = string datatype
if  $x \neq \varepsilon$  then //  $\varepsilon$  = empty string, also returned when  $e$  does not have an @xref attribute
     $t \leftarrow \pi_D(\text{doc}(\$x))$  //  $D$  = XML document node datatype; we dereference the URI  $x$ 
    if  $t \neq \perp$  then //  $x$  pointed to a document/fragment, whose (invisible) document node is  $t$ 
         $c \leftarrow \pi_N(\$t/*[1])$  // the (only) root element node of  $t$ ; an actual implementation would
        combine the previous steps into  $\pi_N(\text{doc}(\$e/@xref)/*[1])$ 
        if  $c$  is not a content markup element then
             $c \leftarrow \perp$ 
        end if
    end if
end if
return  $c$ 

```

---

(i) structural information that is not trivially or not at all available from the original content markup but required during interaction with the presentation markup, and (ii) subterm annotations that an application chooses to visualize as switchable alternative displays. The following paragraphs elaborate on this.

An example of structural information that is not part of the content markup is information about the precedences of operators. The renderer needs it for correctly generating brackets (cf. section 2.4.5.5), but an interactive service that dynamically displays and hides redundant brackets in a rendered document (cf. section 7.4.2.1) can also draw on it. Such information could, in principle, be represented as attributions to the occurrences of the operators in the content markup output by the renderer. But then it would only be accessible from a presentation-oriented interface via two indirections: first content markup lookup, as shown above, then content attribution lookup. Such information can be provided in a more lightweight way by attaching custom XML attributes from a non-MathML namespace directly to Presentation MathML elements (cf. [ABC+10, chapter 2.3.3]).

Descriptions, labels, or natural language abbreviations for subterms are further cases of information that might be displayed or hidden on demand. In the content markup input, they can be provided as attributions to subterms. One way of presenting them to the user is allowing him to interactively switch between the formal subterm and its informal abbreviation. Presentation Math-



```

<semantics>
  <!-- a+b2c -->
  <mrow xref="#E">
    <mi xref="#E.1">a</mi>
    <mo xref="#E.0">+</mo>
    <mrow xref="#E.2">
      <msup xref="#E.2.1">
        <mi xref="#E.2.1.1">b</mi>
        <mn xref="#E.2.1.2">2</mn>
      </msup>
      <mo xref="#E.2.0">&#x2062;</mo>
      <!-- INVISIBLE TIMES -->
      </mo>
      <mi xref="#E.2.2">c</mi>
    </mrow>
    <mo xref="#E.0">+</mo>
    <mi xref="#E.3">d</mi>
  </mrow>
</semantics>

<annotation-xml
  encoding="application/openmath+xml">
  <OMA id="E">
    <OMS cd="arith1" name="plus"
      id="E.0"/>
    <OMV name="a" id="E.1"/>
    <OMA id="E.2">
      <OMS cd="arith1" name="times"
        id="E.2.0"/>
      <OMA id="E.2.1">
        <OMS cd="arith1" name="power"
          id="E.2.1.0"/>
        <OMV name="b" id="E.2.1.1"/>
        <OMI id="E.2.1.2">2</OMI>
      </OMA>
      <OMV name="c" id="E.2.2"/>
    </OMA>
    <OMV name="d" id="E.3"/>
  </OMA>
</annotation-xml>
</semantics>

```

Figure 6.8: Parallel markup with presentation markup elements pointing to content markup elements. The light gray range is the user's selection, with the start and end node in bold face. We first look up their closest common ancestor that points to content markup, and then look up its corresponding content markup – here: *E.2*

ML natively allows for representing such display alternatives using the *maction* element [ABC+10, chapter 3.7.1]. *maction* is a container for at least one Presentation MathML subexpression, plus additional display or interactivity parameters, possibly given as additional child elements; the *@actiontype* attribute allows for distinguishing between different purposes of using *maction*. We will mostly use it as a container for several Presentation MathML expressions that can be displayed alternatively. The *@selection* integer attribute controls which child is displayed. Besides a few suggestions, MathML does not prescribe definitive values for *@actiontype*, so we have introduced custom ones for the interactive services that we offer (cf. chapter 7). Here, we demonstrate the use of *maction* for subterm abbreviations. While the renderer could leave the abbreviations as attributions in the content markup and leave it to the user interface software to retrieve them from there and put them into *mactions*, those *maction* elements, being valid Presentation MathML, can as well be directly generated by the renderer. Consider a physics document, where the author provides  $W_{\text{pot}}(R)$  (potential energy) as an instructive abbreviation of the complex term  $\frac{-e^2}{4\pi\epsilon_0 R/2}$ . We introduced the OpenMath symbol *folding#abbrev* that serves as an attribution key and provided a notation definition that matches content markup expressions attributed that way and renders them as *mactions*. For example,  $\alpha(\text{folding\#abbrev} \mapsto W_{\text{pot}}(R), \frac{-e^2}{4\pi\epsilon_0 R/2})$  is rendered as

**<maction actiontype="abbrev" selection="1">** $\frac{-e^2}{4\pi\epsilon_0 R/2}$  **$W_{\text{pot}}(R)$ </maction>.**

#### 6.4.2.4 Annotations that Give Access to Notation-based Services

One case of structural information that is not present in the original markup but required for interaction with the presentation markup, as mentioned in the previous section, deserves special attention. [Section 6.1.2.2](#) introduces the use case of fixing a notational error spotted in a published document. Mere parallel content markup of symbols is not sufficient once there are multiple alternative notation definitions per symbol and the one chosen for rendering a symbol in a mathematical object is subject to a dynamic presentation context<sup>33</sup>. This use case, as well as the case of interactive notation switching introduced in [section 7.5.2](#), requires additional annotation of the published document.

As long as there is only one notation definition per symbol, it is trivial to locate the notation definition that has been used for rendering a symbol: One would simply follow the links from the rendered symbol to the corresponding semantic symbol declaration, and further on to the notation definition, which, in the underlying knowledge base, might be linked using the *has-NotationDefinition* property of the OMDoc ontology. In the case of context-sensitive selection from multiple notation definitions, the rendering engine is the only component of the system that knows what rendering has been applied; therefore, it *SHOULD* guide the editor by providing an according annotation in the published document. We chose to add a link to the fragment URI of the respective *rendering* element of a pattern-based notation definition<sup>34</sup> as a non-standard *@ec* attribute<sup>35</sup> to the topmost element of the resulting Presentation MathML fragment [KLM+09].

#### 6.4.2.5 Specific Requirements for Annotated Presentation MathML

Summarizing the alternatives outlined above, we pose the following additional requirements on preserving semantic structures in mathematical objects in compliance with the MathML specification.

1. Alternative displays, among which the user can switch, *SHOULD* be realized by *maction* elements and an *@actiontype* attribute that indicates the intended type of interaction.
2. Subterms that are not yet enclosed in an *mrow* or one of the grouping operators listed in [ABC+10, table 3.1.3.2] *MUST* be grouped using the invisible *mrow* element.
3. For services that need access to the semantics of mathematical expressions, the latter *MUST* be provided as parallel content markup [ABC+10, chapter 5.4]. There *MUST* be cross-links from all atoms and from the subterm-grouping elements of [requirement 2](#) to the corresponding content elements.
  - a) All symbols in the content markup *MUST* be linked to a place where further information about them can be retrieved, if the respective CDs have been published as linked data. If the CD URIs conform to the *cdbase/cd#name* format, symbols in the content markup *MUST* be provided as OpenMath OMS symbol references, or as strict Content MathML

<sup>33</sup>See [section 3.2.2.6](#) for the terminology.

<sup>34</sup>Recall that we focus on notational errors caused by the rendering (cf. [section 6.1.2.2](#)).

<sup>35</sup>named for its conceptual relation to the extensional presentation context, as defined in [Mül10a]

*csymbols*<sup>36</sup>. Where the CD URIs have a different format and no redirect interface accepting *cdbase/cd#name* URIs has been set up, non-strict Content MathML with *csymbol/@definitionURL* SHOULD be used instead.

- b) Content elements MAY be annotated with additional attributions.
- 4. If services that directly operate on the presentation markup require efficient access to specific information, e.g. to customize the display of the document, such annotations MAY be added directly to presentation markup elements as attributes from a non-MathML namespace. It is RECOMMENDED that such annotations require considerably less additional space than parallel markup; otherwise parallel markup SHOULD be preferred.
- 5. When services require information about the presentation markup that cannot be inferred from the content markup at all – such as the notation that has been used for context-sensitive rendering –, this information MUST be provided in the rendered object. It MAY be provided as an annotation to the presentation markup according to [requirement 4](#) or to the content markup according to [requirement 3b](#).

#### 6.4.2.6 Preserving Higher-level Structures Using XHTML+RDFa

Logical/functional structures on the statement and theory levels, as well as document and rhetorical structures, metadata, and all other information embedded into documents can be handled uniformly w.r.t. semantic structure preservation. In sections 3.2 to 3.5, ontologies for all of these structures have been introduced, allowing them to be represented as RDF graphs. These RDF graphs can be embedded as RDFa into the XHTML output format, on which this section focuses.

RDFa is already being used much more widely than parallel MathML markup and enjoys some tool support when used in published documents (e.g. via the *rdfQuery* JavaScript library [Ten+]), and as XHTML+RDFa is easier to handle than parallel MathML markup in many respects. Therefore, a treatment as elaborate as given for MathML above is not needed. “Interlinking” RDFa with presentational XHTML markup is trivial in that the RDFa attributes are either simply embedded into the XHTML elements they annotate, or, alternatively, attached to invisible XHTML elements (e.g. *spans*) that use the same *@about* attribute as the XHTML elements to be annotated. Looking up all RDF triples in an XHTML+RDFa document by their subject URI is a standard operation for libraries such as *rdfQuery*; RDFa 1.1 will further facilitate implementing such libraries by specifying an API (cf. [section 2.3.3.5](#) and [SAR+11]). As RDFa can use arbitrary RDF vocabularies, customized lightweight annotations that do not exist in the original semantic markup can easily be added. Displaying or hiding information on demand is easily possible using XHTML, CSS, and JavaScript.

---

<sup>36</sup>Note that, in contrast to *OMS*, *csymbol* only supports the *@cd* and *@name* attributes but not *@cdbase*. Instead, there can be a *math/@cdgroup* attribute for the whole MathML object, which points to an OpenMath CD group file that maps CD names to the corresponding CD URIs [ABC+10, chapter 4.2.3]. This mechanism has to be used for any object using symbols from CD bases other than the default <http://www.openmath.org/cd>, unless the document format embedding the MathML object defines a different mechanism – which, so far, only OMDoc does. In a linked data setting, where objects possibly use symbols from many different CDs from distributed sources, that creates the challenge of where and how to provide such a CD group file.

### 6.4.2.7 Size Complexity of Annotations

This section concludes with an estimate of the size complexity of annotations of Presentation MathML objects. In practical client-server applications, these considerations may influence the choice of how much information to embed unconditionally and redundantly into presentation-oriented documents, and how much information to leave in the semantic markup, from which it could be retrieved on demand.

Compared to generating presentation markup only, adding content markup roughly doubles the size of the output, assuming that the XML element and attribute names for content markup on average have the same length as those for presentation markup. This can be seen by structural recursion over the constructors for content markup: Atoms, i.e. symbols, variables, and numbers, are represented by a single element both in content markup and in presentation markup.<sup>37</sup> Applications of operators – and binders similarly – also have the same size in content and presentation markup, as in both the operator and its arguments occur in either representation<sup>38</sup>, and in most cases there is a grouping element around them in either representation<sup>39</sup>. Cross-links increase the size enlargement factor to up to four, as an ID attribute is added to almost every content markup element, and a link to one such content markup element is added to almost every presentation markup element. *mactions*, when nested, can lead to an exponential blow-up to the base of the number of alternatives in the worst case. This could be avoided if Presentation MathML allowed for structure sharing between expressions – which only Content MathML supports so far [ABC+10, chapter 4.2.7].

The size complexity of RDFa annotations depends on how the original semantic markup is translated to RDF. In most of the cases discussed in section 3.7, (i) an XML element maps to an instance of one ontology class, (ii) which is linked to the resource represented by its parent element by an RDF property, i.e. an average XML element generates two RDF triples. Some elements generate additional triples for resources that they implicitly represent, or relationships in which they implicitly participate. The average XML attribute generates one RDF triple; multi-valued attributes generate, of course, multiple triples. Summarizing, this results in approximately one triple per element tag and per attribute value of the semantic markup. In the worst case, each such triple is represented by an XHTML element of its own, having three attributes (subject, predicate, and object), which blows up the required space by a factor of four.<sup>40</sup> In the best case, when the structure of the XHTML output permits it, triples can be grouped by common subject and predicate, and literal objects as well as link targets can be shared with the presentation markup; that makes the RDFa annotations require about as much space as than the original semantic markup.

<sup>37</sup>Content markup symbols tend to be more verbose than presentation markup symbols, though: In the worst case, they carry a complete *cdbase/cd#name* URI represented by one to three attributes, whereas a presentation markup symbol in the best case only consists of an XML element that contains a single character, such as +.

<sup>38</sup>This figure is correct for prefix, postfix, and binary infix operators. In the case of *n*-ary infix operators and mixfix operators, an occurrence of the operator is placed between every pair of successive arguments in presentation markup, possibly even before the first and after the last argument. Thus, the content markup representation only needs half the size of the presentation markup, plus/minus a small constant.

<sup>39</sup>This figure overestimates the size of presentation markup when subterms are not always grouped into *mrows*, as, in some cases, there is no other markup – such as brackets – around a subterm.

<sup>40</sup>The Krenxor RDFa output module employed in our implementation (cf. section 8.1.1.2) yields a factor of three, as it groups triples by subject but does not perform any other optimizations.

Note, however, that this goal is hard to attain when both the XHTML and its RDFa annotations are generated automatically.

#### 6.4.2.8 Practical Realization

This section outlines how to realize semantics-preserving transformations in practice. [Appendix C.1.2](#) provides technical details of our implementation within the JOMDoc library [[Jom](#)].

**Mathematical Objects:** An integrated system that wants to render mathematical objects has to make the notation definitions of all symbols occurring in these objects available to the renderer. In the interest of comprehensibility, a renderer should support several alternative strategies for collecting and selecting notation definitions to make sure that the output coincides with the intention of the author and the background and preferences of the reader. Possible collection sources include the document to be rendered, the theories imported by the theory to be rendered, explicit links from the document to notation definitions, and built-in defaults that render all symbols in prefix notation [[KMRo8](#); [Mül10a](#)]. Context-sensitive strategies for selecting among alternative notations for a symbol could be guided by explicit context annotations in the document provided by the author – in varying degrees of granularity from the whole document to subterms of mathematical objects – or by user-specific “cascading context files” inspired by CSS [[KMRo8](#); [Mül10a](#)]. An environment that targets developers might, in addition to a presentational rendering, also the content markup source; the XHTML+MathML presentations of the OpenMath CDs even display several alternative content-oriented syntaxes, as shown in [figure 9.1](#).

In addition to requiring a client-side service to look up the declaration of a symbol in two steps, i.e. by looking up its content markup annotation and then dereferencing the CD definition URI given there ([requirement 3a](#)), one can attach direct links to presentation markup symbols. This has the advantage that neither scripting support on the client nor a linked data setup on the CD server is required. The SWiM semantic wiki utilizes these links – then pointing to the wiki pages of the symbol declarations – in its support for the notation fixing workflow introduced in [section 6.1.2.2](#); [section 9.4.2](#) describes the complete workflow support, which involves further services.

**Non-Object Markup:** Above the object level, our implementation extends an existing OMDoc → XHTML transformation, which has not preserved semantics so far, by RDFa output. This extension employs a generic RDFa annotation generator (cf. [section 8.1.1.2](#)), which serializes those RDF data into RDFa that have previously been obtained via the OMDoc → RDF translation specified in [sections 3.7](#) and [5.2.3](#). Thus, the semantics-preserving transformation reuses RDF extracted from semantic markup that an integrated environment needs in any case, if it wants to offer RDF-based services such as linked data browsing or information retrieval. [Figure 6.9](#) demonstrates the integrated setup, with which we have addressed the scenario of serving information needs of learners and instructors introduced in [section 6.1.2.4](#). Here, RDF extracted from OMDoc is not only used for publishing semantically annotated documents (lower left corner), but also to enable information retrieval. This modular approach to semantic publishing provides a high amount of flexibility in that the translation of semantic source markup to RDF operates independently from generating the semantically annotated presentation markup output.

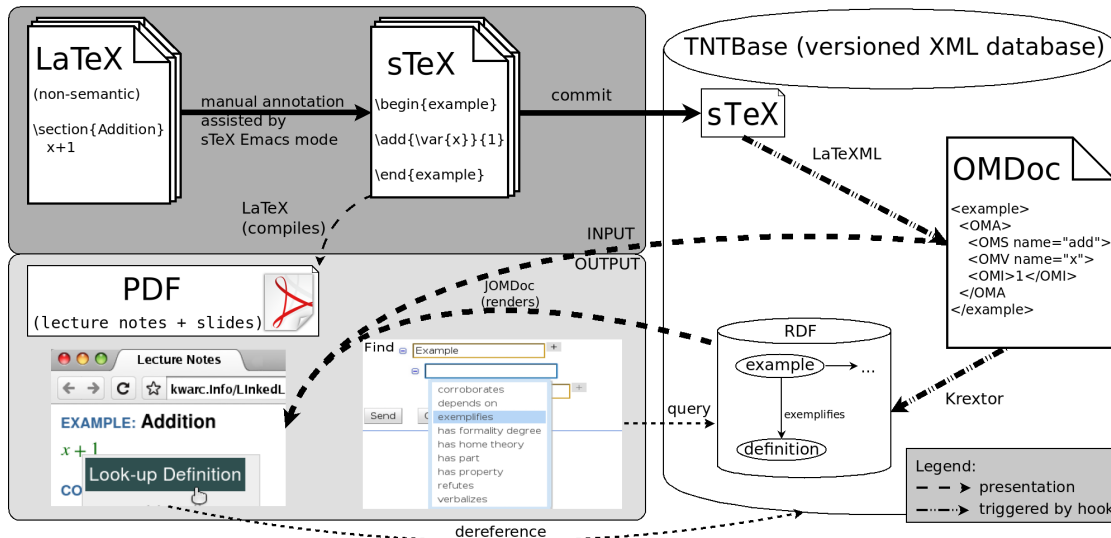


Figure 6.9: Publishing sTeX/OMDoc documents (here: MICHAEL KOHLHASE’s lecture notes) as linked data for humans and machines

Semantic representations of mathematical knowledge occur as highly modularized snippets of any granularity required by knowledge management workflows or service implementations, as discussed in section 8.3.2. However, if an author composed an aggregate document from such reusable parts, the reader may prefer reading it at once in a coherent view, without having to follow links to all included parts. Therefore, a renderer should support document inclusion, which is either built into semantic markup languages or can be added, e.g. using XInclude (cf. section 8.3.2.2 for concrete syntax examples).

**Rendering Notation Definitions** deserves special attention. In OMDoc 1.3, there is no “informal” counterpart to a pattern matching notation definition. Other than for the rest of OMDoc’s statements and contrary to the PlatΩ extension of T<sub>E</sub>X<sub>MACS</sub> [AFN+07], there is no way of inter-

**NOTATION DEFINITION:** for symbol *arith1#plus*

Prototypes	Rendering
<pre> &lt;om:OMA&gt;   &lt;om:OMS cd="arith1" mcd:cr="fun" name="plus"/&gt;   &lt;exprlist name="args"&gt;     &lt;expr name="arg"/&gt;   &lt;/exprlist&gt; &lt;/om:OMA&gt; </pre>	<pre> (arg1 + ... + argn) </pre>

Figure 6.10: Rendered notation definitions for the *arith1#plus* symbol of OpenMath, from the OpenMath wiki at <http://wiki.openmath.org/?title=ntn:arith1> [Lan].



mixing them with natural language, except for (ab)using the informal part of a symbol's definition for introducing an example of its notation, e.g. “we call  $\binom{n}{k}$  the binomial coefficient of  $n$  over  $k$ ”. Therefore, if an explicit documentation of a notation definition is desired, for example addressing authors interested in using the associated symbol, the “formal” one has to be published. In the course of supporting the “notation fixing” workflow from [section 6.1.2.2](#) in the SWiM semantic wiki (cf. [section 9.4.2](#)), notation definitions are rendered to display the source code of the content markup pattern and the preview of its rendering, with placeholder arguments, next to each other, as shown in [figure 6.10](#).

#### 6.4.2.9 Related Work and Future Directions

Within our current approach to publishing semantically annotated documents, certain improvements are possible. Also, there is related work covering different output formats. Both are briefly discussed here.

**Notation Definitions for Higher-level Structures:** The MMT library [[Raba](#)] implements rendering using declarative notation definitions and the XML object model built into the Scala programming language. A notable feature of the MMT library is that it also uses notation definitions to render statements, theories, and documents, and thus provides a uniform management of presentation on all levels of mathematical knowledge. The unification of MMT's declarative notation definitions with JOMDoc's pattern matching notation definitions scheduled for OMDoc 1.6 (cf. [section 2.4.5.4](#)) may allow for the best features of both implementations to be merged.

**Handling  $n$ -ary Associative Operators:** One shortcoming of the content lookup presented in [figure 6.8](#) is that it is not aware of  $n$ -ary associative operators. Suppose a user selects  $b + c$  from  $a + b + c$ . This selection does not have to be expanded to the full term: Due to the associativity of “+”, it makes sense on its own. The content lookup function would, however have to know about the associativity, and it would have to construct a return value that does not exactly exist in the content markup. Finding out whether an operator is associative could be supported by a custom annotation. In a linked data setup, the client could alternatively retrieve the CD and check the symbol's type declaration or notation definition. The content markup fragment to be returned cannot directly be taken from the annotation, as only the full term  $@(\text{arith1\#plus}, a, b, c)$  is available there. Instead, the function would have to look up the content markup corresponding to each top-level element of the selection, i.e.  $b$ ,  $+$   $\mapsto$   $\text{arith1\#plus}$ , and  $c$ , and then construct a new mathematical object  $@(\text{arith1\#plus}, b, c)$ .

**Annotating Presentation MathML with RDFa:** Another possible improvement would be using RDFa for our custom annotations of Presentation MathML to enhance their scalability and machine-comprehensibility. MathML does not officially support the introduction of new unprefixed attributes – which the RDFa attributes are. HTML 5, which supports MathML objects, has not addressed this problem either. Neither the MathML nor the OpenMath developers are currently planning to support RDFa.



**Annotating PDF Output:** In our work, we have not covered output to formats other than XHTML+MathML+RDFa. PDF is a desirable output format due to its high presentation quality. PDF supports arbitrary metadata on document level via XMP (Extensible Metadata Platform [Ado]), which is embedded RDF/XML. XMP officially restricts RDF to triples describing the complete document and blank nodes (cf. [Erio7] for a more detailed critique). That is, it neither defines URIs for fragments of a document, nor does it allow to use custom URIs as subjects of RDF triples. SALT (cf. section 3.3.1) circumvents this restriction by identifying annotated text spans by start and end positions instead of URIs. An alternative approach, taken by the PDFTab Protégé extension, hooks into the annotation API of the Adobe Acrobat PDF editor and embeds OWL as annotation objects in a similar way as text highlightings and post-it notes [Erio7]. Such annotations are, however, not guaranteed to work with PDF/A, the PDF subset for long-term archiving (cf. [Rus09]). An alternative that is compatible with archiving is *tagged PDF*, which contains structural annotations primarily targeting assistive technologies such as screen readers and non-standard rendering engines, e.g. on small mobile devices. Work towards explicitly supporting MathML for tagging PDF is in progress [Moo09].

### 6.4.3 Conclusion

This section has presented two complementary distribution channels for deploying mathematical knowledge on the Semantic Web: publishing “pure” linked data, so that services can easily access them and make them browsable, and embedding them as annotations into human-comprehensible documents, where assistive services such as those that covered in chapter 7 can utilize them right in the reader’s context. With mathematical computations on statistical datasets, I have pointed out a use case for mathematical knowledge on the Web of Data, while also discussing the remaining barriers to publishing mathematical linked data and suggesting how to overcome them. Embedding semantic structures as annotations into human-comprehensible documents is easy when the knowledge is represented in RDF, thanks to RDFa. Due to their higher complexity, less available tool support, and certain restrictions of MathML, annotating the semantic structure of mathematical objects requires special treatment. We have established guidelines on how to embed information about the semantic structure and interaction possibilities for client-side services into Presentation MathML objects in a standards-compliant way that degrades gracefully when such services are not available. With the JOMDoc renderer, we have developed a tool that publishes mathematical documents as human-comprehensible XHTML+MathML while preserving their full semantic structure as mathematical content markup and RDFa. JOMDoc’s rendering and annotation facilities are sufficiently generic to be applicable to any semantic markup language that contains content markup objects or has a translation to RDF.

In the context of the OpenMath CD maintenance scenarios introduced in sections 6.1.2.1 and 6.1.2.2, our implementations of linked data style navigation and rendering mathematical objects and notation definitions have been evaluated; chapter 10 reports on this.

## 6.5 Information Retrieval

Retrieving information and answering questions is a core feature of a system that manages a large knowledge collection. For mathematical knowledge, particularly for objects, this is, however,

harder to realize than for text, where keyword search is often sufficient. “Keywords describing mathematical objects are typically overgeneral” [Min05] in that they fail to narrow down the exact type of mathematical concept covered, e.g., by a publication,<sup>41</sup> and that users searching for the solution of a mathematical problem often do not know appropriate keywords to express their request. The latter is, for example, the case “when different fields of study have different terminology for identical mathematical objects” [Min05].

Research on MKM has so far mainly addressed these problems by a number of approaches to formula search, which section 6.5.1 briefly summarizes. Higher-level logical/functional structures of mathematical knowledge, such as statements or theories, have been covered much less by previous research, and retrieval tasks combining the logical/functional dimension with others hardly ever. Section 6.5.2 compares the potentials of XML and RDF queries w.r.t. such multi-dimensional structures and argue why RDF queries are more adequate to their semantics.

### 6.5.1 Searching MathML/OpenMath Objects (State of the Art)

The problem of efficient text search has long been solved. Nowadays, ready-to-use libraries for full-text indexing and search are available, such as Lucene [Apac]. While text search mainly focuses on determining occurrences of keywords or phrases, formula search requires a better understanding of structures (see, e.g., [You05]). Due to the good availability of high-performance full-text search libraries, formula search has first been implemented by reduction to text search. The Mathdex search engine translates Presentation MathML to text [MM07], whereas the ActiveMath e-learning system implements a similar translation from OpenMath to text [LM06]. These systems translate the functional structures of mathematical objects into text, which is then indexed by a text search engine. The resulting index preserves part of the original structural information, but other information is either completely lost or can at least not easily be retrieved any longer. Consider information about bound variables and their scopes and the example of the Pythagorean theorem introduced in section 1.4: A user searching for  $a^2 + b^2 = c^2$  would most likely also expect  $x^2 + y^2 = z^2$  to be listed as a search result.<sup>42</sup> MathWebSearch is an example of a search engine that indexes such term structures, given in Content MathML or OpenMath [KŞ06]. By its tree matching capabilities, it addresses not only the above-mentioned problem of a user not knowing appropriate keywords for what he is searching, but also the problem that the full *mathematical* structure might not be known. For example, MathWebSearch can be used to retrieve equations that contain an integral of a power of a function ( $\int_? s^2(t) dt$ ) [KŞ06]. It has also been combined with full-text search to make it more generally applicable [KAJ+08; Anc09].

<sup>41</sup>“One can search for ‘quadratic polynomial’, but there is no effective way to narrow the search to a particular polynomial or class of polynomials. This is particularly limiting for educational resources, where the same generic label applies to many different treatments of the same material. Searching for ‘rate of work’ with Google produces some 20,000 references. Finding out which of these documents might shed light on the particular rate-of-work problem in your child’s homework assignment is a laborious, and likely fruitless, task.” [Min05]

<sup>42</sup>The formula would more cleanly be written as  $\forall a, b, c. a^2 + b^2 = c^2$ , which makes the variable binding explicit and thus the possibility to rename the bound variables.

### 6.5.2 Querying Structures above the Object Level

On the levels of mathematical knowledge that are above objects and text paragraphs, wide-spanning relations become more important and demand different query answering approaches. Typical queries against logical and functional structures of mathematical knowledge, such as statements or theories, could be: find all symbols (directly or indirectly) defined in terms of a given symbol, find all theories recursively imported by a given one, or, on a higher level of abstraction, find all dependencies of a given knowledge item, or find structurally equivalent theory graphs (addressing the above-mentioned problem of different terminology in different fields).

The XML- and RDF-based representation of mathematical knowledge introduced in chapters 3 and 5 suggests applying state-of-the-art XML and RDF technology to mathematical problems. After a brief review of previous approaches to querying mathematical knowledge represented in XML and RDF (section 6.5.2.1), section 6.5.2.2 critically compares the capabilities of the XQuery XML query language and the SPARQL RDF query language w.r.t. querying mathematical knowledge, and section 6.5.2.3 points out the particular aptitude of RDF-based queries for multi-dimensional knowledge.

#### 6.5.2.1 XML and RDF Databases for Mathematical Knowledge

Even before general-purpose XML databases with support for XQuery (cf. section 2.3.2.3) and RDF triple stores with support for SPARQL (cf. section 2.3.3.5) had become available on a grand scale, MKM researchers developed their own solutions. MBase is a relational database that stores OMDoc documents (conforming to an older version of the language) [FKo6]. There is roughly one table per XML element in the schema, with one row per occurrence of that element in a document. The developers of HELM explored RDF queries. At that time, databases for storing RDF were available, but no suitable query languages, which is why they developed their own one (see section 1.4.3.1 for a brief review of HELM, and section 2.4.10.2 for the kind of RDF queried there).

TNTBase [Tnt; ZK09; ZK10], integrating the Subversion revision management system [Apar] and the Oracle Berkeley DB XML database [Ber] into a versioned XML database extensible by language-specific plugins, is probably the first serious application of a general-purpose XML database to MKM, as it has been extended with OMDoc-specific plugins [KRZ10]. The design of TNTBase has partly been influenced by the research this thesis reports on. The XQuery support of Oracle Berkeley DB XML and thus TNTBase, which comprises indices to speed up frequently occurring queries, and the XQuery Update facility [CDF+09], but not the XQuery/XPath Full Text extensions [AYBB+10], has proven suitable for querying non-heterogeneous semantic markup. Besides publishing MICHAEL KOHLHASE's OMDoc/TEX lecture notes as linked data for humans and machines using TNTBase and suitable translation and presentation plugins (cf. sections 6.1.2.4, 6.4.1.3 and 6.4.2.8), we have used TNTBase for refactoring OWL ontologies serialized in OWL 2 XML (cf. [LZ10]). The MMT system mentioned in section 2.4.4.1 also relies on a TNTBase backend for part of its computation of dependency relations (see below).

### 6.5.2.2 Querying XML vs. Querying RDF – Concrete Syntax vs. Abstract Semantics

Knowledge management does not necessarily require access to the full XML representation. In the OMDoc-based Logic Atlas, querying a structural outline of a theory graph has, for example, been found sufficient for most theory-level management tasks, for example computing all dependencies of a knowledge item [KRZ10]. I argue that, at least in terms of lines of code required for queries, SPARQL queries against an RDF representation can handle such abstractions from an original XML syntax better than XQueries against an XML representation. Starting with two concrete examples, I compare the general shortcomings of XML-based approaches to advantages of RDF-based approaches, and finally review the software support for RDF-based entailment.

**Examples:** Consider the two OMDoc fragments

```
<theory xml:id="t">
  <imports xml:id="i" from="#u"/>
```

and

```
<proof xml:id="p">
  <derive xml:id="step">
    <FMP> $\neg T = \perp$ </FMP>
    <method><!-- proof by known axiom -->
    <premise xref="#axiom1"/>
```

It would require a considerable effort of declaring, e.g., XML Schema datatypes and implementing XQuery functions to determine that both *#t* depends on *#u* and *#p* on *#axiom1*, whereas an OMDoc  $\rightarrow$  RDF translation would generate the triples ...

```
<#t> oo:hasImport <#i> .
<#i> oo:importsFrom <#u> .
<#p> oo:hasStep <#step> .
<#step> oo:stepExternallyJustifiedBy <#axiom1> .
```

... from which a query engine with description logic entailment support would infer ...

```
<#t> oo:dependsOn <#u> .
<#p> oo:dependsOn <#axiom1> .
```

The MMT library mentioned in [section 2.4.4.1](#) has built-in support for computing such dependencies over all documents that have been parsed into memory. Where this is not desirable, for example when managing large theory graphs, one may use the MMT plugin for TNTBase [Rabb; KRZ10]. In that setting, MMT detects and generalizes one-step dependencies with regard to well-formedness (cf. [section 3.2.1.3](#)) while parsing and validating OMDoc documents ([section 6.3.2](#)). Such direct dependency links – for example “*s* has occurrence of *t* in type” are stored in an RDF-like XML serialization<sup>43</sup> in TNTBase. The MMT plugin for TNTBase consists of a set of XQuery functions that compute, e.g., transitive closures of theory imports represented in that graph. Introducing another transitive dependency relation would, in the worst case, require both extending the MMT library and the XQuery module.

<sup>43</sup>a non-standard serialization that is structurally similar to a subset of RXR

Figure 6.11: Querying an RDF graph with SPARQL, using the input form of the Mocassin mathematical semantic search engine [Zhi] – here preloaded with the OMDoc ontology and running on top of the linked dataset obtained from MICHAEL KOHLHASE’s lecture notes and served by Virtuoso (cf. section 6.4.1.3)

In an RDF-based setting, one would simply introduce a new property  $r$ , make it a subproperty of, e.g., *oo:wellFormedNessDependsOn* (cf. section 3.2.2.4), and declare it transitive, relying on a suitable DL reasoner to do the rest of the job.

**General Shortcomings of XML:** XML Schema datatypes only offer a limited degree of abstraction from the concrete XML syntax – too limited in the context of OMDoc. The required abstractions could be realized by XQuery functions, however, at the expense of seriously complicating the code of queries.

Datatypes can abstract from differences in XML element names via a supertype/subtype hierarchy; consider a datatype *statement* that comprises the OMDoc elements *definition*, *axiom*, *assertion*, etc., as well as their informal *omtext[@type=...]* counterparts. Datatypes can also partly abstract from the choice of whether some information is encoded as an attribute or as a child element. But the mechanisms of XSD cannot abstract from the choice of whether a relation between two resources is represented in a parent-child way or by an element or attribute whose text content is the ID or URI of an external element. Finally, XSD can hardly capture the semantics of RDFa annotations, which we need for representing knowledge that exceeds the native vocabulary of OMDoc. The RDFa attributes often hold URIs of properties and classes from arbitrary ontologies, which can be encoded in various semantically equivalent but syntactically different ways.

Realizing such abstractions by XQuery functions complicates queries in that they would rather have to call these functions than perform simple node tests. The same holds for the above-

mentioned case of transitive closures, which, per transitive relation, requires implementing one XQuery function that computes its transitive closure.<sup>44</sup>

**General Advantages of RDF:** An alternative is querying, with SPARQL, RDF outline extracted from the XML representation. That requires an XML→RDF translation in the first place; however, if that is sufficiently expressive, it can already be used to even out those above-mentioned syntactic (but not necessarily semantic) differences in the XML markup that, e.g., XML Schema datatypes could not handle. In the resulting RDF outline, relations can be queried more flexibly, and higher levels of abstraction can be introduced via an ontology. SPARQL query engines can transparently traverse relations, i.e. RDF properties, in forward and backward direction and compute multi-step joins. In contrast, the XQuery syntax for querying an XML relation in forward direction considerably differs from querying it in backward direction. This is most apparent for ID/URI-style references, which frequently occur in markup languages for mathematical knowledge. Concerning performance, RDF databases usually natively support links and optimize querying them in both directions. XML databases, such as eXist [Exi] and Oracle Berkeley DB XML (see above), can be set up to index attributes, child elements, and sometimes links<sup>45</sup>, for a better query performance.

**Software Support for RDF Entailment:** SPARQL query engines can not only cope with information that is explicitly represented in an RDF graph, but also – optionally, and at the expense of performance – with additional RDF triples *entailed* by the axioms of an ontology. Formally this process is governed by the *entailment regime* that is in effect when processing a SPARQL query, e.g. the RDFS or one of the OWL 2 entailment regimes [GO10].<sup>46</sup> Practically, the entailment regime is determined by the kind of reasoning engine attached to an RDF triple store. Some triple stores, such as OpenLink Virtuoso [Olv], implement certain pragmatic inference rules that yield additional result triples even in the absence of a dedicated reasoning engine. Consider, for example, transitive closures of properties, which are important in mathematical knowledge bases (see above): Virtuoso recognizes properties declared as transitive using *owl:TransitiveProperty*, and a few other OWL features, but does not implement complete support for any profile of OWL.<sup>47</sup>

Furthermore, some SPARQL query engines, including Virtuoso [Olv] and ARQ [Arq], support a full-text extension to SPARQL, which is analogous to the above-mentioned XQuery/XPath extension. Figure 6.11 demonstrates an input form that generates a SPARQL query against a Virtuoso triple store, using the full-text extension as well as Virtuoso’s built-in additional inference rules.

---

<sup>44</sup>With the support for higher-order functions coming up in XQuery 3.0 [RCD+10], it will, however, be possible to implement a generic function that computes the transitive closure of a relation computed by a given function.

<sup>45</sup>While eXist and Oracle Berkeley DB XML do not offer link indices, they are considered a standard XML database feature [SVM+03].

<sup>46</sup>As XQuery is Turing-complete [Kep04], XQuery functions can also realize any desirable entailment. However, as explained above, *using* the entailed information is not as straightforward as using the explicit information, whereas in SPARQL it is. Secondly, the decidability and complexity of entailment under the SPARQL entailment regimes is known, whereas well-behaved subsets of XQuery have not been investigated.

<sup>47</sup>Virtuoso also even allows for querying transitive closures of properties not declared as transitive. This is a special case of the *property paths* support coming up in SPARQL 1.1 [HS10a].



### 6.5.2.3 Querying Multiple Structural Dimensions

The multiple structural dimensions of mathematical knowledge analyzed in [section 2.1](#) often co-occur in practical applications. We have chosen RDF as an appropriate representation for multi-dimensional knowledge due to its uniform structure and its extensibility by arbitrary vocabularies (cf. [section 2.5](#)) and introduced RDF vocabularies, i.e. ontologies, for all relevant structural dimensions of mathematical knowledge in [chapter 3](#).

SPARQL provides a natural way of querying such multi-dimensional structures. [Appendix C.1.3](#) explains some of the queries that we have implemented. In the SWiM semantic wiki, queries for ongoing discussions help to support the “Peer Review and Preparing Major Revisions by Discussion” workflow introduced in [section 6.1.2.3](#). ([Section 9.4.3](#) explains the complete workflow support, which involves further services.) These queries generate listings of recent ongoing discussions about unresolved issues by combining the logical/functional dimension of OpenMath CDs and their mathematical knowledge items, the orthogonal SIOC Core structure of the discussion forums associated to these knowledge items, the additional argumentative structure of these discussion threads, and general-purpose administrative metadata of discussion posts. Such queries would not have been possible on the level of XML, as, in the OpenMath wiki, only the CDs and the contents of discussion posts are represented in XML, whereas the structures of discussion forums and threads are directly represented in the RDF triple store.<sup>48</sup>

Another query, which has been implemented in the software project management scenario introduced in [section 6.1.2.5](#), allows a project manager to find a substitute for the employee Alice. The assumption is that, if Bob has recently worked on a document that depends on one of Alice’s documents, he is capable of substituting Alice. Dependency is determined w.r.t. the application-specific software process ontology and along logical/functional structures of the mathematical model of the software. Further dimensions involved into this query include document management/versioning, FOAF user profiles, and general-purpose administrative metadata. It is easy to imagine how additional dimensions could be employed for increasing precision or recall of the query, or for ranking results. Consider, for example, another filter that only accepts as substitutes employees who have never got a document rejected in any previous certification. Again, this query would not easily have been possible on the level of XML, as the level of abstraction required for finding related objects is rather high, and as the application-specific metadata are represented in RDFa in the original documents in the given scenario.

### 6.5.3 Related Work

This section has focused on tools for retrieving information from XML and RDF representations of mathematical knowledge. Whelp, a search engine for logical/functional structures in HELM, is neither based on XML nor on RDF (cf. [section 1.4.3.1](#)). Similar tools have been developed for the Mizar language (cf. [section 1.4.1](#)): MoMM is a unification/subsumption formula search engine similar to MathWebSearch [Urbo6]. MML Query solves similar problems as the query engines that have been developed for HELM in that it supports querying logical/functional structures – but

<sup>48</sup>With a suitable XML representation of discussion forums and threads, this query could have been answered on the level of XML. However, while SIOC is supported by many blog and forum engines [FGlo9], there is no comparable XML language.



not the full structures of mathematical objects – and administrative metadata [Bano6]. IMMANUEL NORMANN has addressed the initially mentioned problem of similar structures with different terminologies in his work on identifying intersections between theories, particularly aiming at making results (e.g. theorems) obtained in one theory reusable in the other one [Noro8]. He has solved this retrieval problem on a first-order simplification of the MML using a normalization and matching algorithm whose complexity far exceeds that of SPARQL queries.

### 6.5.4 Conclusion

Structural information retrieval is essential for managing mathematical knowledge, as keyword search often fails. This section has reviewed databases and query engines that enable information retrieval on XML and RDF representations of mathematical knowledge. Solutions focusing on specific representation languages have existed before, whereas the investigation of the potential of generic XML-based information retrieval approaches for mathematical knowledge has only started recently and RDF-based approaches have not been analyzed seriously since the early experiments in the HELM project. I have compared the present states of XML and RDF information retrieval technology w.r.t. their applicability to mathematical knowledge. From comparing the effort of implementing queries at the desired level of abstraction, I conclude that queries over RDF outlines extracted from the original XML markup offer a more adequate relation of abstraction level to lines of code to be written and handle structural multi-dimensionality more naturally than XML-based queries against the original markup.

The choice to translate XML to RDF was not primarily influenced by a desire for a higher querying performance, but by the desire to integrate *other* RDF-based services and datasets with mathematical knowledge represented in XML, which requires such a translation in any case. The *performance* of indexing XML plus running an XQuery vs. extracting RDF from XML and running a SPARQL query crucially depends on the efficiency of the XML→RDF implementation; a comparative evaluation is subject to future work.

## 6.6 Arguing about Problems and their Solutions

*A significant barrier to entry was the linear narrative style of the blog.  
This made it difficult for late entrants to identify problems to which their talents could be applied.  
There was also a natural fear that they might have missed an earlier discussion  
and that any contribution they made would be redundant.  
In open-source software development, this difficulty is addressed in part  
by using issue-tracking software to organize development around “issues”  
– typically, bug reports or feature requests –  
giving late entrants a natural starting point,  
limiting the background material that must be mastered,  
and breaking the discussion down into modules.  
Similar ideas may be useful in future Polymath Projects.*

—TIMOTHY GOWERS and MICHAEL NIELSEN [GN09]

The above quotation from the initiators of Polymath (cf. [section 1.4.2](#)) emphasizes the need for user interfaces that support structured discussions in MKM projects. A model for representing

such discussions has already been introduced in [section 2.1.8](#) and formalized and implemented in [section 3.6](#): a generic argumentation ontology with a mathematics-specific extension, focusing on problems with the formalization, comprehensibility, reusability, or applicability of mathematical knowledge items.

This section briefly reviews the state of the art of issue tracking systems, which inspired our approach to argumentation. Discussions in wikis were another source of inspiration; [section 9.1](#) reviews this and the general state of the art of wikis in coherence. Then, I give recommendations on designing user interfaces that guide the discourse from problems to solutions, and on developing assistants that support collaborators in implementing solutions in typical situations. [Section 9.3.3.2](#) presents an actual implementation in the context of the SWiM semantic wiki, where it heavily interacts with other components of that system.

### 6.6.1 Issue Tracking Systems (State of the Art)

In bug tracking systems, users or developers of a software system report issues with that system. Inexperienced users, or developers being in the design phase, often report issues with the system in general (e.g. that a certain feature is missing), whereas developers currently working on the implementation usually narrow issues down to a particular component of the system. Follow-up comments that elaborate on the description of an issue or that propose a solution can be given. Some systems, such as Bugzilla [[Bug](#)], support voting on the importance of bugs. In the end, a developer takes a decision and changes the affected source code, i.e. fixes the bug. Links from bug reports to the affected software artifacts are shown in some bug trackers, which are closely integrated with source code repository management systems, such as Trac with Subversion [[Trab](#)]. Similar patterns (discussion of changes and voting or decisions on their acceptance) are present in source code review systems (see, e.g., [[MRo8](#)]).

Ontology-based approaches to bug and issue tracking have not yet reached a mature state.<sup>49</sup> Approaches known to date emphasize other aspects of software engineering than the structure of discussions about issues. The Dhruv system [[ASH+06](#)] aims at supporting bug resolution in open source software communities – by interlinking code artifacts, bug reports, discussion posts and community members, and then recommending related resources. The EvoOnt family of ontologies for software engineering (cf. [section 3.6.3](#)) comprises a model of issues but has not particularly been used to support issue tracking, but rather for code analysis in software projects. There, the representation of bugs was merely used to provide additional information about source code, with relations such as “which bugs have been filed with this source file?”, or “which revision of a source file resolved a bug?”. Software support for the BAETLE ontology has not yet been implemented at all.

### 6.6.2 Recommendations for Using the SIOC Argumentation Module

Applications have some choice in how much of the SIOC argumentation module they want to support; the options are explained in [section 3.6.1](#). Therefore, we have phrased some recommendations on how to use it in social applications.

<sup>49</sup>For an application of ontologies to the particular case of *searching* bugs, see [[TLC+09](#)].

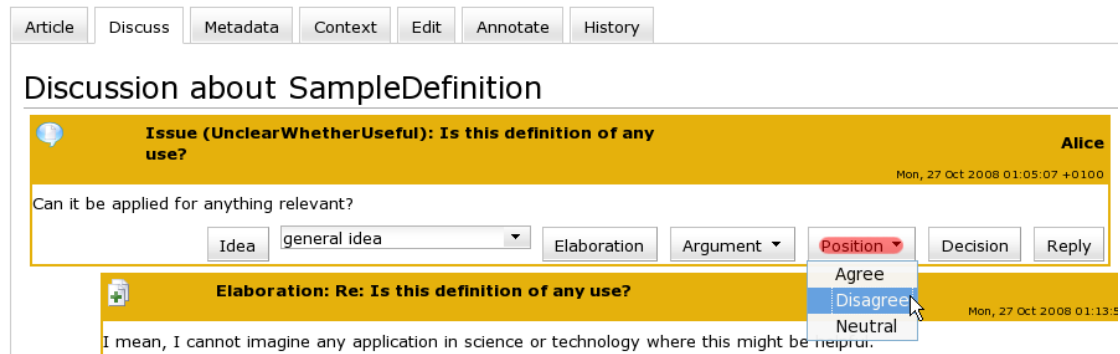


Figure 6.12: Replying to an issue (domain-specific, see below) with a position

It is RECOMMENDED that applications restrict the statement types with which the user can reply to a post to exactly those that are allowed by the schema, plus possible subclasses thereof, in order to keep the discussion focused – as suggested by previous research on argumentation ontologies, cited in [section 3.6.1](#). Where domain-specific types have been defined, their applicability – e.g. of an idea type to an issue type, or of an issue type to a knowledge item type – SHOULD also be taken into account (cf. [section 3.6.2](#)). It is RECOMMENDED to close an argumentative thread with a decision, with no more possibility to submit posts. The possibility to reopen a closed thread, as known from bug tracking systems, MAY be offered. In a small web of trust it may be feasible to let every user make decisions, whereas in larger social networks it is RECOMMENDED to restrict this to moderators.

The SIOC argumentation module does not explicitly capture voting. Voting MAY be emulated by using proper posts of type *sioac\_arg:Position*, but a developer MAY also introduce a mechanism of rating argumentative statements. There exist several possibilities to model voting: (i) The ChaO ontology of Collaborative Protégé (cf. [section 2.1.8.2](#)), for example, allows for either “5-star” or “yes/no” voting [TN07], whereas (ii) The DILIGENT-based Cicero semantic wiki plugin (cf. [section 9.5.7](#)) allows for “yes/no” voting either on individual ideas or in a multiple choice way [DGG+08]. When using voting in problem solving, the process MAY be made more efficient by separating it into two stages, as Cicero demonstrates: setting a deadline until which all argumentation (such as coming up with ideas and arguing on them) has to be finished, and then allowing the community to vote, as to prepare a final decision.

[Chapter 10](#) reports on an evaluation of the argumentation support offered by the SWiM semantic wiki (cf. [section 9.3.3.2](#)) in the context of the OpenMath CD review and discussion workflow introduced in [section 6.1.2.3](#).

### 6.6.3 Automated Problem-Solving Assistance

A simple automated problem solving assistance based on [part of] the argumentation ontology can be specified as follows: Whenever there is a discussion about a knowledge item, the system SHOULD check whether there is an issue that is both unresolved (meaning that no decision on it has been posted yet) and not challenged as invalid by the existence of a majority of disagreement

replies to it. If ideas have been posted on how to resolve this issue, the most popular one in terms of the ratio of agreements to disagreements SHOULD be selected. Formally, any issue  $s$  satisfying

$$D(s) = \emptyset \wedge (P^-(s) \neq \emptyset \Rightarrow \#P^+(s) > \#P^-(s))$$

is considered legitimate, and the idea  $i = \arg\max_{i \in Id(s), P^+(i) \neq \emptyset} \frac{\#P^+(i)}{(\#P^-(i))+1}$  wins, where  $Id$ ,  $D$ ,  $P^+$ , and  $P^-$  denote sets of ideas, decisions, agreements and disagreements with an issue or idea, respectively<sup>50</sup>.

The system MAY provide assistance to any volunteering author to implement the best proposed solution, or one of those that rank highest, in the space of knowledge items, e.g. by automatically creating a template for a new knowledge item that the author can then complete. If an author follows the steps proposed by the system, the system SHOULD conclude the respective discourse by posting an automatically generated decision. Still, freedom SHOULD be left to the community to implement solutions manually, when users feel that the automatic support is not adequate to the wickedness of the current problem. In this case, the author to resolve an issue has to document this decision manually. Any thread that has been concluded by a decision will no longer be considered by the system.

Note that those issues for which automatic support can be offered do not satisfy the definition of a wicked problem. When a user is able to choose a specific type for an issue, that contradicts the traits of a wicked problem to have “*no definitive formulation*” and to be “*essentially unique*” [RW73]. For every type of issue, my model supports a set of predefined solution patterns, but a wicked problem does “*not have an enumerable (or an exhaustively describable) set of potential solutions*” [RW73]. Moreover, the current approach to assistance focuses on one issue with one knowledge item and neglects the previous history of related issues and knowledge items; however, “*every wicked problem can be considered to be a symptom of another problem*” [RW73]. Also, assuming that an issue has definitely been solved once a “decision” reply exists – contradicting the definition that “*wicked problems have no stopping rule*” [RW73]. However, the possibility of reopening a discussion has been mentioned above, and, in any case, a user who considers an existing solution inadequate is always free to file a new issue on the affected knowledge item.

#### 6.6.4 Related Work

MathOverflow [Mate], initially mentioned in section 1.4.2, does not track issues with a concrete knowledge base but is an agile forum for collaboratively solving mathematics problems. It is inspired by Stack Overflow [Sta], which does the same for programming problems, and runs the same software. The community can rate whether questions are useful and clear and whether answers are helpful, and the author of a question can mark the best answer. The only domain-specific adaptations include editing and presentation – source code syntax highlighting and L<sup>A</sup>T<sub>E</sub>X formulæ, respectively – but no domain-specific flow of discussion.

#### 6.6.5 Conclusion and Future Work

Structured collaborative resolution of problems has been addressed before in the related fields of ontology engineering and software engineering, but not particularly for mathematical knowledge

<sup>50</sup> It is not yet clear what idea should be preferred when there is more than one such  $i$ .

engineering so far. Based on an argumentation ontology, we have now enabled a structured flow of *discussing* problems with knowledge items, where the participants are guided towards deliberating solution proposals. Once the community has agreed on implementing a particular solution, the collaboration environment can assist with its implementation, if the solution follows a common pattern for which the environment has built-in support. Up to this step, the workflow does not depend on the mathematics-specific extension of the argumentation ontology, but only on its generic core.

Three problems with the current state of argumentation support and problem solution assistance remain, which future work should address:

**Dependency on Knowledge Items and their Types:** Currently, all issues refer to knowledge items, and any assistance offered depends on the type of knowledge item. That is, the service expects that knowledge items already exist and have some structure. In two situations, this is likely not to be the case: (i) When knowledge about a *new* topic has not even been conceptualized, or (ii) when it has been conceptualized and put on a wiki page, but not yet formalized (here: annotated with a structural type). To improve on this, a global discussion space for conceptualization issues should be provided, as it was originally intended with DILIGENT (cf. [section 2.1.8.2](#)), and a generic issue type “needs formalization” should be introduced, which can be filed with any knowledge item that does not yet have a type from a structural ontology. Assistance with formalization would likely require automated annotation techniques, which are out of the scope of this thesis (cf. the discussion in [section 2.1.2](#)).

**Simplistic Selection of the Best Idea:** So far, the selection of the best proposed solution only takes a subset of the argumentation model into account. The most ambitious feature to be added for a full coverage is a valuation of *Arguments*. An *Argument* in itself is positive, negative, or neutral, as a *Position*, but users can again reply to arguments with positions. A weighted valuation of arguments has been presented in the context of the Zeno argumentation framework [GK97]. The developers of DILIGENT have paid special attention to detecting inconsistent argumentation – e.g. when one user first votes in favor of one argument *a* but then introduces a new one that contradicts *a* [TPS+05] – and fostering consensus, none of which I have investigated so far.

**No Support for Fine-Grained Argumentative Structures:** The problem of fine-grained argumentative structures within discussion posts, which may even make it impossible to classify the argumentative role of the whole post, has been mentioned in [section 3.6.3.2](#). Besides developing an appropriate conceptual model for this<sup>51</sup>, users have to be assisted in making fine-grained structures of their argumentation explicit – once more, possibly, by automated annotation techniques.

## 6.7 Conclusion

This chapter has provided a comprehensive overview of essential building blocks required for an environment for collaborating on mathematical knowledge. For supporting the process of

---

<sup>51</sup>... which will inevitably be incomplete in other regards [SM93]

creating, formalizing, and organizing mathematical knowledge that is comprehensible, reusable, and applicable, we need editing, validation, publishing, information retrieval, and argumentation services. For the representation formats advocated by this thesis – OMDoc, OpenMath, MathML, and RDF(a) – a number of primitive services already exist and have been reviewed here. Other components required for an integrated collaboration environment have been scarce so far; therefore, we had to develop them:

- a semantic document editor that can also handle formulæ and metadata
- a technique for validating metadata and links in mathematical semantic markup using RDF
- methods and tools for publishing mathematical knowledge, both in XML and RDF representations, as linked data – both as pure data for general-purpose services and as annotations for assistive services in human-comprehensible documents.
- recommendations on querying an RDF representation of mathematical knowledge with SPARQL
- a user interface for arguing about problems with mathematical knowledge items and their solution, as well as an initial method for automated problem-solving assistance based on the argumentative structure of a discussion thread

While each of these primitive services is useful in itself, the actual *workflows* in collaboration on mathematical knowledge can only be supported by combining multiple services, as discussed in sections 6.1.2 and 6.1.3. Consider the following two cases, which have been addressed by previous research:

- Knowledge formalized with an editor that does not perform validation is not necessarily comprehensible, reusable, and applicable by subsequent services. [Section 9.5.1](#) reviews systems that integrate editing and validation.
- Editors that retrieve information that is reusable in the current editing task are well known from integrated development environments for software; consider auto-completion of function names in scope. Similar services for MKM have been investigated, e.g., in the  $\text{\LaTeX}$ IDE mentioned in [section 6.2.1.1](#).

The remainder of this thesis covers further cases. Examples besides the use cases and workflows introduced in [section 6.1.2](#) include the following:

- Expressive representation languages, such as OMDoc, enable, in principle, a concurrent evolution of vocabularies (symbols and notation definitions, or metadata) and the documents in which they are used, and thus contribute to efficient knowledge organization. The SWiM semantic wiki presented in [chapter 9](#) combines editing interfaces for such different structures of knowledge.
- Being able to explore a knowledge collection along relations such as dependency, and viewing published versions of each knowledge item at the same time, helps to make the collection more comprehensible. This combination of information retrieval and publishing is another typical feature of semantic wikis.



Table 6.2: Estimated potential of service integration

Relevance of ▼ for ►	Editing	Validating	Publishing	Inf. Retr.	Arguing
Editing	++ <sup>a</sup>	n/a	+	+ <sup>b</sup>	○ <sup>c</sup>
Validating	++	++	+	++	○
Publishing	++ <sup>d</sup>	○ <sup>e</sup>	++	++	+ <sup>f</sup>
Information Retrieval	+	○	○	++	○
Arguing	+	+	○	n/a	++

<sup>a</sup> The values on the diagonal refer to the integration of two different services of the same kind, e.g. a formula editor with a document editor, or a formula search engine with a theory search engine.

<sup>b</sup> Here, “editing” means annotation or formalization.

<sup>c</sup> Once fine-grained argumentative structures are concerned, editing becomes more relevant.

<sup>d</sup> Publication is relevant for editing insofar as it helps authors to verify whether they got their edits right (e.g. via a preview functionality), and as annotations in published documents can provide quick access to an editor.

<sup>e</sup> Publication is relevant for validation by human readers or external services.

<sup>f</sup> Published documents should enable their readers to give quick feedback.

Table 6.2 summarizes the estimated potential of integrating different services with each other.

Services that *apply* mathematical knowledge have not been covered here and are beyond the scope of this thesis. However, integrating useful applications into the environment where knowledge is created, formalized, and organized may provide an incentive to contributors, as argued in section 1.6.1, and further enhance the comprehensibility of that knowledge. Section 7.6 gives examples for the latter by describing the integration of external computation services into published documents.

A challenge for integrating different services is that they often operate on different representation formats. For example, type or proof checking an OMDoc document may require translation to a language for formalized mathematics that the respective checker natively understands. Or consider  $\text{\LaTeX}$  documents: One can directly publish them as static PDF by compiling them, but publishing them as interactive web documents requires translation to OMDoc (cf. figure 6.9). Overall, this chapter shows that document-oriented editing and publication for human audiences works better with an XML representation, whereas RDF is preferable for information retrieval except on the object level, and both representations are good to have for a thorough validation.

The following chapters address the integration of the services presented so far: Chapter 7 presents an approach for integrating services into documents, where they offer interactive assistance, utilizing the semantics-preserving transformations introduced in section 6.4.2. Chapter 8 addresses the challenge of different representations preferred by different services by transparently translating between these representations. Building on these results, chapter 9 finally presents a prototypical collaboration environment that integrates the primitive services needed for accomplishing the collaborative knowledge management workflows introduced in section 6.1.2. Chapter 10 evaluates the usability of that environment and its services, covering metadata and notation editing, partly also editing of formulæ and logical structures in documents, browsing linked data, comprehensibility of published documents, argumentative discussions, and, marginally, RDF-based information



retrieval. The evaluation particularly focuses on the support for the overall workflows as well as on general usability challenges in such heterogeneous environments.

## Acknowledgments

The educational scenario introduced in [section 6.1.2.4](#) has been studied in collaboration with MICHAEL KOHLHASE, FLORIAN RABE, NIKITA ZHILTSOV, and VYACHESLAV ZHOLUDEV [DKL+10b]. Project management in a software engineering context, as mentioned in [section 6.1.2.5](#), has been studied in collaboration with ANDREA KOHLHASE and MICHAEL KOHLHASE; the queries over the multiple dimensions of software engineering documents discussed in [section 6.5.2.3](#) are another result of that [KKL10a]. Possibilities for managing Flyspeck formalization project have been investigated in collaboration with SEAN McLAUGHLIN and FLORIAN RABE [LMR08]. GORDAN RISTOVSKI has added most of the mathematical element insertion and deletion facilities to the annotation plugin presented in [section 6.2.3](#). ALBERTO GONZÁLEZ PALOMO has integrated his Sentido formula editor into the TinyMCE HTML editor, as explained in [section 6.2.4](#), for the most part following a roadmap that I had established for integrating TinyMCE+Sentido into the SWiM semantic wiki for the purpose of maintaining OpenMath CDs (cf. chapters 9 and 10). The text of [section 6.2.4](#) has been adapted from a joint publication [LGP08]. A parser for the linear syntax for notation definitions presented in [section 6.2.5](#) has been implemented by MAJA GRINTAL and myself, and integrated into the JOMDoc library by DIMITAR MIŠEV. The idea of generating pragmatic XML markup from RDF-based metadata vocabularies discussed in [section 6.3.3.1](#) is due to MICHAEL KOHLHASE [LK09]. The initial idea of annotating the mathematical semantics of statistical datasets with OpenMath functions (cf. [section 6.4.1.1](#)) has been jointly developed with DENNY VRANDEČIĆ and elaborated in a joint publication with MICHAEL HAUSENBLAS, JIE BAO, and LI DING [VLH+10]. The idea of visualizing argumentative discussions in the Exhibit timeline widget, as shown in [section 6.4.1.2](#), is due to TUUKKA HASTRUP [LHC08]. The requirements for preserving semantic structures of mathematical objects when publishing (cf. sections 6.4.2.2, 6.4.2.3 and 6.4.2.5) have been developed jointly with FLORIAN RABE [GLR09]; the physics example in [section 6.4.2.3](#) is due to HEINRICH STAMERJOHANN. The way of pointing from rendered formulæ to the notation definitions used by the renderer (cf. [section 6.4.2.4](#)) has been developed in collaboration with CHRISTINE MÜLLER [KLM+09]. The JOMDoc renderer, which satisfies these requirements, has been implemented by NORMEN MÜLLER and DIMITAR MIŠEV. The XSLT stylesheets for rendering OMDoc 1.2 documents mentioned in [appendix C.1.2](#) have originally been developed by MICHAEL KOHLHASE, but then improved and extended by me. DAVID CARLISLE kindly supported me in adapting and integrating his XSLT stylesheets for rendering OpenMath CDs. The possibilities for SPARQL queries in Virtuoso (cf. [section 6.5.2.2](#)) have been investigated in collaboration with NIKITA ZHILTSOV and VYACHESLAV ZHOLUDEV [DKL+10b]; moreover, the “XQuery vs.SPARQL” comparison is the result of an in-depth discussion with the latter. The recommendations for using the SIOC argumentation module ([section 6.6.2](#)) have been established in collaboration with ULDIS BOJĀRS and TUDOR GROZA [LBG+08]; the review of issue tracking systems is partly based on the same publication.



## Integrating Assistive Services into Interactive Documents

I conceive reading a [mathematical] document as a process of understanding it by interacting with it. The JOBAD architecture<sup>1</sup> enables the integration of interactive services into documents – services that assist readers in adapting the document’s appearance to their preferences or in looking up additional information and displaying it right in place, i.e. without forcing them to switch their attention away from the document. Note that, in this chapter, the term *client service* refers to code that runs inside the user’s document viewer/browser. This may involve communication with *web services*, which usually reside on remote hosts.

In a client/server web environment, part of this adaptation and content selection can – and should! – already be performed on the server before delivering the document. Besides the context-sensitive notation selection mentioned in [section 6.4.2.8](#), analogous functionality for document-level structures is ready to use (cf. [Mül10a]) and could be fed with information from a user profile or user model (cf. [section 3.5.1](#)). JOBAD focuses on situations where sufficient information about the user is not available, as, e.g., in the case of a casual, possibly anonymous visitor, or where the adaptively generated document does not fully satisfy the needs that the user has just in this moment. Suppose an employee of a European engineering company reads and implements a specification: His general user profile would be set up to display SI units and concise formulæ without intermediate steps and explanations. But, as the company also has American customers, the European engineer might occasionally be interested in getting the same specification displayed in Imperial units, but then, being less familiar with these units, with more intermediate steps and explanations.

---

<sup>1</sup>named for its original implementation as a JavaScript API for OMDoc-based Interactive Documents. The original name referred to “active documents”. I do, however, believe that the term “active document” is too general for this thesis. For example, Microsoft has used it for an API for embedding document widgets into foreign applications, such as web browsers [Mic]. A search in scientific digital libraries reveals a multitude of interpretations of the term. In the discourse in the KWARC research group, it also refers to documents generated adaptively to user preferences, an aspect that this thesis does not contribute to. Therefore, I use the more restricted term “*interactive document*” here.

Section 6.4.2 has introduced semantics-preserving transformations for publishing human-readable documents that additionally represent the complete knowledge they contain as machine-comprehensible annotations. In the JOBAD architecture, these annotations serve as anchors for assistive services. For example, looking up the definition of a concept is only possible on a symbol in a mathematical object, or on a technical term in a text, either of which would be a concrete occurrence of that concept. We have designed client services that rely exclusively on annotations given in the rendered document – then, mostly for customizing its appearance, – and thus also work on local files, client services that retrieve additional information from the primary server backend that has generated the document, and client services that retrieve information from arbitrary external sources.

Section 7.1 reviews the state of the art of interactive mathematical documents, and then establishes requirements for a more flexible and more extensible architecture for integrating services into documents in section 7.2. The JOBAD architecture with its generic client- and server-side components is described in section 7.3. Sections 7.4 to 7.6 present a representative selection of client services that we have realized. Appendix C.2 describes technical details of our implementation.

## 7.1 State of the Art and Related Work

Interactive documents have been investigated in a number of research efforts, covering topics such as adapting documents, interactive exercises, and connecting to web services. The developers of the ActiveMath e-learning system have investigated how to *aggregate documents* from a knowledge base such that the resulting document contains exactly the topics that the reader wants to learn and their prerequisites [Act]. *Interactive Exercises* have been realized in ActiveMath and MathDox [Matb; GMo8; CCK+o8; CCJ+o6]. Here, the user enters the result into a form and then gets feedback from a solution checker in the server backend. ActiveMath comes with its own collection of web services [MGH+o6]. MathDox has originally been designed for talking to CAS – e.g. inputting an expression and getting it computed or rendered – but can also connect to other web services via MONET (see below). ALEX GERDES et al. have developed a reusable exercise feedback web service for mathematical exercises that has also been integrated with MathDox [GHJ+o8]. Besides supporting MathDox’s own communication protocol, GERDES’s web service also complies to the XML-RPC and JSON data exchange standards [GHJ+o8]. The web services developed within the MathDox and ActiveMath projects, such as the ActiveMath course generator, are potentially open to any client, but have not been used with any frontend other than their primary one so far [Ullo8; MGH+o6].

Rich architectures for *mathematical web services* have been designed for integration with multiple systems, such as the ones developed in the SCIENCE and MONET projects mentioned in section 1.4.3. SCIENCE targets symbolic computation and grid computing and does not consider documents as user interfaces. MONET is an architecture that, in principle, allows for any kind of mathematical web service. Still, mainly computational web services have been developed and evaluated within that framework. Web services can register with a central MONET broker that accepts requests, which do not directly call a web service but consist of a problem description (e.g., solve an equation, given as an OpenMath expression). The broker then forwards the request to the

best-matching web service. The above-mentioned MathDox allows for access to MONET web services via a document interface.

*Asynchronous communication* with a server backend (AJAX [Gar05]) allows for client/server interaction without submitting forms. It is a prerequisite for responsive in-browser applications: A client-side script can exchange small data packets with a server backend and insert responses from the server into the current page. This technique is employed by MathDox [CCK+08] and GERDES's frontend to their feedback web service [GHJ+08].

Despite the efforts mentioned above, there is still a lot of static mathematical content on the Web. Where documents act as frontends to web services, as in the above-mentioned systems, they have usually been designed to give access to a small selection of web services performing very specific tasks (mostly giving feedback to exercises and symbolic computation) – as is the case with ActiveMath and MathDox.

Beyond the foundational AJAX, several development kits have given drive to the construction of *mashups* – lightweight interactive Web 2.0 pages that combine, aggregate and transform data from different static sources or web services (cf. section 1.3.2). Combining and aggregating data is not addressed in this section,<sup>2</sup> but the user interfaces that give access to such data are. The former Mozilla Ubiquity [Moz] was a library for realizing interactive commands that the user could apply to things selected on the web page currently viewed in the browser, e.g. computing the direction to a selected geographical location. Jigs for OWL [VL; VLL10] offers a collection of widgets for searching and exploring knowledge collections represented as RDF graphs or OWL ontologies. Similarly, some of the browsing widgets mentioned in section 6.4.1.2 can be combined with other components of an interactive web page.

## 7.2 Requirements for Integrating Services into Documents

Our goal was the development of a mashup-like architecture that enables the integration of a wide range of services into mathematical documents, making them interactively accessible. The initial focus was on assistive services that support users in adapting the appearance of a document to their needs or that provide further information on demand, right in place.

The requirements for client services running inside the documents and for the backend generating interactive documents were as follows:

1. A backend that generates interactive documents **MUST** generate fine-grained semantically annotated presentation markup. Any information required by a client service **MUST** either be provided in the document, or there **MUST** be a reference that the client can use to retrieve the missing information.
2. Client services **SHOULD** be prepared to cache information that they have retrieved from a server locally to avoid unnecessary subsequent requests for the same information. In particular, client services that rewrite part of a document **SHOULD** retain the previous state of that part so that the user can switch back to it. Constructs built into the respective representations languages **MAY** be used; however, they **MUST** be distinguished – e.g. by

<sup>2</sup>The most prominent development kits in this field follow the Unix pipe paradigm. Yahoo! Pipes [Yah] support RSS, JSON, and several other common web 2.0 formats. Semantic Web Pipes transfer this approach to RDF [LPPH+09].

annotation attributes – from analogous constructs that occur in a document and serve the purposes of other client services.

3. Any markup that a client service adds to a document or changes in a document, e.g. when rewriting a mathematical expression, **MUST** satisfy [requirement 1](#).
4. It **SHOULD** be possible to invoke a client service via any kind of user interface; a client service **SHOULD NOT** make more assumptions than necessary about a particular kind of user interface.
5. The server backend that generates an interactive document **SHALL** serve the client-side scripting code for all client services supported in that document to the client. Initialization code for each client service **MUST** be embedded into any such document. To modules that access web services or remote data sources, a URL **MUST** be provided to which they can connect. Such a URL **MAY** contain placeholders that are filled in on each request from the user – e.g. a placeholder for the URI of the mathematical symbol that the user has selected. Information from web sources **SHOULD** be made accessible via a simple interface, preferably linked data or REST [[Fieoo](#)]. Web services and data sources **MAY** reside anywhere on the Web.
6. The range of accessible web services **SHOULD** not be limited to a particular protocol, such as XML-RPC or MONET; the architecture **SHOULD** be open to support any web service with an HTTP interface.

In the specific case of XHTML+MathML+RDFa documents, these requirements can be refined as follows:

1. Documents **MUST** be annotated according to the requirements from [section 6.4.2.5](#) in the case of mathematical objects, and with RDFa (cf. [section 6.4.2.6](#)) in the case of XHTML.
2. Inactive *maction* children **MAY** be used for caching in MathML, *divs* or *spans* in XHTML.
5. JavaScript **SHOULD** be used for client-side scripting. Initialization code **SHOULD** be provided as JavaScript embedded into the document *head*.

### 7.3 The JOBAD Architecture

The JOBAD architecture, shown in [figure 7.1](#), consists of a document format that enables interactive documents, client services operating on such documents, a user interface giving access to these client services, a lightweight communication protocol for connections from a document to web services, and a set of generic communication and document manipulation functions utilized by the user interface and client service components. While a JOBAD-enabled document may be authored manually by hard-coding annotations and the URLs of web services into the document, we rather assume that documents have automatically been rendered from a semantic representation, e.g. using a renderer as described in [section 6.4.2.8](#), and that the backend that serves the rendered documents also controls what web services they have access to by putting appropriate initialization

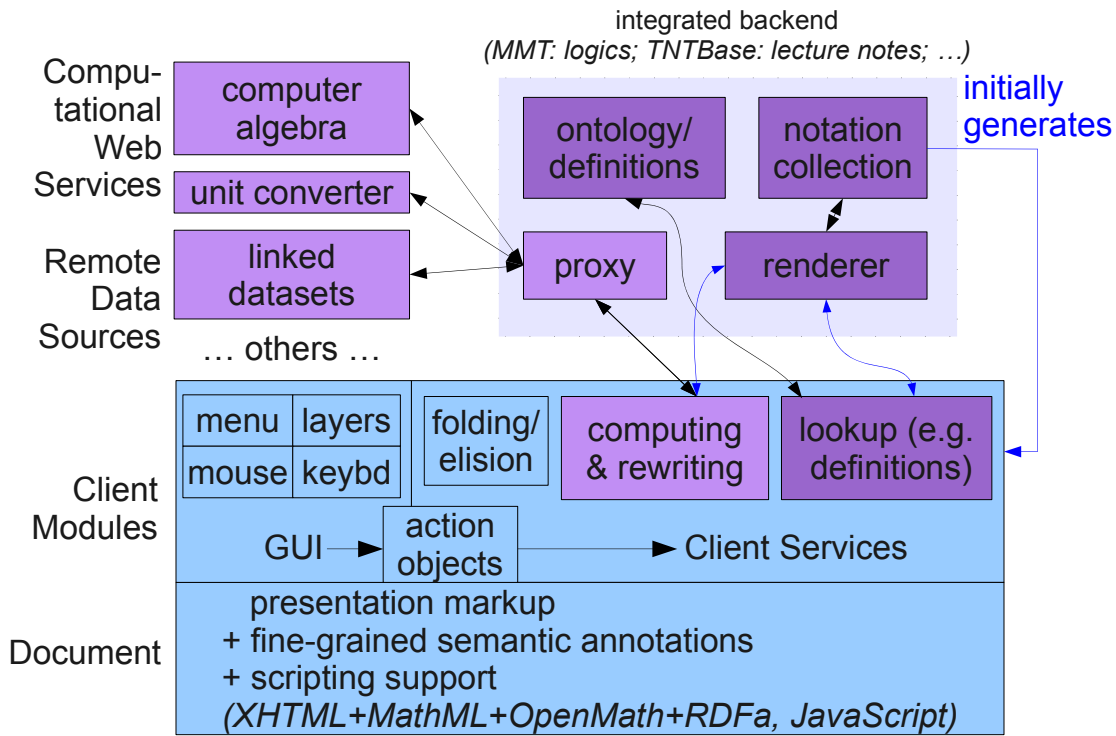


Figure 7.1: The JOBAD architecture (concrete implementation systems/languages in *italics*)

code for those service modules into the document. Besides rendering a document initially, the rendering web service is also employed by other client services in order to have markup rendered on the fly. [Section 7.6.1](#) details rendering as a web service.

### 7.3.1 Types of Client Services and their Initialization

Three kinds of client services can be distinguished by the complexity of data they exchange with a web service backend:

**In-Document Client Services** that exclusively draw on information that is already embedded into the document,

**Symbol-based Client Services** that retrieve remote information about a single mathematical symbol or RDF resource (either by dereferencing its URI in a linked data style, or by sending the URI and an identifier of the requested kind of information to a web service defined when initializing the respective client service), and

**Expression-based Client Services** that send complex content, e.g. the content markup corresponding to a whole mathematical expression selected by the user, to a web service.

The decision of what kind of client service to develop for realizing a particular functionality depends on the requirements of the particular application w.r.t. the size of the documents and the



amount and speed of client-server communication. For example, one can expect that definitions of symbols will only be looked up occasionally, so it would be a waste of resources to put all definitions of all symbols into hidden parts of a document when rendering it.

The execution of a client service is triggered by the user interacting with user interface elements (see below) in a document. Therefore, a server backend has to add initialization code for all client services that should be available in that document when generating it according to [requirement 5](#). Usually, these will be client services giving access to all functionality that that backend offers itself, plus functionality offered by remote web services known to that backend.

### 7.3.2 A Generic Proxy for Accessing Remote Web Services

The primary server backend, which initially renders the documents annotated in compliance with [requirement 1](#), plays a strong role in the JOBAD architecture. Certain assistive functionality and sources of additional information are, however, clearly beyond the scope of primary backend system or have already been realized by existing external data sources or web services – for example the computational services discussed in sections [7.6.2](#) and [7.6.3](#). Depending on the environment chosen for implementing JOBAD, requests to external web services and data sources may have to be routed through the primary backend for security reasons. This is the case with JavaScript in contemporary browsers. The most generally applicable workaround, which the our implementation of the JOBAD architecture employs, is having the primary backend act as a proxy for remote web services. [Appendix C.2.1](#) explains our implementation.

### 7.3.3 User Interface Elements

JOBAD provides for context-sensitive and global user interface elements for input from the user, and for displaying information to the user. Context-sensitive input can be realized via a context menu, which the user requests by right clicking on an object in the document, or on a selection previously made with the mouse. Keyboard shortcuts can operate context-sensitively on a selection, or globally. Data input for controlling global display preferences, such as elision (cf. [section 7.4.2](#)), has not yet been generalized. The elision client service is currently controlled by top-level input boxes.

In accordance with [requirement 4](#), the implementation of our client services is largely independent from a concrete user interface. On initialization of a client service, an “action object” is added to a central registry, which is part of the state of the document in the browser. Among the properties of an action object, there are mappings of several input methods to functions to be executed: a  $key \mapsto function$  mapping for a keyboard shortcut pressed, a  $target \mapsto function$  mapping for a target in the document clicked, a  $target \times item \mapsto function$  mapping for one out of many context menu items clicked, and a  $target \mapsto function$  mapping for a target in the document hovered with the mouse. Any input generated by the user is forwarded to all registered action objects; we have not specified a mechanism for prioritizing between two client services declaring responsibility for the same input event. Where an action takes a *target* parameter, that means that a client service inspects the semantic annotation of the target in order to determine whether it has any functionality to offer here. Thus, the annotations act as *anchors* for client services.

The output of a client service may result in a part of the document, e.g. a mathematical expression, being rewritten, as in the cases of definition expansion (cf. [section 7.5.1.2](#)) and unit conversion (cf. [section 7.6.3](#)), in navigation to a different document (cf. [section 7.5.3](#)), or in information being displayed in a tooltip-like popup.

## 7.4 In-Document Client Services

The two in-document client services that we have developed so far deal with *elision*, which is a common practice in mathematics. Experienced mathematicians frequently use shorthand notations to avoid distracting the reader with information that is deducible from the context. In a first step, reading aids that support inexperienced readers in understanding the structure of an expression are removed, without affecting the well-formedness of the expression. Reading aids that we have investigated so far include redundant brackets and type annotations (cf. [section 7.4.2](#)). In a second step, entire ranges of symbols are removed from the expression and replaced by ellipses (usually “...”). Ellipses are commonly used in discrete sets, vectors, and, most outstandingly, in matrices [[SSo6](#)]; they abbreviate a finite or infinite range of discrete values that follows “obvious” construction rules, e.g.  $1, \dots, n$ , or is not relevant for the current mathematical consideration, e.g. when discussing diagonal matrices that may have *any* content off the diagonal [[SSo6](#)]:

$$\begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{pmatrix}$$

Ellipses leave an expression in a state that is no longer strictly well-formed but still rigorous.

We have investigated folding (i.e. dynamic showing and hiding) of subterms and elision of presentational structures that guide the reader (here: brackets and type annotations) so far; ellipses are considered a more complex topic subject to future work. The presentation process that we rely on (cf. [section 6.4.2.8](#)) operates in two steps: *composing* visual sub-presentations to larger ones, and then *eliding* parts of mathematical objects that can be deduced from the context. If the desired output format is interactive, such as XHTML+MathML, the latter parts are not actually removed from the output, but merely made invisible, so that client services like the ones described here can display them on request.

### 7.4.1 Folding Subterms and Undoing Interactions

The folding service makes abbreviations for subterms, which the author of the content markup has defined (cf. [section 6.4.2.3](#)), accessible from the user interface and allows for switching between a term and its abbreviation. Besides that, we provide for folding *arbitrary* subterms on demand – as far as they can be recognized from the presentation markup –, so that a reader can hide them if he feels distracted. When the user requests folding of a subexpression for the first time, we put both the original subterm and its folded version into a dynamically generated cache element in order to make the folding action undoable.

This is a general pattern, on which other client services rely as well: Whenever a client service rewrites a term  $t$  into  $t'$  (e.g. a folded subterm  $t$  into “...”), a cache element is created on the fly,

which preserves the previous state  $t$  of the term, so that the user can switch back to it. Not only do interactions become *undoable* locally, but they also become *redoable*: When information from a remote web service has been used to rewrite  $t \rightsquigarrow t'$ , as is the case, e.g., with unit conversion (cf. [section 7.6.3](#)), and the user switches back to  $t$ , he can recover  $t'$  without causing the information to be retrieved from the Web once more, as it is still cached in the document.

As an example, consider the expression  $[1 + [2 \cdot x]]$ , where square brackets denote subterms grouped in the presentation markup. Suppose the user selects [part of] the subterm  $2 \cdot x$  or right clicks somewhere in that term and requests it to be folded. Then, the expression will display as  $[1 + \dots]$ . Clicking on the dots and selecting the “unfold” action from the user interface (e.g. the context menu) will restore the original appearance.

[Appendix C.2.2.1](#) describes the details of our implementation.

## 7.4.2 Flexible Elision and Display of Reading Aids

### 7.4.2.1 Brackets

The most prominent presentational structure that aids reading mathematical expressions are brackets: They enclose subterms in order to explicate their grouping structure. However, they are usually omitted in two cases: (i) when the operator of the current subterm binds stronger than the operator of the enclosing term, as explained in [section 2.4.5.5](#), or (ii) when the current subterm is formed by a constructor that has outer fences itself, e.g. the set constructor; consider  $\{a, b\} \cap \{b, c\}$  vs.  $(\{a, b\}) \cap (\{b, c\})$ <sup>3</sup>. Particularly in the first case, bracket elision can be confusing for inexperienced readers who do not know the binding precedences of all operators in a complex expression. This becomes apparent when operators from multiple fields of mathematics occur together in one expression. Consider the following example<sup>4</sup>:

$$5(x + y)^{n+3} \leq (ab)! \vee \neg p \wedge \neg(q \leq \pi) \quad (7.1)$$

Here, some additional brackets clarify the structure, ...

$$(5(x + y)^{n+3} \leq (ab)!) \vee (\neg p \wedge \neg(q \leq \pi)) \quad (7.2)$$

... whereas more brackets already interfere with readability,

$$(5(x + y)^{n+3} \leq (ab)!) \vee ((\neg p) \wedge (\neg(q \leq \pi))) \quad (7.3)$$

... and the fully bracketed structure would finally be unreadably cluttered:

$$((5 \cdot (x + y)^{(n+3)}) \leq ((a \cdot b)!)) \vee ((\neg p) \wedge (\neg(q \leq \pi))) \quad (7.4)$$

The rendering algorithm that we employ (cf. [section 6.4.2.8](#)) enables a flexible elision of redundant brackets. That mode of rendering does not completely omit redundant brackets but merely hides them and annotates them with the difference between input and output precedence as the

<sup>3</sup>Such fences look like brackets but are not brackets in the strict sense, as they may never be omitted.

<sup>4</sup>This example is, admittedly, contrived, but cases with operators from two or three fields are common, e.g. expressions with set operators and logical operators.

# Flexible Elision Demo

- Powered by [JOMDoc](#) and [JOBAD](#)
- Tested with [Firefox ≥ 2.0](#) and [Opera ≥ 9.0](#)



$$((5 \cdot (x+y)^{(n+3)}) \leq ((a \cdot b)!)) \vee ((\neg p) \wedge (\neg(q \leq \pi)))$$

$$\Lambda^*_{(I \rightarrow O) \rightarrow ((I \rightarrow O) \rightarrow (I \rightarrow O))} = \lambda F_{I \rightarrow O} G_{I \rightarrow O} X_l.F_{I \rightarrow O}(X_l) \wedge_{O \rightarrow (O \rightarrow O)} G_{I \rightarrow O}(X_l)$$

$$5 \cdot (x+y)^{n+3} \leq (a \cdot b)! \vee \neg p \wedge \neg(q \leq \pi)$$

$$\Lambda^* = \lambda F G X_l.F(X_l) \wedge G(X_l)$$

Visibility Thresholds:

- brackets: ☐ 0 ☐ 100 ☐ 200 ☐ 300 ☐ 400 ☐ 500 ☒ infinite
- types: ☐ 0 ☐ 100 ☐ 200 ☒ 300

Operator	Mixfix declaration
$x^y$	$\boxed{199 _1} \boxed{\infty _2} : 200$
$!$	$\boxed{300 _1} ! : 300$
$\cdot$	$\boxed{400 _1} \cdot \boxed{400 _2} : 400$
$+$	$\boxed{500 _1} + \boxed{500 _2} : 500$

Operator	Mixfix declaration
$\neg$	$\neg \boxed{600 _1} : 600$
$\leq$	$\boxed{700 _1} \leq \boxed{700 _2} : 700$
$\wedge$	$\boxed{1000 _1} \wedge \boxed{1000 _2} : 1000$
$\vee$	$\boxed{1200 _1} \vee \boxed{1200 _2} : 1200$

Figure 7.2: Demo of bracket and type elision with global visibility threshold control and color depending on elision levels

*elision level*, which can be an integer number, or one of the two special values *infinity* or *-infinity*. Thus, the decision whether brackets should be displayed can be deferred to the time of reading a document. The user can then set a *visibility threshold* for elision levels; any bracket with an elision level below or equal to the threshold would be displayed.

The sequence of successively displaying more brackets from [formula 7.1](#) to [7.2](#) to [7.3](#) above can be achieved by giving the operators involved the following precedences<sup>5</sup>:

$$\begin{array}{cccc} \neg & \leq & \wedge & \vee \\ \hline 600 & 700 & 800 & 850 \end{array}$$

The pairs of brackets that are additionally displayed in [formula 7.2](#) thus have elision levels of 150 ( $\leq$  vs.  $\vee$ ) and 50 ( $\wedge$  vs.  $\vee$ ), which means that a visibility threshold of at least 150 is required to achieve that appearance. Increasing the threshold to 250 would lead to state [7.3](#). In the demo

<sup>5</sup>... assuming same values for input and output precedence for simplicity, and lower precedence values for operators that bind stronger

implementation shown in [figure 7.2](#), we have additionally used the elision level information to display content with a high elision level in a lighter color.

Identifying reasonable precedence values is subject to further investigation once the elision client service has been deployed to a significant number of users.<sup>6</sup> It might turn out that the elision level a *user* would assign to a particular bracket depends on his background knowledge [of the operators involved] and other personal preferences.

#### 7.4.2.2 Types and other Elision Groups

Besides brackets, our renderer supports other *elision groups*. So far, we have investigated type annotations: Some of them are essential for determining the type of an expression, whereas others can be inferred from the context. Consider the following expression:<sup>7</sup>

$$\text{and}^{\mathcal{I}} = \lambda F G X_l. F(X) \wedge G(X) \quad (7.5)$$

Suppose it is background knowledge that the boolean conjunction operator  $\wedge$  is of type  $o \rightarrow o \rightarrow o$ . Then, all types in [expression 7.5](#) can be inferred from knowing that  $X$  is of type  $\iota$ : the types of the predicates, ...

$$\text{and}^{\mathcal{I}} = \lambda F_{\iota \rightarrow o} G_{\iota \rightarrow o} X_l. F(X) \wedge G(X) \quad (7.6)$$

... and the type of the complete expression:

$$\text{and}^{\mathcal{I}}_{(\iota \rightarrow o) \rightarrow (\iota \rightarrow o) \rightarrow \iota \rightarrow o} = \lambda F_{\iota \rightarrow o} G_{\iota \rightarrow o} X_l. F(X) \wedge G(X) \quad (7.7)$$

Finally, these are all type annotations:

$$\text{and}^{\mathcal{I}}_{(\iota \rightarrow o) \rightarrow (\iota \rightarrow o) \rightarrow \iota \rightarrow o} = \lambda F_{\iota \rightarrow o} G_{\iota \rightarrow o} X_l. F_{\iota \rightarrow o}(X_l) \wedge_{o \rightarrow o \rightarrow o} G_{\iota \rightarrow o}(X_l) \quad (7.8)$$

Note the interplay with bracket elision: As the function type constructor  $\rightarrow$  is right-associative, the type of the complete expression would render as  $(\iota \rightarrow o) \rightarrow ((\iota \rightarrow o) \rightarrow (\iota \rightarrow o))$  with all brackets displayed. Redundant type annotations are represented in the same way as redundant brackets, just with a different elision group annotation. Our elision client service allows the user to choose one *visibility threshold* per elision group in the document. If  $T_g$  is the threshold of elision group  $g$ , then all elements of group  $g$  whose elision level is above  $T_g$  are invisible.

Besides the hand-crafted demo document shown in [figure 7.2](#), we have not yet investigated elision levels on redundant type annotations further; however, this client service is under continuous evolution on the Logic Atlas site [\[KMR\]](#).

<sup>6</sup> A variant of this service is employed in the human-readable presentation of the OMDoc-based Logic Atlas [\[KMR\]](#), but usage statistics have not been collected so far.

<sup>7</sup> The base formula is courtesy of MICHAEL KOHLHASE and cited from his lecture notes on computational semantics of natural language. The formula defines the semantics of the “and” connective for verb phrases in natural language (e.g. “Ethel howled and screamed”) by reducing it to the boolean conjunction.

### 7.4.2.3 User Interface

The best user interface for controlling visibility thresholds has to be determined yet. We believe that the thresholds for different elision groups should be controlled separately. We also believe that it is adequate to the complex rules of when reading aids are redundant or not, if their display is controllable in a rather large scope. Our current implementations support control on object and document level, but not yet on the level of a complete document collection or system. It is likely that many distinct elision levels will occur in a document, but we believe that the reader will not care about their exact discrete values. Therefore, a slider seems most suitable for controlling the visibility threshold. Alternatively, if a slider is not available, a sequence of discrete radio buttons could be used, as shown in [figure 7.2](#), or a sequence of keyboard shortcuts (e.g. ranging from 0 to 9).

## 7.5 Symbol-based Client Services

Symbol-based client services retrieve information about a mathematical symbol or an RDF resource<sup>8</sup>. In the most fortunate case of a linked data setup (cf. sections [2.3.1](#) and [6.4.1](#)), they can simply be realized by dereferencing the URI of the respective symbol or resource. If linked data are not available, or if they would not be sufficiently flexible, e.g. in cases where the desired response depends on additional parameters, a web service that adopts the REST pattern [[Fie00](#)], where URLs also represent resources but optionally accept parameters, is easiest to contact.

Symbol-based client services send the URI of the symbol in question and, optionally, an identifier of the requested kind of information to a web service. As symbol-based client services for mathematical objects, we have so far realized support for looking up the definition of a symbol and, analogously, the type declaration. In-place expansion of definitions, as well as interactive notation switching are described conceptually; they have not yet been realized, but everything they need is in place. Finally, I explain two client services for non-formula markup: a generic navigation service and a domain-specific service that visualizes rhetorical structures.

### 7.5.1 Definition and Type Declaration Lookup

The user can activate the definition and type declaration lookup client service on every occurrence of a symbol in a mathematical object. The client service sends the URI of the symbol to the server and expects as a response a content markup object containing a term that defines the symbol, or, analogously, the type of the symbol. This result is then rendered and displayed in a popup window. This complements the linking of symbols to the place where they are declared (cf. [section 6.4.2.8](#)) by a mode of interaction that does not force the reader to abandon his current context for a quick lookup – whereas traversing the link from a symbol occurrence to its declaration offers the reader the possibility to understand the symbol in its theory or CD context. This section explains how the lookup client service has been realized so far and discusses how a future extension of this client service to in-place definition expansion could be realized.

<sup>8</sup>In terms of the ontologies presented in [section 3.2](#), mathematical symbols are merely a special case of RDF resources; however, I treat them separately here due to the different ways of annotating them (MathML vs. RDFa).

Listing 7.1: A symbol and its definition in OMDoc 1.2/1.3

```

<symbol name="sin">
  <meta property="dct:description">the sine function</meta>
  <type><OMOBJ> $\mathbb{C} \rightarrow \mathbb{C}$ </OMOBJ></type><!-- Content markup omitted to save space -->
</symbol>
<definition for="sin" type="simple">
  <OMOBJ> $\sin z = \frac{1}{2i}(e^{iz} - e^{-iz})$ </OMOBJ>
</definition>

```

The syntax for representing symbols and their definitions and type declarations constrains the options for realizing lookup. In OMDoc 1.2/1.3, the type declaration of a symbol  $\sigma$  is given as a child element, whereas the definition  $\text{def}(\sigma)$  is given as an element in the same theory, referencing the symbol by its (theory-local) name, as shown in [listing 7.1](#). Both are optional. As OMDoc 1.2/1.3 uses hash URIs for all markup elements, including symbol declarations and definitions, a linked data approach would force the client to download a complete theory and then locate the type declaration or symbol<sup>9</sup>, as discussed in [section 6.4.1.3](#). In OMDoc 1.6, simple definitions of the form “ $\sigma := \text{def}_f(\sigma)$ ”<sup>10</sup> are given as children of *symbol* as well. For strict markup, it is sufficient to have simple definitions<sup>11</sup>, whereas pragmatic markup will likely continue to have pattern-based, inductive, and implicit definitions to capture textbook practice (see “definition expansion” below). Moreover, symbols have “hash-like” MMT URIs, as discussed in [section 6.4.1.3](#), which facilitates a linked data approach. Linked data definition/type lookup from OpenMath CDs would once more suffer from the use of hash URIs, forcing the client to retrieve a complete CD – or signature dictionary in the case of type lookup –, and the circumstance that CDs are often larger than the little theories advocated by OMDoc (cf. [section 2.1.2](#)). Moreover, the most reasonable result of a definition lookup operation would be the *CDDefinition* element for the respective symbol, which may contain a lot of non-definitional properties. [Appendix C.2.3.1](#) explains our current implementation for OMDoc 1.2/1.3.

On the client, our current realization of lookup displays  $\rho(\text{def}(\sigma))$ , where  $\rho: \text{content} \rightarrow \text{presentation}$  is the rendering web service explained in [section 7.6.1](#), in a tooltip overlay at the cursor position, as shown in [figure 7.3](#).

### 7.5.1.1 Content Negotiation Between Content and Presentation Markup

As the desired MIME type of the response may be indicated in the HTTP request header for content negotiation (explained in [section 6.4.1.3](#)), the server backend can distinguish requests for

<sup>9</sup>It does not make a difference whether the client requests OMDoc 1.2/1.3 content markup or presentation markup from the URI. In the former case, it would get at least the whole containing theory – i.e., actually: the document that contains at least this theory, possibly additional ones – as OMDoc, and would then have to extract the fragment to be rendered by a subsequent call to the rendering web service. In the latter case, it would get the presentation markup of the containing theory.

<sup>10</sup>This is not to be mistaken as a definition of  $\sigma$  in terms of itself; it denotes  $\sigma$  being defined as some expression, where the expression is *structurally* a function of  $\sigma$ , spoken “the right hand side of the definition of  $\sigma$ ”.

<sup>11</sup>Implicit definitions, for example, can be written as simple definitions employing a special operator that returns the solution of an equation (if it exists and is unique).



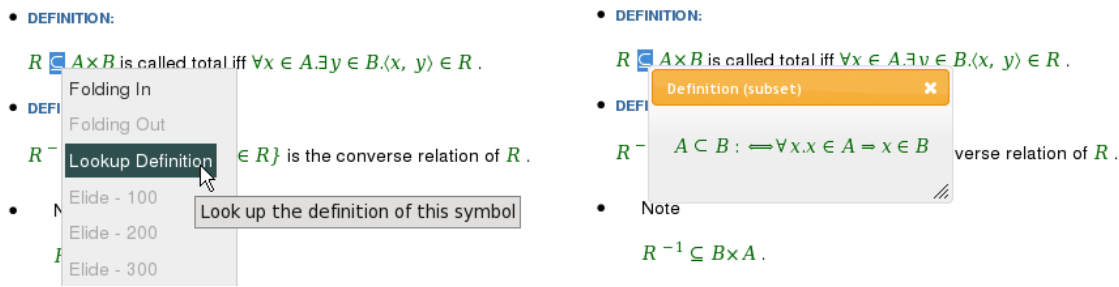


Figure 7.3: Looking up a definition (left: selecting the action, right: the result); example taken from MICHAEL KOHLHASE's lecture notes

Listing 7.2: A request for the OpenMath source of a definition

```
GET /backend?action=lookup-definition&cdbase=...&cd=transcl&name=sin HTTP/1.1
Host: jobad.mathweb.org
Accept: application/openmath+xml
```

content markup from requests for a rendered mathematical object while still using the same URL, as shown in [listing 7.2](#). Analogously, using the MIME type *application/xhtml+xml* would yield a response rendered in XHTML with Presentation MathML. Retrieving content markup, such as OMDoc or OpenMath, makes sense for definition expansion (see below); it is even required when the definitions to be looked are not accessible from the same host as the rendering web service and thus the rendering web service would have to be called separately. Directly retrieving presentation markup is more efficient when looking up definitions using a backend that hosts both the definitions and the rendering web service.

### 7.5.1.2 Definition Expansion

An alternative to definition lookup is in-place definition *expansion*, which *replaces* an occurrence of a symbol in a mathematical object with its definition. This works differently depending on the type of definition. OMDoc supports the following types, which are common in mathematics [[Koh06b](#), section 15.2.4]:

**simple:** a plain symbol  $\sigma$  is defined by an expression  $\sigma := \text{def}_f(\sigma)^{12}$ , whose right hand side can be substituted for  $\sigma$ ; consider the definition of one as the successor of zero:  $1 := s(0)$ .

**pattern:** a symbol that can be applied to arguments is defined by an expression, in which the arguments occur, but not the original symbol; consider the above-mentioned definition of the sine function by the equation  $\sin z := \frac{1}{2i}(e^{iz} - e^{-iz})$ . There can also be several cases, e.g. for positive vs. negative arguments.

<sup>12</sup>Here,  $\text{def}_f$  denotes the right hand side of a definitional equation, in case it exists uniquely.

**inductive:** like “pattern”, but the symbol to be defined may also occur on the right hand side of an equation. Integer addition, for example, can be defined by recursion on the second argument:  $x + 0 := x, x + s(y) := s(x + y)$ .

**implicit:** An equation is given that has exactly one solution for the symbol to be defined; consider the example given in [listing 3.1](#) on page 101.

The first step of definition expansion is a content markup to content markup transformation. In case of a simple definition, the symbol can simply be replaced by the right hand side of its definition. For pattern-based and inductive definitions, the equation for the right case has to be found by matching against the occurrence of the symbol in the mathematical object. Then, the symbol and its arguments can be replaced by the right hand side of that equation, substituting the correct values for the arguments. The unification required here may be non-trivial, so we recommend leaving it to a symbolic computation web service that has access to the same theories or CDs. Simple cases, however, such as matching a left hand side that consists of a function with atomic arguments – like  $\sin z$  in the example above –, can be realized by straightforward content markup processing on the JOBAD client side. Finally, expanding implicit definitions is not possible at all.

Several interaction modes for definition expansion are conceivable. If a symbol occurs more than once in a document, one could either only replace the occurrence the user selected, or all occurrences. For an inductive definition of a symbol  $\sigma$ , the client service could perform one step of expansion only, then requiring the user to request the next step for the remaining occurrence(s) of  $\sigma$ , or one could offer a user interface that allows to request the desired number of expansion steps – a positive number, or “maximum”, i.e. until termination – at once.

So far, I have outlined how to expand definitions in content markup, but the content markup expression resulting from this expansion will ultimately have to be rendered. As long as compositional notation definitions are involved ([section 2.4.5.4](#)), it suffices to render the expanded expression  $e := \text{def}_f(\sigma)$ , and to substitute  $\rho(e)$  for the original occurrence of the symbol in the presentation markup. This is possible thanks to the fine-grained annotations that [requirement 1](#) demands (e.g. cross-linked parallel markup; cf. [section 6.4.2.2](#)). However, according to [requirement 3](#), definition expansion must not violate the integrity of the cross-linked parallel markup of the whole mathematical object. Therefore, we also have to substitute the content markup part of  $\rho(e)$  for the content markup counterpart of  $\sigma$  in the content markup of the object.

This approach will no longer work once non-compositional pattern matching notation definitions are involved. Suppose the symbol  $\sigma$ , defined as  $\sigma := \tau$ , occurs in a mathematical object as  $f(\sigma)$ , and a special notation has been defined for  $f(\tau)$  – then, the latter expression would have to be re-rendered. A sophisticated approach to this problem would probably maintain information about the compositionality of notation definitions involved in rendering an object in the markup created by the renderer and then re-render the minimum necessary subterm. Given that (i) mathematical objects in documents are usually quite small, (ii) the client would have to inquire the compositionality of a notation definition by another call to the server<sup>13</sup> – unless this information is added to the rendered object as a special annotation –, and (iii) we can plausibly expect interactive definition expansion to be used infrequently and highly selectively, the pragmatic approach of

<sup>13</sup>... which can be supposed to consume more time than rendering a mathematical object on the server and substituting markup on the client

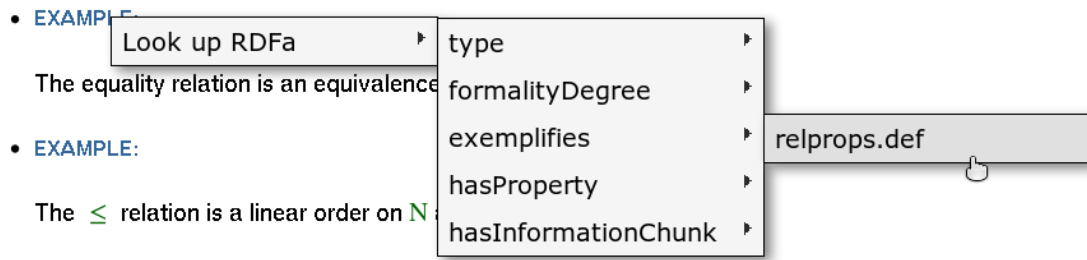


Figure 7.4: Navigating from an example to the concept it exemplifies

simply substituting  $\sigma$  by  $e$  in the content markup and re-rendering the whole object seems reasonable – however, in that case, definition expansion would no longer be undoable *locally* using the technique introduced in [section 7.4.1](#), but the substitution of the whole mathematical object would become one undoable action.

## 7.5.2 Interactive Notation Switching

Renderer support for alternative notations per symbol (cf. [section 6.4.2.8](#)) enables the realization of a client service for interactively switching among them. As set out initially in this chapter, such a client service would complement an adaptive document generation in situations where the server backend knows too little about the user’s [notational] preferences.

A renderer can leave information on what renderings from notation definitions have been used when rendering each symbol in the generated markup, as explained in [section 6.4.2.4](#). When the user wants to change the rendering of a symbol  $\sigma$ , the server is queried for the URIs and textual descriptions of all alternative renderings<sup>14</sup>, given the URI of the rendering currently used. A menu is populated with this information, and the user can select the desired rendering. Then, the content markup of the mathematical object is sent to the rendering web service (cf. [section 7.6.1](#)), specifying the URI of the desired alternative rendering as an “extensional context” in an *@ec* attribute on the occurrence of  $\sigma$  in the content markup, as specified in [[Mühloua](#), chapter 4.4.1], thus requesting the object to be re-rendered, using the desired rendering for  $\sigma$ .

As said for definition expansion above, the compositionality of the notation definitions involved influences whether re-rendering can be confined to the subterm for whose head operator an alternative notation has been chosen, or whether the whole mathematical object has to be re-rendered. As with definition expansion, the question of how to design interaction in the case of multiple occurrences of the same symbol in one document remains open, i.e. whether to change the notation only for the occurrence selected by the user, or for all occurrences at the same time.

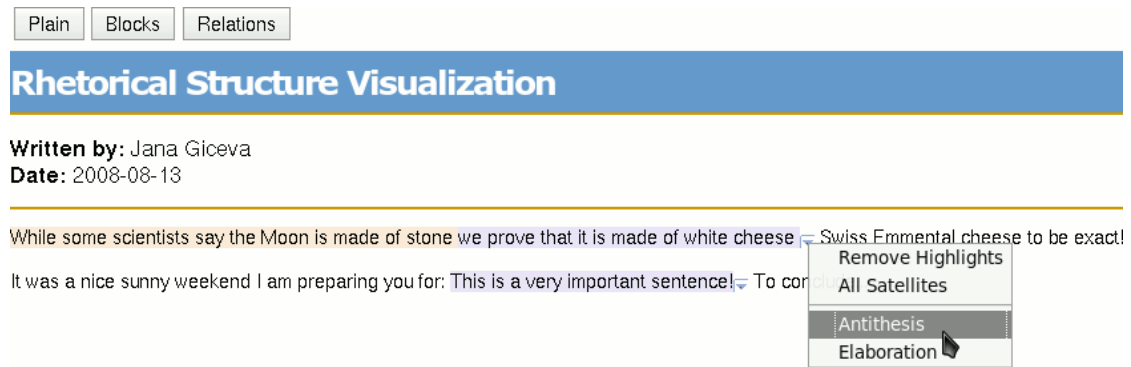


Figure 7.5: Visualization of rhetorical relations annotated in terms of the SALT ontology

### 7.5.3 Linked Data Navigation

As an analogous client service to definition lookup, navigation to link targets is available for all resources above the object level. Our current implementation (see [appendix C.2.3.2](#) for technical details) is a proof of concept that makes all links going out from the currently selected resource accessible in a context menu, as shown in [figure 7.4](#). Note that, for the example element clicked, no other client service is available. Definition lookup, the other client service enabled in that document, is only available on symbols in mathematical objects and therefore cannot be selected here. The navigation client service offers the same functionality as the other linked data browsers discussed in [section 6.4.1.2](#) – as locally as supported by the granularity of the annotations in the document. Making all annotations available in a uniform way and ignoring that they involve several, quite different structural dimensions of knowledge makes this client service mainly useful for debugging purposes. The following section discusses an example of a user interface tailored to a specific subset of the annotations. The current implementation of the navigation client service navigates to the selected link target; a complementary in-place lookup, having analogous advantages and disadvantages as discussed for definition lookup in [section 7.5.1](#), could also easily be realized within our implementation.

### 7.5.4 Visualizing Rhetorical Structures

The navigation client service introduced in the previous section does not filter annotations by relevance (the only really relevant property in [figure 7.4](#) is *exemplifies*, whereas the *type* and *formalityDegree* information is irrelevant to the reader, who can recognize the same information from the rendered output), and it makes all of them available via the same uniform user interface. For the particular case of rhetorical structures annotated in terms of the SALT ontology (cf. [section 3.3](#)), we have realized a specific mode of interaction [[Gico8](#)]. [Figure 7.5](#) shows how RST-style rhetorical relations are visualized: Every nucleus gets equipped with a small button that

<sup>14</sup>The actual implementation that currently exists provides a whitespace-separated list that, following the rendering that has been used, also contains all alternative renderings known to the renderer. However, the notation switching client service will have to contact the server once more anyway in order to obtain descriptive labels for the alternative renderings.

gives access to a popup menu, from which the user can select the corresponding satellites to be highlighted by type of rhetorical relation.

## 7.6 Expression-based Client Services

Expression-based client services send complex content markup expressions to a web service, most reasonably in the body of an HTTP request. The client side of the rendering web service, which receives a content markup expression and returns a rendered fragment of presentation markup, has already been mentioned, as many other client services use it. Other than that, we have realized two computational client services – a generic one that looks up computationally related information from the Wolfram Alpha “computational knowledge engine”, and a domain-specific one that converts units using a third-party web service. The motivation for that was, on the one hand, the relevance of such services for science and engineering. On the other hand, they also formed the first test case for utilizing external web services, after we had realized the in-document client services and the client services driven by the web services offered by our own primary backend(s), i.e. at a time where the other key functionalities of the JOBAD implementation – document manipulation, client/server communication including a proxy for connecting to remote hosts, user interface elements, rendering – had already been in place.

### 7.6.1 Rendering as a Web Service

The client side of the rendering service is not accessible to the user in itself, but it is a prerequisite for making the output of other services human-readable. In its simplest form, it sends a content markup fragment in the body of an HTTP POST request to its web service backend and returns the result of rendering it to presentation markup according to [requirement 1](#). This web service is most reasonably offered by the same backend that also provides the content markup and thus the notation definitions for the symbols to be rendered. (Recall that, in mathematical practice, new symbols and their notation are usually introduced at the same time; cf. [section 2.1.5](#)!)

### 7.6.2 Looking up General Computational Information with Wolfram Alpha

As a first step towards an integration of CAS into JOBAD-style interactive documents, we have developed a client service that looks up information that is related to the mathematical expression selected by the user in a computational sense [[DLR10](#)]. This information is obtained from the Wolfram Alpha “computational knowledge engine” [[Urlb](#)], a web frontend to the Mathematica CAS, combined with natural language processing capabilities and databases (e.g. of statistical facts), via its web service API [[Urlc](#)]<sup>15</sup>. For a mathematical expression, Wolfram Alpha returns everything that it knows about it and deems relevant, e.g. its factorization, its roots, or a plot. As Wolfram Alpha does not understand any content markup other than the Mathematica syntax, another web service has to be employed for the translation; [appendix C.2.4.2](#) describes the service composition in detail.

<sup>15</sup>Before January 2011, this API required an access key, which developers were not allowed to publish, e.g., in any JavaScript code. In our setup, the key was stored in the JOBAD proxy, which appended it, as a URL parameter, to any remote URL starting with <http://api.wolframalpha.com>.

Listing 7.3: A physical quantity in OpenMath

```
<OMA>
  <OMS cd="arith1" name="times"/>
  <OMF>1.5</OMF>
  <OMS cd="units_metric1" name="metre"/>
</OMA>
```

### 7.6.3 Unit Conversion – a Case of Domain-specific Computation

In physics and engineering, different unit systems are known (e.g. SI units vs. imperial units). Adequate software support is crucial to help engineers cope with these differences, as, for example, the loss of the Mars Climate Orbiter in 1999 demonstrated [Obeg99]. This spacecraft was destroyed in the Mars atmosphere due to a navigation error caused by the thrusters. These had been manufactured by a contractor and reported their performance in English pound forces, whereas the central controller expected Newton values. The \$125 loss could have been avoided by a stricter contract management and software validation. Our contribution to avoiding similar mistakes is making technical documents more comprehensible by enabling readers to interactively convert any unit they are not familiar with to a more familiar unit right in place, while focusing their attention on the document.

The unit conversion client service assumes the OpenMath encoding for physical quantities as specified in [DNo3] and demonstrated in listing 7.3: Base units are symbols from special CDs; derived units can be formed by multiplication or division of base units with numeric factors or other base units. The unit conversion client service accepts one such expression  $o$ , plus a target unit  $u_t$ . If a conversion is possible, the result is returned as an OpenMath expression (denoted by  $uc(o, u_t)$ ). On the client side, this result has to be integrated into the current mathematical object. Let  $p$  with  $o = c(p)$  be the presentation markup that the user selected; then we add  $p' = \rho(uc(o, u_t))$  as an alternative for  $p$  to the document and hide  $p$  in order to achieve undoability, as explained in cf. section 7.4.1.

The computation is performed by a web service according to the OpenMath *FMPs* of the unit symbols involved<sup>16</sup> [SDo8; Stro8]. Appendix C.2.5 describes the technical details of the connection to that web service.

Our initial user interface offers conversion to a small, fixed set of target units, as shown in figure 7.6. Internally, the unit conversion web service knows whether a conversion is admissible, from analyzing the *FMPs* that define the unit symbols and constructing a graph of possible conversions. If this information were exposed via an additional web service interface, the interaction with the unit conversion client service would work as follows, assuming a context menu user interface:

---

<sup>16</sup>In absence of definitional *FMPs* (cf. section 2.4.3), the unit converter assumes that there is always at most one *FMP*, which is definitional and usable as a conversion rule.

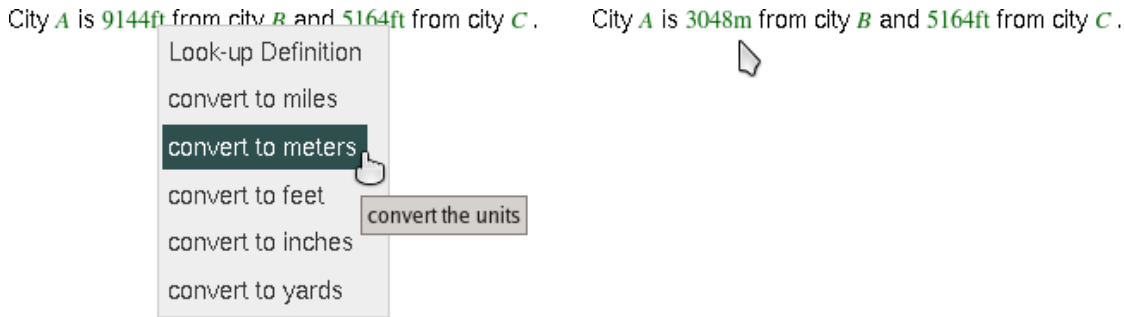


Figure 7.6: Converting the unit of a physical quantity with JOBAD

1. The user opens the context menu for the selected expression. When this expression is recognized as the product of a number and a unit (a simple XPath node test on the content markup<sup>17</sup>), the unit conversion submenu is enabled.
2. When the user opens the unit conversion submenu, a query for possible target units reachable from the source unit  $u_s$  is sent to the unit conversion web service. The web service returns a list of symbol identifiers and labels, which are used to populate the submenu.
3. When the user picks a target unit  $u_t$ , the conversion is performed, as described above.

While such a context menu enables local conversion, a reader's preference for certain units is actually global and should be part of a user profile or user model, as set out initially in this chapter – complemented by interactive on-demand unit conversion.

## 7.7 Conclusion and Future Work

We have designed a lightweight architecture for making mathematical documents interactive, plus a preliminary implementation with a representative collection of assistive embedded services. JOBAD turns documents published on the Web into command centers for client services that the user can activate on demand in order to adapt the appearance of the document, or to look up – as locally as possible – additional information related to the content of the document. If the semantic structures of these published documents are sufficiently annotated, our implementation shows that just a thin layer of client-side scripting code, anchoring services to the semantic structures on which they operate, has to be added in order to achieve the desired interactivity. For some client services, the annotations in the document already provide all required information, whereas other client services have to be initialized with instructions on where to retrieve their information from. This source of information can be the primary backend that has initially served the rendered document – an arrangement that makes sense when that backend is a knowledge base containing semantic representations of these documents –, or it can be any other external web service or other data source. This possibility to retrieve external information and to integrate it into the

<sup>17</sup>... if we assume an easy check for whether a symbol is a unit. So far, all names of unit CDs with conversion rules start with *units\_*.



display of the local document effectively turns the document into a powerful semantic *mashup* (cf. [section 1.3.3](#)). We have concretely demonstrated that with the lookup of computationally related information from Wolfram Alpha and the in-place unit conversion powered by a web service.

We have developed a diverse range of client services within the JOBAD architecture – in-document client services, symbol-based client services, and expression-based client services, some powered by the primary server backend, others by external data sources or web services. We claim that the design of the JOBAD architecture – the use of standard annotation markup, the independence of service functionality and user interface elements, and the communication via RESTful HTTP interfaces – enables the integration of arbitrary mathematical web services or data sources, as well as new user interaction elements, with little implementation effort. Validation of this claim by third-party developers is pending, but we have obtained preliminary evidence for it when implementing the unit conversion client service. There, the most laborious step was adapting the string-oriented interface of the web service to our OpenMath interface. Most of the other required functionality was already available from other JOBAD components and just had to be composed. To the context menu, we had to add a submenu of target units. Checking whether the selected term is a quantity reduced to an XPath node test on the corresponding content markup. Sending a string to the web service and obtaining a the response is a standard JavaScript function. Rendering the result of the conversion is done by another JOBAD service. Finally, replacing two XML subtrees in a mathematical object – both in the presentation and the content markup – and caching the previous presentation markup is a core JOBAD utility function also used by other client services.

The KWARC research group has developed further client services, which I have not mentioned in detail [[Job](#)]:

- a dialog for selecting the client services to be enabled in a document, which stores its settings in a cookie [[DLR10](#)],
- a client service that communicates with desktop applications via a locally installed communication daemon,
- an integration with a web discussion forum application that enables discussions about individual mathematical objects, but not argumentative so far (cf. [section 11.3.1](#)),
- a simple feedback form for reporting errors about any XHTML element in the document, and
- a generalization of the folding and elision client services presented in [section 7.4](#) to conditional visibility of arbitrary fragments of a document [[KMR](#)] – however, depending on specific annotations generated by the MMT backend.

Future directions for service integration, which are conceivable in the context of this thesis, include:

- extending the client service selection dialog to store the desired configuration in a server-side user profile,
- logging *all* interactions with the document to a server for analyzing usage patterns,

- giving quick local access to an editor from a JOBAD-enabled user interface, e.g. leveraging the notation annotations introduced in [section 6.4.2.4](#),
- extending the folding/visibility client service to arbitrary semantic structures annotated with RDFa (e.g. for folding proof steps, as imagined in the mockup in [figure 4.1](#) on page 156),
- extending single-step navigation to guided tours along dependency paths, and
- searching for other occurrences of the selected expression, or for similar expressions, with a text, formula, XML, or RDF search engine.

The impact of local, context-sensitive document adaptation and information lookup on usability, particularly its contribution to the *comprehensibility* of mathematical knowledge represented in a JOBAD-enriched way, will require future evaluation with test users. Some interaction paradigms employed by JOBAD – local interaction with a document published on the Web and symbol-based navigation through a collection of mathematical knowledge – are covered by the evaluation of the SWiM semantic wiki presented in [chapter 10](#).<sup>18</sup> We have enabled JOBAD for MICHAEL KOHLHASE’S OMDoc lecture notes in order to realize the scenario introduced in [section 6.1.2.4](#), but not yet in the place where the students would usually access them and discuss their problems [Koh+b]. In the latter setting, individual client services could be evaluated by letting two groups of test users – one with JOBAD enabled, one with JOBAD disabled – solve exercises based on the lecture notes and assessing their speed and the accuracy of the replies.

For conceptual clarity of the JOBAD architecture, I have treated all web services as independent black boxes. From an efficiency point of view it does, however, make sense to couple multiple web services more tightly in one backend. Consider, for example, unit conversion: The web service we employed internally relies on OpenMath CDs that declare unit symbols and define conversion rules [Stro8]. The definitions of the unit CDs would reasonably be looked up from the same CDs. Last but not least, the rendering web service needs notation definitions for the unit symbols, which, by default, would also be provided in the CDs. Offering three independent web services for these tasks would require redundant data storage. An integrated backend would also save time; consider the case of definition lookup: With separate lookup and rendering web services, a JOBAD-enabled client has to connect to two web services in succession. An integrated backend could, however, offer readily rendered definitions by composing two of its internal functions and only minimally extending its external HTTP interface to provide content negotiation between content and presentation markup (cf. [section 7.5.1.1](#)). The TNTBase backend, for example, which powers many of the JOBAD client services developed so far, is extensible by plugins. Besides extensibility there is, however, another prerequisite to service integration: translation between different representation formats supported by different [web] services, such as XML vs. RDF. This topic is addressed in [chapter 8](#).

<sup>18</sup>Historically, SWiM predated JOBAD. Functionality originally prototyped with SWiM has then been factored out into JOBAD client services and web services for reuse (cf. the discussion in [section 9.6](#)), whereas JOBAD remains to be integrated back into SWiM.

## Acknowledgments

The JOBAD architecture for integrating services into interactive documents has been developed jointly with FLORIAN RABE and has first been implemented by JANA GIČEVA under our joint supervision [GLR09]; this includes the initial versions of the folding, definition lookup, and unit conversion services. CATALIN DAVID has been continuing the implementation since 2009 and particularly redesigned the action objects, the context menu widget, and implemented the RDFa navigation service [DKL+10b] and the Wolfram Alpha service [DLR10]. The work on flexibly eliding and displaying reading aids presented in [section 7.4.2](#) has mostly been done before the conception of the proper JOBAD framework, in collaboration with MICHAEL KOHLHASE and FLORIAN RABE. Definition lookup with the TNTBase backend has been realized in collaboration with VYACHESLAV ZHOLUDEV and initially documented in [KGL+09]. JONATHAN STRATFORD kindly provided support for his unit converter [SD08; Stro8], whose integration is presented in [section 7.6.3](#). JANA GIČEVA has implemented the visualization of rhetorical structures under joint supervision of TUDOR GROZA and me [Gico8], with further advice from SIEGFRIED HANDSCHUH and MICHAEL KOHLHASE. The development of the Wolfram Alpha service has been supported with a pioneer grant from Wolfram Research for accessing their web service API; and JAN WILLEM KNOPPER from the MathDox has supported us in using their translation web service.

# Transparent Translations in Knowledge Bases

*Die Mathematiker sind eine Art Franzosen:  
redet man zu ihnen, so übersetzen sie es in ihre Sprache,  
und dann ist es alsobald ganz etwas anders.<sup>1</sup>*

—JOHANN WOLFGANG VON GOETHE [Goe06]

Making knowledge and knowledge-based services reusable requires translation. Knowledge in foreign repositories may be represented in different languages; even different services operating on the same repository may use different knowledge representation formats. Consider, for example, the case of developing a software assistant that supports a team of theoretical physicists in computing transformations of non-Riemannian hypersquares (cf. [section 2.1.1.4](#)) and planning an analysis of data collected from observing black holes to see whether they expose certain properties of non-Riemannian hypersquares. Suppose the CAS used by the physicists does not yet know non-Riemannian hypersquares, and suppose they want to reuse the rich astronomical vocabulary of the SWEET ontologies (cf. [section 3.5.2](#)) for conceptually modeling the planned analysis. Let there be a seminal paper on the topic, defining all required formal concepts, on the arXiv [[Arx](#)]. One team member might download its L<sup>A</sup>T<sub>E</sub>X sources and annotate the formal mathematical concepts with an S<sup>T</sup>E<sup>X</sup>-enabled editor (cf. [section 6.2.1.1](#)) to prepare their reuse. The straightforward ability to translate S<sup>T</sup>E<sup>X</sup> to OMDoc would already allow for sharing an interactively enhanced version of the document with the team colleagues. For preparing a phrasebook that teaches the CAS about non-Riemannian hypersquares, they would, however, have to translate the OMDoc document to an OpenMath CD. Another colleague might model and validate the data analysis plan in the Protégé ontology editor [[Proc](#)], using the SWEET ontologies. Talking about non-Riemannian hypersquares in this setting involves a translation of the OMDoc document to RDF in terms of the OMDoc ontology.

This chapter presents translations between different representations of mathematical knowledge, which knowledge base systems can execute *transparently*, thus relieving users from manually exe-

---

<sup>1</sup>“Mathematicians are [like] a sort of Frenchmen; when you talk to them, they translate it into their own language, and forthwith it means something quite different.”

cutting them for individual files, and supporting developers who integrate services. The translations presented here do not depend on a particular storage backend. While a dedicated XML database (cf. [section 6.5.2.1](#)) certainly facilitates the management of knowledge represented in a semantic markup language, the choice of storage backend is often influenced by external technical constraints, historical legacies, or compatibility concerns. Other researchers have also acknowledged that; NORMEN MÜLLER's approach to change management on semi-structured documents, for example, does not assume a specialized database but also works in the file system [[Mül10b](#)].

[Section 8.1](#) covers translations from XML-based semantic markup languages to RDF, so that, e.g., existing query and reasoning engines can use it. For that purpose, I have developed Krextor, a generic library that allows for quickly implementing such translations for any given input XML language and target ontology. [Section 8.2](#) discusses the opposite direction: translating existing knowledge from a language with limited expressivity – here concretely: OWL –, to a more expressive language – here: OMDoc – for the purpose of enriching and refining its formalization and documentation and ultimately reusing it in a wider setting. [Section 8.3](#) discusses a specific problem that occurs when connecting a knowledge base system with special support for a certain representation language to a legacy file system based repository: how to adapt the representation to the internal needs of the knowledge base system on import without disrupting the structure of files exported back to the file system repository.

## 8.1 Extracting Structures from Semantic Markup

This thesis advocates the representation of mathematical knowledge in semantically rich XML markup languages (cf. [section 2.5.2](#)). In order to make this representation comprehensible to a wider range of services and to enable a reuse of mathematical knowledge on the Web of Data, I have formalized the conceptual models of these languages in ontologies and specified a translation in [chapter 3](#). RDF-based services that thus become reusable comprise metadata editing forms ([section 6.2.6](#)), metadata and link validation ([section 6.3.3.1](#)), linked data browsers ([section 6.4.1.2](#)), SPARQL query answering for general purposes ([section 6.5.2](#)) and for specific tasks such as problem-solving assistance ([section 6.6.3](#)).

This section presents the Krextor (KWARC RDF Extractor) library for XML→RDF translations. While OMDoc is the primary representation language recommended in this thesis, it is not the only one: (i) RDFa enables the annotation of mathematical structures in documents written in any XML language; in particular, it enhances OMDoc's coverage of structural dimensions (cf. [chapter 5](#)), (ii) the less expressive OpenMath CD language still has a right to exist, as argued in [section 3.2.3.1](#), and (iii) I have outlined the possibility of representing mathematical knowledge in markup languages for books and manuals in [section 2.4.8.6](#). Reusability and extensibility concerns have therefore influenced the design of Krextor; Krextor intends to facilitate the implementation of translations from arbitrary semantic XML markup languages to RDF by requiring little code for defining those XML→RDF mappings that occur frequently, while also supporting more complex ones. The primary goal was, however, supporting the translation of OMDoc+RDFa documents and OpenMath CDs to the ontologies relevant for this thesis, according to the requirements established in [section 3.7](#).

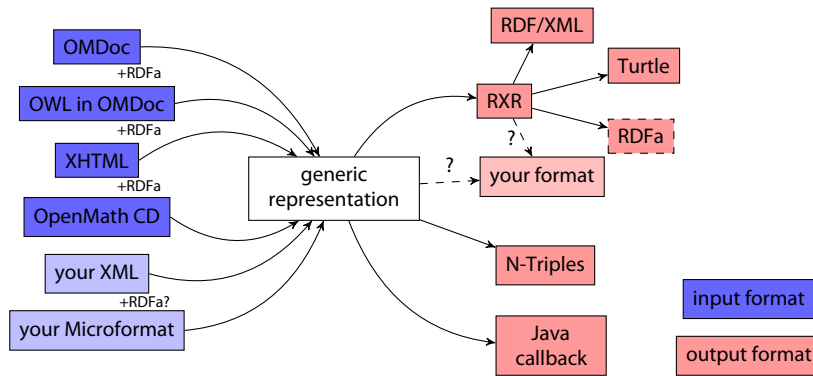


Figure 8.1: Krextor's extraction process and modules

After introducing the general architecture of Krextor in [section 8.1.1](#), [section 8.1.2](#) provides an overview of the semantic markup languages supported so far, focusing on the specific case of translating OWL ontologies implemented in OMDoc to their standard RDF representation in [section 8.1.3](#). [Appendix C.3.1](#) provides technical details about Krextor's XSLT-based implementation and explains how to implement support for new semantic markup languages and RDF output serializations. [Section 8.1.4](#) discusses the diverse landscape of related approaches, and [section 8.1.5](#) concludes with remarks on integration into knowledge base systems.

### 8.1.1 The Krextor XML→RDF Framework and its Translation Process

The Krextor framework allows developers to define translations from any XML language to many RDF serializations. [Figure 8.1](#) visualizes the translation process with all extraction and output modules currently supported.

#### 8.1.1.1 Extraction Modules and URI Minting Functions in General

A Krextor extraction module defines the translation from an XML language to an ontology. The generic core module provides convenience templates and functions for defining translation rules in a way that abstracts from a concrete output RDF serialization. Instead, the semantics of XML structures is defined on the level of the RDF semantics, using terms such as “resources” or “properties”; see [appendix C.3.1.4](#) for examples. Krextor's generic “representation” of RDF, depicted as the central box of [figure 8.1](#), is actually transient; the generic module is merely a step in the translation pipeline, grouping extracted data into triples and forwarding them to the selected output module. Algorithms [8.1](#) to [8.5](#) summarize the key functionality of the generic module.

Every RDF resource that an extraction rule creates from an XML node needs a URI or a blank node identifier. (The algorithm listings only show URIs.) Particularly for publishing linked data it is essential to mint URIs in a coherent, well-defined way. Krextor's built-in URI minting functions construct the URI of an RDF resource extracted from a document from the document's base URI and a fragment identifier. Built-in functions for generating fragment identifiers include reusing *@xml:id* attributes, generating unique identifiers for XML elements, etc. (cf. [appendix C.3.1.1](#) for details)

---

**Algorithm 8.1** Extracting RDF from a complete document in an XML language  $S$ :  $\text{extract}(d)$

---

**Require:**  $d$  is an XML document that is valid w.r.t. a schema  $S$

**Ensure:**  $R \in U \times U \times (U \cup L)$  is an RDF graph //  $U = \text{URIs}$ ,  $L = \text{literals}$ . This generic algorithm does not ensure that  $R$  is valid w.r.t. an ontology, as that depends on the extraction rules employed to be well-structured

$b \leftarrow \text{base\_uri}(d)$  // we need the base URI of the document for minting (fragment) URIs, and as an initial parent subject

$R \leftarrow \text{extract}(b, \text{root\_element}(d), b)$  // see [algorithm 8.2](#)

**return**  $R$

---

**Algorithm 8.2** Extracting RDF from one node of an XML document:  $\text{extract}(b, n, p)$

---

**Require:**  $b, p \in U$ ,  $n$  is an XML node

**Ensure:**  $R \in U \times U \times (U \cup L)$  is an RDF graph

$R \leftarrow \emptyset$

$r \leftarrow \text{rules}_S(n)$

**if**  $r \neq \perp$  **then** // if there is an extraction rule  $r$  for the current situation, ...

$R \leftarrow r(b, n, p)$  // ... execute it

Here, we assume rules to be curried functions invoking `create_resource` ([algorithm 8.3](#)), `add_literal_property` ([algorithm 8.4](#)), or `add_uri_property` ([algorithm 8.5](#)) with the parameters they take in addition to  $b, n, p$ . Calling  $r(b, n, p)$  completes the function call.

**end if**

**return**  $R$

---

Whenever an extraction module instructs Krextor to create a resource from an XML node, it either passes an explicit URI or blank node ID, or Krextor applies the global URI minting functions set up for the module. When the first one of them fails to mint a URI (e.g. if `@xml:id` is tried but there is no such attribute), Krextor tries the next one, and so on. When no URI minting function succeeds, Krextor does not extract any RDF from that node and its children. Otherwise, when a URI has been minted from an element and that element has children, Krextor recursively applies the given extraction rules to the latter, passing on the URI just minted for that *parent subject*, so that further rules can attach properties to the parent subject or create new resources related to it.

Extraction modules can provide their own URI minting functions. For example, when the OpenMath CD extraction module processes a *CDDefinition* element, it assumes that the input document has a base URI of the format `cdbase/cd` and generates a fragment ID from the *Name* child element, resulting in `cdbase/cd#name`.

### 8.1.1.2 Output Modules for Different RDF Serializations

Supported output formats (see [section 2.3.3.2](#) for an overview) include RXR, RDF/XML, Turtle, N-Triples, and a special callback interface for efficient integration into Java applications.

RDFa is a special case, as it is embedded into a host language. Therefore, Krextor does not offer a proper output module for RDFa but a set of utility functions to be called in the course of rendering a document in a semantic markup language to, e.g., XHTML. A renderer that generates



---

**Algorithm 8.3** Creating an RDF resource from an XML node: `create_resource( $b, n, p, u, T, P$ )`


---

**Require:**  $b, p, u, T, P \in U$ ,  $n$  is an XML node,

 $T$  is the URI of an ontology class or empty,  $P$  is the URI of an ontology property or empty

**Ensure:**  $R \in U \times U \times (U \cup L)$  is an RDF graph

```

 $R \leftarrow \emptyset$ 
if  $u = \varepsilon$  then                                     // if no explicit URI is defined by the rule, ...
     $u \leftarrow \text{mint}(b, n)$  // ... try to mint one, using built-in or custom minting functions (configurable
    // per extraction module)
end if
if  $u \neq \varepsilon$  then                                     // if we got a URI, ...
    if  $T \neq \varepsilon$  then
         $R \leftarrow R \cup \{ \langle u, \text{rdf:type}, T \rangle \}$  // make this resource an instance of the given class
    end if
    if  $P \neq \varepsilon$  then
         $R \leftarrow R \cup \text{add\_uri\_property}(\perp, p, P, u)$  // create a link (e.g. of a type like hasPart) from the
        // parent subject to this resource (algorithm 8.5)
    end if
    for all  $c \in \pi_{NS}(\$n / * | \$n / @*)$  do // from each element and attribute child node (determined
    // using an XPath evaluation function returning a nodeset) ...
         $R \leftarrow R \cup \text{extract}(b, c, u)$  // ... recursively extract RDF, using the newly created resource as a
        // parent subject
    end for // i.e. the recursion terminates for nodes without children
end if
return  $R$ 

```

---



---

**Algorithm 8.4** Adding a literal-valued property to a resource: `add_literal_property( $n, p, P, O$ )`


---

**Require:**  $p, P \in U$ ,  $n$  is an XML node,  $P$  is the URI of a literal-valued ontology property,  $O \in L$ 
**Ensure:**  $R \in U \times U \times (U \cup L)$  is an RDF graph containing one triple

```

if  $O = \varepsilon$  then                                     // if no explicit literal object is defined by the rule, ...
     $O \leftarrow \pi_S(\$n)$  // ... take the string value of the XML node
end if
 $R \leftarrow \{ \langle p, P, O \rangle \}$ 
return  $R$ 

```

---

**Algorithm 8.5** Adding a URI-valued property to a resource: `add_uri_property( $n, p, P, O$ )`


---

**Require:**  $p, P, O \in U$ ,  $n$  is an XML node,  
 $P$  is the URI of a literal-valued ontology property  
**Ensure:**  $R \in U \times U \times (U \cup L)$  is an RDF graph containing one triple  
  **if**  $O = \varepsilon$  **then** *// if no explicit URI object is defined by the rule, ...*  
     $O \leftarrow \pi_U(\text{resolve\_uri}(\$n))$  *// ... interpret the string value of the XML node as a URI*  
    **if** a URI syntax error occurred **then**  
       $O \leftarrow \varepsilon$   
      output a warning message  
    **end if**  
  **end if**  
   $R \leftarrow \{\langle p, P, O \rangle\}$   
**return**  $R$

---

presentation markup – e.g. an XSLT processor – has to call Krextor in RDFa output mode for every semantically relevant element of the input document while rendering it. Krextor then mints a URI for the resource represented by the current element. The renderer has to instruct Krextor to apply the same URI minting function as in the case of translating the original semantic markup to RDF, because the RDFa output routine looks up any further information about the current resource from a given triple store. In the easiest setup (as shown in [figure 6.9](#) on page 221), this is the same triple store that hosts the RDF triples that Krextor has previously extracted from the same source documents.<sup>2</sup> From that triple store, the RDFa output routine looks up all triples having the current resource as subject and adds them as RDFa annotations to the output.

The serializations generated by Krextor only partially utilize syntactic sugar that would improve human readability and save space. Semantically, that does not make a difference. After all, Krextor’s “target audience” are not humans, who usually do not want to read raw RDF anyway, but services that utilize the RDF output and make it accessible to users – as, e.g., the browsing interfaces explained in [section 6.4.1.2](#).

### 8.1.2 Extraction Modules for Mathematical Markup

[Figure 8.1](#) shows the Krextor extraction modules for mathematical markup languages that exist so far:

**OMDoc:** The OMDoc extraction module translates logical/functional structures using the OMDoc ontology, as specified in [section 3.2.2](#) and [appendix B.1](#), rhetorical and document structures using the SALT ontologies, as specified in [section 3.3.2](#), the metadata supported by the OMDoc 1.2 syntax, as specified in [sections 3.4.4](#) and [3.4.5](#), and arbitrary annotations made in RDFa, as specified in [section 5.2](#). This translation does not currently preserve the full structure of mathematical objects, for reasons discussed in [section 3.2.2.7](#).

---

<sup>2</sup>The triple store does not *have* to be populated using Krextor. The minimum prerequisite for generating RDFa output with Krextor is that RDF annotations for the document to be published are available at all, and that Krextor can reproduce their URI format when run in RDFa output mode.

**OWL in OMDoc:** Krextor can translate OWL ontologies implemented as OMDoc theories (cf. [section 4.3](#)) to the RDF representation of OWL instead of an RDF graph that uses the OMDoc ontology. Thus, existing OWL reasoners can use them, as detailed below in [section 8.1.3](#). This translation also considers RDFa annotations.

**OpenMath CDs:** The OpenMath CD extraction module translates the full structure of OpenMath CDs, signature dictionaries, and notation dictionaries, except the deep structure of OpenMath objects, to RDF, as specified in [sections 3.2.3](#) and [3.4.5](#) and [appendix B.2](#).

The RDFa extraction utility module has been designed for integration into any Krextor extraction module for a [semantic] markup language that hosts RDFa. In “strict” RDFa-compliant mode, it overrides all URI minting functions mentioned in [section 8.1.1.1](#) by the RDFa processing rules for identifying a new subject, as discussed for the case of OMDoc in [section 5.2.3](#); when a new subject cannot be determined, the element currently processed is considered semantically irrelevant, but the extraction nevertheless recurses down to the children. In contrast, in “pragmatic” mode, the RDFa processing rules only take precedence on elements that have RDFa attributes, whereas other URI minting functions may be applied otherwise.

### 8.1.3 Reasoning with OWL Ontologies Represented in OMDoc

[Chapter 4](#) introduces OMDoc as a more expressive alternative to the semantic web ontology languages OWL and RDFS. However, for accomplishing standard reasoning and information retrieval tasks, ontologies represented in OMDoc have to be translated back to the languages that the respective tools understand. This section explains how Krextor extracts the OWL/RDFS-compatible subset from an OMDoc implementation of an ontology to an RDF serialization.

The OWL extraction<sup>3</sup> is realized as a usual Krextor extraction module.<sup>4</sup> Extracting RDF triples from OMDoc symbol declarations, definitions, and axioms is straightforward, but minting correct URIs for entities of semantic web ontologies is more involved. Krextor traverses the graph of theory imports and collects the namespace URIs of all theories that carry an *oo:vocab* metadatum. Whenever it encounters a reference to a symbol *onto#sym* from an ontology whose implementation is an OMDoc theory *onto*, it mints the semantic web compliant URI by concatenating the namespace URI of the theory and the name of the symbol.

[Listing 8.1](#) shows the RDF generated from the example first introduced in [listing 4.1](#) on page 152 in Turtle serialization. The extraction represents the class, which a proven assertion is defined to be equivalent to, as a union class of a set of classes, and serializes it as an RDF collection. Most of the statement- and theory-level structure of OMDoc, such as the distinction between defined and inferred statements and theory morphisms, is lost and uniformly translated to less expressive OWL axioms; preserving the informal documentation of the OMDoc definition as an OWL 2

<sup>3</sup>I will only mention OWL henceforth, but, wherever the target of translation is an RDF serialization, this also comprises, analogously, RDFS.

<sup>4</sup>Additionally, we have formalized some of the translation rules from the OMDoc representation of OWL to its RDF representation in the OMDoc theory for OWL. There is, for example, a set of OMDoc axioms stating that an application of the *owl#Restriction* symbol to admissible arguments translates to an RDF resource of type *owl:Restriction* that has certain RDF properties.

Listing 8.1: RDF generated from an OWL ontology in OMDoc (somewhat pretty-printed)

```
oo: a owl:Ontology ;
    owl:imports foaf: .
oo:ProvenAssertion a owl:Class ;
    rdfs:comment "an assertion that has been proven" ;
    owl:intersectionOf (oo:Assertion
        [ a owl:Restriction ;
            owl:someValuesFrom oo:Proof ;
            owl:onProperty oo:provedBy ]) .
```

axiom annotation is not yet supported. The OWL extraction ignores any expressions that use constructs from theories other than OWL and RDFS, e.g. from more expressive logics.

[Listing 8.1](#) is the result of post-processing Krextor’s actual output. The actual output, shown in [listing 8.2](#) in RDF/XML serialization – the only serialization of OWL that all OWL-compliant tools are required to understand! – is far less legible due to machine-generated namespace prefixes, a seemingly random order of resources, and lack of syntactic sugar for RDF collections. However, as stated in [section 8.1.1.2](#), this output addresses machines; [section 4.3.4](#) covers human-friendly rendering of ontologies. This output consumes more space, but it can be assumed that it can be parsed into the internal representation of an OWL ontology at least as efficiently as a serialization with human-friendly abbreviations, which are not relevant for the internal representation. Thus, one can consider Krextor’s OMDoc→RDF translation of OWL ontologies to work like a compiler that creates (RDF/XML) object code from a higher-level OMDoc source code.

### 8.1.4 Related Work and Discussion

Approaches to giving XML-based languages a semantics in terms of RDF and ontologies are diverse and range from pragmatic one-shot hacks, which translate a specific XML language to a specific serialization of an RDF graph using a specific ontology, to general approaches that work with arbitrary XML languages (or their implementation in a specific schema language), often relying on formal models. Examples for the latter have already been discussed in [section 3.8](#); implementations based on such model-based XML→RDF mappings usually expose a limited flexibility w.r.t. the target ontology or RDF instances, e.g. in that they always map elements to instances of classes, and attributes to properties of resources.

#### 8.1.4.1 Plain XSLT Implementations and GRDDL

Algorithmic XML→RDF translations not backed by a formal model or methodology have often been implemented as monolithic XSLT stylesheets that work for one source XML language and one output RDF serialization. GRDDL (Gleaning Resource Descriptions from Dialects of Languages [[Cono7](#)]) specifies a uniform way of linking from an XML instance document or from an XML Schema to implementations of transformations that extract RDF from this particular document, or from all instances of a schema, respectively. The GRDDL specification is not concerned with *how* to implement such a transformation, but it recommends XSLT. Krextor extraction modules

Listing 8.2: RDF generated from an OWL ontology in OMDoc (raw RDF/XML output)

```

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ns1="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:ns2="http://www.w3.org/2002/07/owl#">
  <rdf:Description rdf:nodeID="collection-d30e63">
    <rdf:first rdf:resource="http://omdoc.org/ontology#Assertion"/>
    <rdf:rest rdf:nodeID="collection-d30e63-1"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="collection-d30e63-1">
    <rdf:first rdf:nodeID="collection-d30e66"/>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="collection-d30e66">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Restriction"/>
    <ns2:onProperty rdf:resource="http://omdoc.org/ontology#provedBy"/>
    <ns2:someValuesFrom rdf:resource="http://omdoc.org/ontology#Proof"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://omdoc.org/ontology#">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Ontology"/>
    <ns2:imports rdf:resource="http://xmlns.com/foaf/0.1"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://omdoc.org/ontology#ProvenAssertion">
    <ns1:comment>an assertion that has been proven</ns1:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <ns2:intersectionOf rdf:nodeID="collection-d30e63"/>
  </rdf:Description>
</rdf:RDF>

```

are written in XSLT. Therefore, legacy translations implemented in XSLT can be migrated step by step to Krextor extraction modules, which are more flexible and reusable due to the possibilities to mint URIs in different formats, to reuse ready-to-use RDFa extraction routines, and to create alternative output RDF serializations. The command-line or Java wrappers for Krextor are not yet capable of selecting the right extraction module according to GRDDL links; instead, the extraction module has to be chosen explicitly.

While the translations built into Swignition [Ink], covering RDFa, several microformats, and legacy ways of embedding RDF into HTML, have been implemented in Perl, it can also apply XSLT translations referenced by GRDDL annotations. New extraction modules can also be implemented in Perl, however, requiring more lines of code than Krextor extraction modules.

#### 8.1.4.2 Declarative Mappings

The authors of XSDL (XML Semantics Definition Language<sup>5</sup> [LMY+04]) stroke a balance between the two extremes of a formal model and a purely algorithmic translation. They have done substantial

<sup>5</sup>not to be confused with the XML *Schema* Definition Language, formerly known as “XML Schema” (cf. section 2.3.2.2).

theoretical elaboration on a semantics-preserving translation of XML into RDF but also provide a concise declarative syntax mapping XML to OWL-DL. There is, however, to the best of my knowledge, no implementation of XSDL. WEESA [Reio5; RGJo5] offers a similar declarative mapping language based on XPath and a Java implementation. WEESA is remarkable in that the time complexity of its translation algorithm has been analyzed, and for its integration into a web publishing process. Comparably to the integration of Krextor's RDFa output utility module into an XSLT stylesheet, WEESA has been integrated into the Apache Cocoon web application framework and embeds the RDF extracted from the XML input into the XHTML output generated from the same input. However, as RDFa had not yet been available at the time of its development, WEESA embeds the complete extracted RDF as a single RDF/XML fragment. Krextor's fine-grained per-element RDFa annotation is potentially advantageous in use cases that only deal with a small fragment of the whole XHTML+RDFa page, such as copy/paste. DAVY VAN DEURSEN et al. have suggested another declarative mapping syntax [VDPM+o8]. An additional feature, compared to XSDL and WEESA, is that the target class to map an XML element to or the RDF value of a property generated from an XML node can be determined by a SPARQL query against a given ontology. For example, the resource generated from an element can be made an instance of that class from the ontology whose label matches the value of some attribute of that element. Krextor takes a position between plain XSLT and declarative mappings and offers a library of XSLT templates and functions that lowers the investment for implementing a translation from a new XML language to RDF, compared to plain XSLT, and enhances the reusability of such implementations, while also allowing a declarative syntax for certain types of mappings. These mappings are, however, interpreted at runtime. A declarative *input syntax*, which a preprocessor *compiles* into optimized but less human-friendly XSLT, could possibly offer a reasonable compromise between ease of implementation and performance. As XSDL, WEESA, and VAN DEURSEN's syntaxes also use XML and XPath, each of them would qualify as such an input language for Krextor, except for VAN DEURSEN's usage of SPARQL, to which Krextor does not currently have comparable functionality.

#### 8.1.4.3 Combinations of SPARQL and XML Queries

SparqPlug [CHMo8] is remarkable for offering, as Krextor, a library of functions for minting URIs. It differs from Krextor and the other related approaches discussed here in that it directly translates the Document Object Model (DOM [W3C]) of the XML input document to an RDF representation, whereas it performs the actual mapping to the target ontology on the RDF side, using SPARQL *CONSTRUCT* rules. Reducing XML→RDF translations to RDF→RDF translations actually gives access to other mechanisms for rules in RDF, such as the first order rules of N<sub>3</sub> (cf. [section 2.4.10.1](#)). What makes SparqPlug mappings cumbersome is that there is no shorthand syntax such as XPath for matching patterns in the input XML; instead, one has to match much more verbose SPARQL graph patterns against the RDF graph representing the DOM.

Studying SparqPlug and VAN DEURSEN's approach helps to realize the benefit of combining queries against RDF and XML. XSPARQL [AKK+o8] does that by incorporating the complete SPARQL language into XQuery, thus entirely avoiding the necessity of first converting from one representation into the other. The SPARQL parts of an XSPARQL query are rewritten into XQuery.



In absence of nested queries in SPARQL,<sup>6</sup> the SparqPlug translation approach cannot fully be reproduced in XSPARQL, but VAN DEURSEN's can. As with XQuery, but in contrast to XSLT, translating complete documents to RDF using XSPARQL requires explicit recursion. Moreover, any maintenance information passed from the XQuery function that processes parent elements to another function that processes child elements, such as the URI of the resource created from the parent, has to be passed as explicit parameters, whereas XSLT 2 supports hidden tunnel parameters.

### 8.1.5 Conclusion and Future Work

The Krextor framework presented in this section comprises translations from the OMDoc and OpenMath CD semantic markup languages for mathematical knowledge to RDF. That makes knowledge represented in these language comprehensible to services for metadata editing, metadata and link validation, linked data browsing, and RDF-based query answering. A translation of OWL ontologies from their OMDoc implementation to the widely supported RDF serialization enables reasoning with ontologies formalized and documented in OMDoc. Krextor supports a great variety of RDF output serializations, from the standard RDF/XML to RDFa generation embedded into a publication process and callbacks to Java applications that integrate Krextor.

Three extensions particularly of the OWL/OMDoc→OWL/RDF translation may be desirable in future. Firstly, the translation currently ignores any knowledge that is not expressible in OWL, such as documentation beyond entity and axiom annotations, or axioms formalized in different logics. Consider, however, a round-trip editing workflow, which combines Krextor and the reverse translation introduced below in [section 8.2.2](#), so that ontology engineers can edit the OWL parts of an ontology in OWL tools such as Protégé, whereas they maintain the documentation and enhanced formalization of the same ontology in an OMDoc environment. That would require the translation to preserve all non-OWL constructs in a format that OWL tools ignore but preserve, e.g. as a special kind of annotations. A more ambitious translation would include axioms that have not, syntactically, been formalized in OWL, but that would be expressible in OWL via translation from their original logic.<sup>7</sup> Such translations can be modeled in a logical framework; in fact, implementations of the OWL→FOL direction already exist in OMDoc (cf. [KMR]) and Hets [Mos; MML07]. Finally, serializing OWL as OWL XML [MPPS09] or as functional-style text [MPSP09] instead of RDF would be more natural in that, e.g.,  $n$ -ary structures would not have to be emulated by RDF collections, and axiom annotations work without reifying the axioms.<sup>8</sup> An OWL XML or text “output module” – where the functional-style text can be obtained from the XML serialization – does not directly fit into Krextor's current RDF-centric architecture. However, the OWL/OMDoc extraction module could be modified to call OWL-specific utility templates and functions instead of the RDF-specific ones, which would, depending on whether OWL or RDF output is requested, generate an OWL serialization or call the existing RDF output routine.

Due to its modularity and extensibility, Krextor is not limited to the languages mentioned so far. I have demonstrated that Krextor's rich supply of utility templates and functions supports the implementation of frequently occurring structural mappings with little code, while the full expressive power of XSLT is still available for hard cases. Another example, covering the hCalendar

<sup>6</sup>SPARQL 1.1 will support them [HS10b].

<sup>7</sup>For example, the FOL axiom  $\forall x. C(x) \rightarrow D(x)$  can be conceived as equivalent to the DL axiom  $C \sqsubseteq D$ .

<sup>8</sup>For a more detailed comparison of OWL XML to RDF/XML, see [LZ10].



microformat, can be found in [Lan09a]. From a general semantic web point of view, Krextor enables ontology engineers to develop an XML syntax that is convenient for human editors and amenable to XML validation – compare, e.g., the direct XML serialization of OWL [MPPS09] to its RDF/XML serialization –, without worrying about the compatibility to RDF. Moreover, the RDFa extraction routines are ready for reuse by extraction modules specific to any language that hosts RDFa; in fact, MARÍA JOSÉ IBÁÑEZ et al. have reused them for translating semantic annotations of business processes represented in BPEL (Business Process Execution Language) [JIVE+10].

The SWiM semantic wiki and the TNTBase XML database integrate Krextor for translating mathematical markup to RDF. Section 9.3.2.1 covers the integration into SWiM. The Krextor plugin for TNTBase [Zho+], which repository maintainers can configure to be executed in the post-commit hook for OMDoc documents (cf. section 8.4), translates OMDoc to RXR and stores the latter as XML documents inside TNTBase. The publication of OMDoc documents as XHTML+RDFa has been realized in this setup, using JOMDoc and Krextor’s RDFa output utility module, as explained in appendix C.1.2.2 and depicted in figure 6.9 on page 221. In the same setting, also shown in figure 6.9, TNTBase employs a variant of the Krextor plugin, which translates OMDoc to N-Triples fed into a Virtuoso triple store using Mocassin’s Virtuoso data access object.<sup>9</sup> From that triple store, users or external services can query the RDF extracted from OMDoc in a controlled way using the Mocassin query input form introduced in section 6.5.2.2, or freely via Virtuoso’s SPARQL endpoint.

These setups prove Krextor’s suitability for integration in different settings. A systematic evaluation of the performance of RDF extraction is pending. Validating the claim of easy extensibility, particularly w.r.t. new extraction modules for semantic markup languages, would require another evaluation. The “cognitive dimensions of notations” methodology for evaluating programming languages, which has been applied to the Semantic Pipes language for defining composable RDF→RDF translations (cf. [LPPH+09]), is likely to yield useful results for Krextor as well.

A final direction of future work concerns maintaining extraction rules from a semantic markup language to RDF as annotations to its XML schema (cf. section 2.3.2.1). That would unify two tasks that belong together but have been separated so far in practice: specifying the syntax and the semantics of a semantic markup language.

## 8.2 Migration to More Expressive Languages

This thesis advocates representing mathematical knowledge in OMDoc, with embedded OpenMath, MathML, and RDF(a), as this combination covers all structural dimensions of mathematical knowledge in flexible degrees of formality (cf. section 2.5). Nevertheless, there may be good reasons for continuing to maintain knowledge collections in less expressive languages. In most cases, the reason is better tool support for specific tasks. Consider, for example, validation: Some of the validation steps outlined in section 6.3.2, such as type or proof checking, enjoy little support by tools that natively understand OMDoc. Or consider specialized editors, such as Protégé for

<sup>9</sup>Figure 6.9 merges both ways of storing RDF into one component. The two alternatives merely exist for historical reasons. With future support for the SPARQL Query Results XML format by the Krextor RDFa output utility module, as mentioned in section 8.1.1.2, both services will be able to access the triple store.

OWL ontologies. In other cases, users are more familiar with the terminology and structure of legacy languages, as discussed in [section 3.2.3.1](#) for the case of OpenMath CDs.

Translating knowledge represented in legacy languages to OMDoc paves the way for further formalization and documentation and for applying services that require an OMDoc representation (e.g. publication as interactive web documents, as discussed in [chapter 7](#)) or a *different* non-OMDoc representation that can be obtained from OMDoc. Translating existing knowledge collections to OMDoc also enables bootstrapping an OMDoc formalization of new fields. For example, algebra enjoys a wider coverage by OpenMath CDs than by OMDoc theories, and theoretical physics is better covered by OWL ontologies (cf. [section 3.5.2](#)).

[Section 8.2.1](#) briefly summarizes existing approaches to translating legacy languages to OMDoc. [Section 8.2.2](#) discusses the particular case of translating OWL ontologies from RDF to an OMDoc representation, which complements the introduction of the inverse OMDoc→RDF translation in [section 8.1.3](#).

### 8.2.1 Translating Less Expressive Languages to OMDoc (State of the Art)

Translating mathematical knowledge from a non-OMDoc representation to OMDoc involves a translation of the constructs of the respective source language in any case. In the case of a formal representation, it also requires OMDoc implementations of the meta-theories for the logics chosen for formalization. [Chapter 4](#) demonstrates that for OWL. Further logics have been implemented in OMDoc in the course of the LATIN project [[KMR](#)] and in older efforts of translating individual representation formats to OMDoc (cf. [section 2.4.4.1](#)).

Previous translation efforts have applied different ways of translating constructs of non-OMDoc languages. XML languages can be translated to OMDoc on the XML→XML level, e.g. by an XSLT implementation. There is, for example, an (incomplete) XSLT stylesheet for translating OpenMath CDs to OMDoc theories (cf. [section 3.2.3.1](#)). A common approach to translating a non-XML language to OMDoc is hooking into an existing parser for it and making it emit OMDoc XML markup. This has, for example, been done for the Twelf language (cf. [[KMR](#)]). In cases where this has succeeded so far, OMDoc's expressivity – most importantly in the logical/functional dimension – has subsumed the expressivity of the source language from the outset, or it has been extended to do so. The latter has, for example, been done by introducing OpenMath-compatible CD metadata into OMDoc 1.2 (cf. [section 3.4.5](#) and [[Koho6b](#), appendix A.1]).

With the RDFa extension of OMDoc presented in [chapter 5](#), future translations from legacy languages to OMDoc are more likely to be realizable without extending OMDoc's XML schema. For example, the CD metadata of an OMDoc theory could now also be represented as RDFa annotations, reusing the OpenMath CD ontology. On the level of mathematical objects, such an extension path has already been available for a long time, by way of OpenMath attributions. The easy ability to extend OMDoc's expressivity via annotations does, of course, not preclude the promotion of new, useful annotations to proper OMDoc syntax by extending the XML schema, as discussed in [section 5.4](#).

### 8.2.2 Translating OWL and Other Ontologies to OMDoc

In order to bootstrap the formalization and documentation of existing OWL (and, equally, RDFS) ontologies in OMDoc, and in order to benefit from the good existing tool support for creating OWL ontologies, we have implemented a translation of OWL ontologies – OWL 1 for now – from RDF to OMDoc [Kuro9]. [Appendix C.3.2](#) summarizes the key features of this implementation.

While it was feasible to implement this translation on the RDF→XML level, a direct OWL→OMDoc implementation would operate on a higher, possibly more adequate level of abstraction. Such an implementation could be realized by parsing the input using the OWL API [Owlb] – which supports additional OWL serializations beyond RDF – and creating the OMDoc output using the JOMDoc library [Jom]. DIMITAR MIŠEV has demonstrated JOMDoc’s suitability for creating OMDoc implementations of ontologies by implementing a translation between SUO-KIF and OMDoc. There, the Java libraries of the SigmaKEE development environment for SUO-KIF and SUMO are responsible for parsing and serializing SUO-KIF, and JOMDoc is responsible for parsing and serializing OMDoc [Mis10]. A particular challenge in the application of that translator to the SUMO ontology were circular dependencies among different modules, such as, in the simplest case, module  $M_1$  using a concept from module  $M_2$  and vice versa. SUO-KIF does not have any notion of imports. OWL has, but it does not require importing external ontologies before using concepts from them. OMDoc does, and it does not allow circular imports. The above-mentioned OWL→OMDoc translation ignores such problems, but the SUO-KIF→OMDoc translation addresses them by identifying import cycles and eliminating them by refactoring theories. This algorithm has been implemented for general OMDoc theories as a part of the JOMDoc library and could thus also be reused by a reimplemented OWL→OMDoc translation.

Finally, in the discussion of the OMDoc→OWL translation in [section 8.1.5](#), the necessity of preserving those OMDoc constructs that OWL does not support as special annotations in order to enable round-trip editing of an ontology both with OWL and with OMDoc tools has been emphasized. In such a scenario, an OWL→OMDoc translation would have to turn such special annotations found in its OWL input back into OMDoc structures.

## 8.3 Coping with Different Representation Granularities on Import and Export

This section discusses translations that enable the integration of services that expect entirely different granularities of knowledge representation – large files vs. small knowledge units, and metadata embedded into semantic markup vs. metadata as RDF triples.

Most traditional collections of mathematical knowledge have begun their history as collections of files in a file system or in a repository with a file-based interface. For example, the OpenMath CDs are maintained in a central Subversion repository, and the maintenance of the MML is being transferred into a distributed version control system (cf. [section 1.4.2](#)). But even a specialized knowledge base might run on top of a file system storage backend. Moreover, despite the presence of web interfaces to knowledge bases and the services they integrate, the file system remains an important “lowest common denominator” interface to legacy tools such as general-purpose

text editors that give experienced users full access to the raw representation of mathematical documents.

[Section 8.3.1](#) briefly summarizes the challenges that file-sized units of knowledge pose to semantic services and point out the need for fine-grained access to resources. [Section 8.3.2](#) presents a way of physically splitting representations of mathematical knowledge from the coarse granularity of files to smaller fragments upon import into a knowledge base, so that services can immediately utilize them in a granularity they can cope with. [Section 8.3.3](#) discusses an approach for preparing metadata of documents in such a way that they become accessible to RDF-based services such as query engines or editing forms, while at the same time remaining editable as parts of their documents. Both transformations have to be undone when exporting knowledge back to the file system; this is considered as well. [Section 8.3.4](#) briefly discusses the potential of XML databases with fine-grained fragments access to these procedures.

### 8.3.1 Challenges that Complex Files Pose to Semantic Services

For a system that fully understands the semantic structures of mathematical knowledge, files may be too large a unit of knowledge management, as one file typically contains many semantically relevant resources. Semantic services often have a preferred granularity at which they can optimally utilize mathematical knowledge; for example:

**Editing** semantic representations does not always scale to large units with substructures. For example, an editing form for metadata is easiest to realize if all of its entries are simple key/value pairs, where values are literals or URIs pointing to resources, but not nested resources that the user expects to be editable in place.

**Semantic Validation**, as discussed in [section 6.3](#), may be expensive; its performance can be improved by only applying it to the logical unit that has actually been edited – for example a single symbol definition within an OpenMath CD file that contains multiple of them.

**Linked Data Browsers** often display one resource and its links to other resources, allowing the user to traverse them (cf. [section 6.4.1.2](#)). Often, such interfaces cannot cope with subparts of resources, e.g. symbols in a CD, being resources of their own, except for allowing the user to explicitly traverse the “has part” links leading to them.

**Semantic Information Retrieval** is not concerned with finding files that contain lines matching a given text, but with finding resources that have certain properties or links to other resources (cf. [section 6.5.2](#)).

**Discussions** about resources in a semantic repository cannot be supported effectively by services unless they are explicitly linked to the resource(s) they deal with. For example, the problem solving assistant presented in [section 6.6.3](#) requires at least discussion forums on the level of mathematical statements.

### 8.3.2 Splitting and Reassembling Files on Import and Export

The algorithm specified in this section originates from a setting where multiple services had been integrated that required access to mathematical knowledge on the level of logical/functional

statements, whereas the original file-based storage had to be retained as an alternative access. The storage backend underlying the semantic knowledge base did not support fine-grained fragment access, which precluded storing one imported file as one unit in the knowledge base. These differences have been accommodated for by physically splitting the files into statement-level fragments on importing them into the knowledge base, and reassembling them on export.

### 8.3.2.1 A Generic Import and Export Algorithm

Let  $D_0(V, \rightarrow)$  be an XML document (cf. [section 2.3.2.1](#)) with node set  $V$  and parent $\rightarrow$ child relation  $\rightarrow \subseteq V \times V$ , and root node<sup>10</sup>  $r =: r_{D_0} \in V$ . Let  $s: V \rightarrow \mathbb{B}$  be a boolean predicate that is satisfied for a node  $v \in V$  iff it forms the root of a subtree that contains a knowledge unit that should be factored into a separate unit when *importing*  $D_0$  into our knowledge base. Let  $t(v) := \{w \in V \mid v \rightarrow^* w\}$  be the subtree starting at root  $v$ , where  $\rightarrow^*$  is the reflexive and transitive closure of the parent $\rightarrow$ child relation  $\rightarrow$ . On import, for each node  $v$  satisfying  $s(v)$ , the respective knowledge (sub)unit **MUST** be added to the knowledge base as a new document tree  $D_v := t(v)$  with root  $v$ .<sup>11</sup>  $D_v$  **SHALL** have a unique name  $\text{id}(D_v)$ , which **MAY** be derived from  $\text{id}(D_0)$  and the fragment identifier  $\text{id}(t)$  of the subtree via some function  $\text{id}(D_v) := f(\text{id}(D_0), \text{id}(t))$ . This splitting process **SHALL** be recursively applied to  $D_v$ . Instead of the original document  $D_0$ , a document  $D'$  **SHALL** be added to the knowledge base, in which, for each root  $v$  of a knowledge unit that is now represented by its own document  $D_v$ , the subtree  $t(v)$  is replaced by a new node  $i(D_v)$  that *references*  $D_v$  for the purpose of inclusion. The concrete syntax of  $i$  is determined by the inclusion facilities of the respective representation language; if inclusion is not natively supported, `XInclude` [[MOV06](#)] **SHOULD** be used. For any two documents  $D_1$  and  $D_2$ , I will henceforth write  $D_1 \rightarrow_i D_2$  if  $D_1$  includes  $D_2$ , i.e. if  $D_1$  contains a node  $i(D_2)$ .

When *exporting* a document  $D$  from the knowledge base is requested, the application **MUST** first determine if its root node  $r_D$  is an admissible root element of a self-contained file. For this check, another predicate  $e: V \rightarrow \mathbb{B}$  is **REQUIRED**. If  $e(r_D)$  is satisfied, then  $D$  **SHALL** be exported as a file, otherwise the *nearest exportable parent document*  $D_p(D)$  **SHALL** be exported, i.e.  $D_p(D) := \min_j \{D' \mid D' \rightarrow_i^j D \wedge e(r_{D_p(D)})\}$ . The export function **MUST** replace any inclusion reference  $i(D_s)$ , pointing to a subunit stored in a document  $D_s$  in the knowledge base, by the tree  $D_s$ , i.e. it **MUST** resolve all inclusions.

### 8.3.2.2 Specializing the Algorithm to a Representation Language

Specializations of the algorithm introduced in the previous section to OMDoc documents and OpenMath CDs have been implemented. This section summarizes the features of these languages that are relevant for the algorithm; [appendix C.3.3.1](#) describes the technical details of the implementations.

<sup>10</sup>For simplicity I assume that the root node is an element, whereas in XML the actual root node of a document is the *parent* node of the topmost element node.

<sup>11</sup>It may be necessary to add some additional maintenance information to  $D_v$  to facilitate working with that knowledge unit while it is inside the knowledge base, but I disregard this peculiarity here.

OMDoc can natively represent a document inclusion  $i_{\text{OMDoc}}(D)$  as `<ref type="include" xref="id(D)"/>`<sup>12</sup>; on the level of mathematical objects, `<OMR href="id(D)"/>` works in OpenMath, and `<share href="id(D)"/>` in Content MathML. Besides complete documents, theories, and non-constitutive statements – if referring to their home theory – can constitute roots of an exported file, provided that they are wrapped into an *omdoc* element. A fragment of an OMDoc document is identified via the *@xml:id* attribute; logical/functional fragments can also be identified by *@name* attributes, where the name of a symbol is merely unique within a theory.

The OpenMath CD language does not support an inclusion mechanism above the object level, which requires representing  $i_{\text{OpenMath}}(D)$  as `<xi:include href="id(D)"/>`, using XInclude. Only whole CDs (having a *CD* root element) are admissible for export. Signature dictionaries and notation dictionaries are treated analogously. CDs, symbol and notation definitions, and symbol type signatures are the only fragments that have identifiers, represented by the *CDName* and *Name* child elements and *@name* attributes, respectively.

### 8.3.3 Making Metadata Accessible for Different Services and Editors

Translating mathematical knowledge originally represented in XML to RDF makes it accessible to services such as validation (cf. [section 6.3.3](#)), linked data publication (cf. [section 6.4.1](#)), and information retrieval (cf. [section 6.5.2](#)). This affects the primary mathematical knowledge (the “data”) as much as the *metadata*. However, an XML representation of the same metadata is still required, not only for exporting both data and metadata in a file, but also for publishing. The XSLT stylesheets for publishing OMDoc and OpenMath CDs, for example, have special support for the metadata vocabularies natively supported by OMDoc 1.2 and by the OpenMath CD language.

[Section 6.2.6](#) has introduced two alternative interfaces for editing metadata – forms and a document-like interface –, and reported a case where users requested both. A form can easily be populated from an RDF triple store, whereas a document interface would rather give access to the XML representation of the respective knowledge item. Making two representations of the same metadata editable via two separate interfaces causes synchronization problems, unless every change made to one representation is immediately applied to the other one. As the latter would require additional time and space, I have developed an alternative approach where metadata are exclusively stored in the RDF triple store but made accessible to an document-oriented services on demand and inserted back into the documents when exporting them to the filesystem. [Appendix C.3.3.2](#) describes how that has been implemented for OpenMath CDs in the SWiM semantic wiki (cf. [chapter 9](#)).

### 8.3.4 Related Work

#### 8.3.4.1 XML Databases with Fine-grained Fragment Access

Splitting fragments off into new physical documents can be avoided when running an XML database that supports fine-grained fragment access – as, e.g., TNTBase does with its XQuery support (cf. [section 6.5.2.1](#)). As TNTBase combines a file-based Subversion repository [[Apaa](#)] with

<sup>12</sup>This neglects the case when authors want such inclusions to be preserved even on export. This is, however, possible by choosing a custom *ref* type different from “include”, or any other means of application-specific annotation.



an XML database, users and services can not only access documents as files, but they can also access arbitrary fragments of them. With such a database, the problems discussed in sections 8.3.1 and 8.3.2 reduce to making fragments that represent relevant units of knowledge accessible to services. To that end, it may still be necessary to assign short, meaningful URIs to such fragments – i.e. URIs that do not contain a lengthy XQuery parameter –, as, e.g., suggested by the implementation described in appendix C.3.3.1. If documents have been split into smaller fragments for other reasons, such as performance or reusability, the virtual document mechanism of TNTBase also offers rich, user accessible functionality for creating parameterized views powered by XQuery that, e.g., compose multiple fragments into one coherent XML document [ZK10]. An advanced application of that mechanism could be used for making self-contained knowledge collections accessible to external services – for example wrapping a proof and all of its dependencies, such as all symbols, axioms, and theorems that it requires, into a self-contained OMDoc document – or a translation into a more suitable language – for verification by an external proof checker.

#### 8.3.4.2 Metadata Editing Forms on Top of Semantic Markup

For the metadata editing scenario outlined in section 8.3.3, I have focused on RDF-based editing forms. Forms have, however, also been realized on top of semantic markup, for example in the case of the Semantic Forms extensions to KiWi and Semantic MediaWiki mentioned in section 6.2.1.4. There, the semantic markup defines the order and labels of metadata fields, whereas they store the *values* in an RDF triple store. In contrast to the form- and document-based metadata editing interfaces discussed here, they do, however, not support adding new metadata fields without modifying the template of the form.

### 8.4 Recommendations for Running Translations Transparently

This section gives brief recommendations on how to use translations when developing knowledge base systems. A system **SHOULD** execute translations transparently, so as to relieve users from remembering what service requires what knowledge representation and initiating translations manually, but the integrating system is free to choose how to achieve that. A system **MAY** (re)generate other representations of the knowledge that has just been edited or imported as soon as possible, but it **MAY** also produce them on demand, e.g. whenever running a service that needs them. For example, the TNTBase XML database allows for configuring translation plugins, such as Krextor, to be applied in the post-commit hook (cf. [PCSFo8, chapter 5]) of its Subversion component [Zho+; ZKR10]. A knowledge base **SHOULD** not only expose its contents in its “primary” representation but also in any other representation that it is capable of generating – these might be useful for external users or services. Finally, despite all transparency, a user or service **SHOULD** also be able to explicitly request a knowledge item in a specific representation. Such a functionality **MAY** be realized using HTTP content negotiation (cf. sections 6.4.1.3 and 7.5.1.1).



## 8.5 Conclusion

This chapter has introduced translations between different ways of representing mathematical knowledge. Such translations help to make mathematical knowledge reusable and comprehensible across different knowledge bases. They also make it accessible to a large number of services that have been integrated on top of the same knowledge base, as they feed the knowledge to them in a granularity or language they can handle best. Finally, they enable specialized knowledge bases to communicate with legacy file system repositories. In whatever way it is applied, the translation approach presented in this chapter finally enables the combination of heterogeneous services in an integrated collaboration environment, such as the one presented in the following chapter.

## Acknowledgments

The description of the Krextor XML→RDF translation framework in [section 8.1](#) and the overview of related work is partly based on an earlier publication [[Lan09a](#)]. SIARHEI KURYLA has contributed the extraction rules for OMDoc's syntactic sugar for representing OWL to the OMDoc→OWL extraction module described in [section 8.1.3](#); he has also implemented the OWL→OMDoc translation introduced in [section 8.2.2](#) under joint supervision of FLORIAN RABE and me [[Kuro9](#)]. Under the same joint supervision, DIMITAR MIŠEV has designed and implemented the SUO-KIF→OMDoc translation discussed in [section 8.2.2](#). The reference introduction and contraction facility of the JOMDoc library has been implemented by DIMITAR MIŠEV under supervision mainly of CHRISTINE MÜLLER.



# The Semantic Wiki SWiM – An Integrated Collaboration Environment

This chapter introduces the SWiM Semantic Wiki for Mathematical Knowledge Management, a prototype developed for the purpose of evaluating the feasibility of effectively integrating heterogeneous services for producing (creating, formalizing, organizing) as well as consuming knowledge in a coherent work environment. After a review of the state of the art of wikis in [section 9.1](#), [section 9.2](#) discusses the requirements for SWiM, given the collaborative workflows introduced in [section 6.1.2](#) it is particularly intended to support. [Section 9.3](#) explains how SWiM integrates the primitive services introduced in [chapter 6](#), how it employs the translations introduced in [chapter 8](#), and what functionality it offers beyond that in order to support the given workflows. [Section 9.4](#) walks through SWiM's support for the three OpenMath CD maintenance workflows. [Section 9.5](#) reviews related work, covering not just wikis, but generally environments for collaborating on mathematical knowledge. [Section 9.6](#) discusses how well SWiM serves its intended purpose from an engineering perspective, whereas [chapter 10](#) covers the usability perspective.

## 9.1 Wikis and Semantic Wikis (State of the Art)

SWiM integrates mathematical services in a wiki environment. Wikis as lightweight web 2.0 CMS have briefly been introduced in [section 1.3.2](#). This section reviews the state of the art of wikis, insofar as it is relevant for this thesis.

### 9.1.1 Elementary Wiki Characteristics

To start with, I clarify the two central terms “wiki” and “page” [[Lano6b](#)]:

**Wiki** denotes wiki *sites* as well as the CMS-like software used to manage them, then also called wiki *engines* or wiki *systems*. This thesis speaks of a *wiki* wherever the meaning is clear from the context.

**Pages:** This chapter uses the general term *page* when speaking from the general perspective of the user interface or database of a wiki. When wiki pages hold mathematical knowledge items, I name them accordingly. Wiki-based encyclopedias, such as Wikipedia, mostly use the term *article* for a page. Not only when building an encyclopedia, it is encouraged that a wiki page covers a distinct topic, or a set of closely related topics, so that other pages can more easily link to it. Therefore, some wikis use the term *topic* for a page. Some semantic wikis (cf. [section 9.1.2](#)) speak of *concepts*, if pages describe real-world concepts.

In a wiki, one usually creates a new page by entering its desired URL, or, preferably, by linking from an existing page to the page to be created. In both cases, the wiki offers an editing form for the new page. Usually, anyone is allowed to edit pages, but access can as well be restricted to a work group – in a corporate intranet, for example. “Wikis are generally designed with the philosophy of making it easy to correct mistakes, rather than making it difficult to make them.” [Wikiod] Therefore, they permanently store old page versions and offer facilities to display differences between two versions and to restore a certain version. Other functionality typically offered by wikis includes notification about recent changes, full-text search and, in most cases, a simple kind of user account management. The main characteristic features of wikis are openness, simplicity, as well as – thanks to hyperlinking – their incremental and organic structure (see [Cun+] for more). Considering open wiki communities in special, up-to-dateness, principles of grassroots democracy – for example, when discussing about the bias of some page – and the motive of learning from each other come along<sup>1</sup>. Key aspects of wikis are also more and more being shared by systems that are not called “wikis”; consider the Google Wave technology [Goo].

Application areas of wikis range from content management to e-learning to groupware to collaborative corporate and personal knowledge management [Lano6b; EGHo8]. Wikis have become established chiefly through their use for public and open community projects. The first public wiki, a repository for know-how on design patterns and extreme programming [C2p], had grown to more than 30,000 pages by 2004. The biggest and most well-known wiki is the free encyclopedia Wikipedia [Wik].

### 9.1.2 Semantic Wikis

Semantic wikis combine the above-mentioned wiki characteristics with the Semantic Web vision (cf. [section 1.3.3](#)) in diverse ways, with two main strands [SVo8]:

**“Semantics for Wikis”:** Semantic web technologies are used to enhance content presentation, navigation, personalization, social networking, and data exchange in wikis.

**“Wikis for Semantics”:** Conversely, semantic wikis are used as lightweight collaborative knowledge formalization environments or ontology editors.

Besides knowledge formalization and ontology editing and the above-mentioned application areas of non-semantic wikis, semantic wikis have also been used for lightweight prototyping of semantic web applications [BDH+09]; the discussion in [section 9.6.3](#) explains why my SWiM wiki has had a similar role.

<sup>1</sup> A survey of many well-known wiki communities and their characteristics can be found in [Lano6b, chapter 4].

An overview of many semantic wiki engines is given in [BGE+08]; a few of them are mentioned in the following. In the early age of semantic wikis (2005/06), a lot of new features were prototyped by developing new engines from scratch, whereas, nowadays, a few engines have stabilized and been extended by third-party plugins. Semantic MediaWiki (SMW [Sema]), an extension of the non-semantic MediaWiki engine, is the most widely used and most stable semantic wiki engine. IkeWiki (cf. section 9.3.1), the system on which SWiM is based, and its successor KiWi (cf. section 9.6.4.4) stand out by their strong XML and RDF infrastructure. KiWi is positioned as an operating system like platform for semantic social software, the wiki just being one out of many plugins.

Since about 2008, semantic wikis have become more mainstream, which is witnessed by commercial distributions and the incorporation of semantic wiki functionality into the core of formerly non-semantic wikis. SMW+, a commercial distribution of SMW, is mentioned in section 1.3.4. zAgile Wikidsmart [zAg] is a similar semantic extension of the non-semantic Confluence wiki engine, which is popular in corporate intranets. Tiki Wiki [Tik], marketed as “*the most feature-complete CMS*”, is an example of a traditional wiki that has more recently been extended by the semantic concept of typed links.

Shallowly annotated text prevails in some semantic wikis, whereas in others, formal knowledge prevails and unstructured text only appears in comments or labels that describe formal concepts [ODM+06; BGE+08]. Yet others mix annotated text and highly formalized problem-solving knowledge [BP08a]. The most common approach is, however, to represent knowledge about one subject of interest – a “knowledge item”, in the terminology of this article, – by one wiki page and to annotate pages and links between pages with types defined in an ontology. In this kind of semantic wikis it is advisable to keep pages small and refactor them if they tend to describe more than one knowledge item. Some semantic wikis escape the “page = resource” restriction: KiWi supports tagging “fragments”<sup>2</sup>, which can span arbitrary ranges of a page [BEK+09]. The Semantic Internal Objects extension [Kor10] for SMW allows for embedding additional resources and their properties into a page, like blank nodes. Ultimately, the PAUX system<sup>3</sup> [PAU] even treats sentences and words as resources, which can be linked, annotated, searched, and reused. These fine-grained structures can be exported from the internal relational database to RDF linked data.

Most semantic wikis represent the graph of typed pages and links in RDF. Existing ontologies, such as FOAF (cf. section 3.5.1), are either preloaded into the wiki or imported later; some semantic wikis support collaborative construction of new ontologies. SMW, for example, prefers the latter approach, where an ontology is implicitly extended whenever a page or a link is assigned a new type [Sema; VKV+06].

### 9.1.3 Mathematical Services in [Semantic] Wikis

This section briefly reviews to what extent existing wikis, semantic or not, support the primitive services that are relevant for collaboration on mathematical knowledge and have been discussed in chapter 6.

<sup>2</sup>not to be confused with XML fragments; the KiWi fragments are more flexible.

<sup>3</sup>PAUX is not a wiki in the traditional sense, as its authoring environment is a rich client separate from the browsable view on the content.

**Editing:** In non-semantic wikis as well as in state-of-the-art semantic wikis, support for mathematical knowledge is scarce. With a few exceptions, discussed in [section 9.5](#), none of these systems support a semantic representation of mathematical knowledge. Most wiki engines support editing presentation-oriented mathematical formulæ, either natively or via a plugin, usually using  $\LaTeX$  syntax (cf. [section 2.4.7](#)). As the rendering engines used for publishing such formulæ only support a restricted vocabulary of  $\LaTeX$  commands (see below), it is not possible to use semantic macros such as those of  $\S\TeX$  (cf. [section 2.4.7.2](#)). Other than generic support for RDF or ontologies, which would allow for using the ontologies for mathematical structures presented in [section 2.4.10](#) and [chapter 3](#), contemporary semantic wikis do not support a semantic representation of mathematical knowledge.

**Publishing:** Mathematics-specific publishing support in contemporary wikis is limited to presentation-oriented formulæ. Some wikis, or plugins available for them, use  $\LaTeX$  as a familiar input syntax for a non- $\LaTeX$  renderer, such as jsMath [[Cer](#)]. Other wikis, including MediaWiki, depend on an actual  $\LaTeX$  installation but only support a restricted repertoire of macros.

**Information Retrieval** in most non-semantic wikis is limited to full-text search. Semantic wikis usually offer support for SPARQL, a subset thereof, or a similar query language, and allow for embedding *inline queries* into the content of a page, where they are evaluated on rendering [[KSV07](#)]. The query language of SMW resembles the text syntax of MediaWiki. The syntax of KiWi’s query language KWQL [[BW10](#)] resembles the one of full-text search engines such as Lucene [[Apac](#)]. KWQL supports querying wiki pages (“content items”) and annotated fragments within pages by a restricted set of metadata, arbitrary author-defined tags, link targets, and full text. However, queries that involve RDF types or properties are not yet supported.

**Argumentation** has enjoyed deeper coverage by previous research on [semantic] wikis and is therefore covered separately in the following section.

The only known case where a semantic wiki in the narrower sense<sup>4</sup> has been applied in a mathematical setting so far is SlugMath [[Weia](#)], a companion site to mathematics courses taught at the University of California, Santa Cruz. SlugMath is powered by SMW [[Sema](#)]. It has its own ontology, with structural concepts of mathematical knowledge such as definitions, other statements, lexemes<sup>5</sup>, and structures<sup>6</sup>, and educational concepts such as skills, which are, however, not yet linked to mathematical concepts. Creating instances of this ontology is supported via the semantic forms (cf. [section 6.2.1.4](#)). The semantic structures – coarse-grained in comparison to the ontologies introduced in [chapter 3](#) – are used for information retrieval via inline queries. Mathematical objects are still purely presentational.

---

<sup>4</sup>i.e. not counting sites such as Connexions, as argued in [section 1.4.2](#)

<sup>5</sup>natural language terms that represent, in the terminology of this thesis, references to symbols

<sup>6</sup>concrete mathematical objects, e.g. “the group with two elements”

### 9.1.4 Argumentative Discussions

Even though existing wikis offer little technical support for reporting problems with knowledge items and discussing possible solutions, as detailed in [section 9.1.4.1](#), conventions for arguing about problems have emerged in communities such as Wikipedia ([section 9.1.4.2](#)) and have been conceptualized and formalized into argumentation models ([section 9.1.4.3](#)).

#### 9.1.4.1 Lack of Technical Support

Widespread shortcomings of wikis w.r.t. argumentative discussions include (i) poor support for pointing out what exactly is wrong with what [part of a] wiki page, i.e. lack of a problem vocabulary and, in non-semantic wikis, a limited structure of pages, (ii) lack of threaded discussions (in many wikis), and (iii) lack of a well-defined flow of argumentation.

Even many non-semantic wikis allow for tagging knowledge items, e.g. as “needs improvement”. Complementarily, collaborators can insert warning messages directly into a [section of a] page affected by an issue. Some systems, such as MediaWiki [[Med](#)], offer macro-like mechanisms for creating pre-defined building blocks for such warning messages, which can then be included into pages. In semantic wikis, tags can be made more precise by defining their meanings in an ontology, but there are hardly any ontologies in use that describe types of problems. Attempts to distill such ontologies from analyzing the practice of using Wikipedia are reviewed in [section 9.1.4.3](#).

Commenting on knowledge items, e.g. for discussing a problem that a collaborator has pointed out, is possible in almost every wiki. Most commonly, a separate discussion page is associated with the primary wiki page, serving as a local discussion forum about one subject of interest. In most wikis, discussions are not threaded. MediaWiki’s discussion pages, for example, are even completely unstructured, except that the good practice of creating one section per “thread” is supported, but not enforced, by a button that adds a new section [[Med](#)].<sup>7</sup> The only systems that represent the semantic structure of threads and link posts to the user profiles of their authors in a machine-comprehensible way, using the SIOC ontology (cf. [section 3.5.1](#)), are IkeWiki<sup>8</sup> and its successor KiWi. Making every post a distinct RDF resource and preserving the threaded structure of a discussion is a prerequisite for adding an argumentation ontology layer, as explained in [section 3.6.1](#).

The argumentation itself proceeds without technical support, the DILIGENT-powered research prototypes coefficientMakna and Cicero being exceptions discussed in [section 9.5.7](#). The community is left alone with devising reasonable issue warning messages and establishing a workflow of reporting, discussing, and solving issues and documenting the solutions. This is mostly done by jointly agreeing on best practices in conflict resolution and authoring, and making them official policies for the community, as has been investigated in the case of Wikipedia [[KSP+07](#)].

#### 9.1.4.2 Conventions for Argumentation in Wikipedia

As a concrete example for arguing about a problem without technical support, consider the MediaWiki-driven Wikipedia. One can consider a Wikipedia article a result of conceptualization

<sup>7</sup>The more recent LiquidThreads extension [[MG10](#)] enables threaded discussions in MediaWiki.

<sup>8</sup>I have contributed support for the argumentation ontology presented in [section 3.6.1](#) to IkeWiki.



and “formalization” (here rather: structured presentation according to certain conventions), of which only the formalization is done inside Wikipedia. The concept has already existed before and must be widely agreed upon [Wik09f]. Therefore, Wikipedia’s policy demands that only the “formalization” be discussed on the article’s discussion page (officially called “talk page” in the English Wikipedia).

Suppose an article violates the fundamental principle of a neutral point of view (NPOV [Wik09e]<sup>9</sup>). Any author who is concerned about this can tag the article by inserting the building block “POV” (neutrality [Wik10g]). It is then recommended to justify why the neutrality of the article is debated by adding a respective section to the discussion page of the article. Within that section, the general conventions for discussions pages apply [Wik09g]: The author has to make clear what section of the article his discussion post applies to<sup>10</sup>, he has to verbalize his report in a comprehensible way, and finally, lacking the forum infrastructure that other wikis offer, he has to append his signature (a link to his user profile with a timestamp). An author who wants to discuss an existing issue has to look up the corresponding section on the discussion page and then indent his reply by one more level than the post he is replying to.

Solutions to issues would be proposed in natural language only, and if users come to vote on proposals, they would do it in an ad hoc manner, e.g. using list items prefixed with “yes” or “no”. A solution for restoring the neutrality of a controversial article could be citing reliable arguments in favor of the view that has been underrepresented so far. Eventually, one trustee of the community would judge whether there is a consensus about a particular solution, or simply count the votes, and then implement the solution approved by the community, again without any assistance from the system. A justification for the resulting revision of a page can be given by a descriptive edit summary that links to the section of the discussion page where the respective issue was discussed [Wik09a]. However, authors do not always do this, which sometimes makes it hard to retrace decisions. Note that for procedures with a higher impact, such as deleting an article, it is more highly regimented who may implement a solution. Only users with administrative permissions, which are awarded by public vote, may technically do so.<sup>11</sup>

In large communities such as Wikipedia, these procedures work sufficiently thanks to the large user base; indeed, the quality of articles has been found to strongly correlate with the number of authors [Brä05].

#### 9.1.4.3 Distilling Argumentation Models from Wikipedia

Other researchers have previously analyzed the structure of discussions in Wikipedia in order to obtain a conceptual model. These analyses have been conducted manually, as Wikipedia articles are largely unstructured, and discussions and edit summaries are given in natural language, and

<sup>9</sup>Wikipedia’s different language editions have developed slightly different conventions. The following citations refer to the English Wikipedia. Pointers to related information in other Wikipedias can be found on the respective pages by following the links to other languages.

<sup>10</sup>The English Wikipedia has special building blocks referring to the neutrality of a section of an article, its introduction, or its title. However, this does not yet help to establish a machine-comprehensible relation from the disputed part of a page to the corresponding discussion post.

<sup>11</sup>For the work reported on here, I have not assumed any such technical restrictions, but well-behaved and cooperative users. Encouraging or enforcing orderly behavior is an interesting research question in itself but not considered here.

the software does not restrict the space of possible arguments. The analyses did not result in formal argumentation ontologies and did not have the purpose of developing a strong software support for discussing and solving problems, but they constitute important steps towards a future development of an argumentation ontology for wikis.

CHRISTIAN PENTZOLD and SEBASTIAN SEIDENGLANZ have analyzed how edit summaries and discussion posts are related to changes made to Wikipedia articles, and what types of changes occur. From that, they have derived a conceptual model for Wikipedia argumentations, but not a formal argumentation ontology [PSo6]. JODI SCHNEIDER et al. have classified the comments posted on Wikipedia's discussion pages into 15 types, divided into references to content or actions, such as articles that have been vandalized and formalized this classification as a module of SIOC [SPB10].

## 9.2 Requirements Analysis and Design Decisions

SWiM, as it is presented in this thesis, is the evolution of an older prototype of a mathematical extension [Lan07a] of the IkeWiki general-purpose semantic wiki [Scho6]. This section briefly recapitulates the reasons for choosing IkeWiki back then and argues why these reasons were still valid when I had to make this choice once more, facing the application scenarios introduced in [section 6.1.2](#).

### 9.2.1 Original Reasons for Choosing IkeWiki

SWiM 0.1, the predecessor of the version 0.2 presented in this thesis, was developed in the first half of 2006. Its purpose was prototyping the possibilities of editing OMDoc documents in a wiki. The requirements were [Lan07a, section 4.1]:

- support for semantic web technology, including an RDF triple store, RDF query answering, and optional support for OWL reasoning (because of the OMDoc ontology, an early version of which had existed at that time)
- the possibility to store wiki pages represented in the OMDoc XML language
- the possibility to integrate the OMDoc presentation process, which was fully implemented in XSLT at that time
- the possibility to add components to the user interface, e.g. a mathematics-specific navigation bar
- support for user profiles

Among the systems evaluated as a possible foundation for SWiM, there were SMW 0.2a, IkeWiki Snapshot 2006-03-08, and Rhaptos 1.5, the system driving Connexions (cf. [section 2.4.8.5](#)). IkeWiki was found to satisfy all of the above-mentioned requirements, as it represented wiki pages in XML and had an integrated RDF triple store, support for SPARQL inline queries, a modular rendering process and user interface, and FOAF user profiles.

### 9.2.2 Requirements for Project Management

The decision for IkeWiki was confirmed once more from the point of view of the Flyspeck collaborative formalization project management scenario introduced in [section 6.1.2.5](#). In a feasibility study conducted in early 2008, we compared the feature set of SWiM, then based on IkeWiki 2.0, and an initial prototype on top of SMW 1.0<sup>12</sup> [[LMRo8](#)]. The wiki was envisaged to host human-comprehensible as well as computerized representations, to support collaborators in semantically annotating the human-comprehensible representation and interlinking it with the computerized representation, and to serve as a platform for coordinating work to be done, as outlined in [section 6.1.2.5](#). As minimum annotation requirements, we identified [[LMRo8](#)]:

- categorization by topic, including the possibility to use a classification scheme (cf. [section 2.1.7.4](#))
- support for creating and using a project-specific metadata vocabulary
- a vocabulary of formal and informal dependency link types
- typed discussion posts

IkeWiki/SWiM, with its support for importing RDFS and OWL ontologies, full SPARQL queries, linked data style navigation, and a semantic representation of mathematical objects, satisfied these criteria better than SMW. In contrast, SMW can only reference entities from existing ontologies but not fully import their axioms, its query language does not support unrestricted negation (e.g. querying a lemma for the absence of a classification as “proven”), and mathematical objects are presentation-only (cf. [section 9.1.3](#)). Advantages of SMW were its more concise query language and its support for ad hoc formalization by first annotating pages with terms from a local ontology and formalizing them later on by editing their wiki pages. In contrast, annotating IkeWiki pages with a custom vocabulary first requires defining these terms using the built-in ontology editor.

### 9.2.3 Requirements for Maintaining OpenMath CDs

Due to the similarity of OpenMath CDs and OMDoc documents, I did not consider any alternatives to the existing SWiM for OMDoc when establishing requirements for supporting the maintenance of the official and contributed OpenMath CDs. Initial requirements were established in informal conversations with core members of the OpenMath community, mostly via the [om@openmath.org](mailto:om@openmath.org) mailing list. Further requirements were gathered after the deployment of an initial prototype, again via that list:

**CD Language:** As a general prerequisite for supporting the specific maintenance workflows, SWiM **MUST** support storing, editing, publishing, and importing/exporting not only OMDoc documents but also OpenMath CDs, as well as notation dictionaries in the pattern-based language presented in [section 2.4.5.2](#).<sup>13</sup> This **SHOULD** be realized as generically as possible, so that, in future, support for further languages can be added with little additional effort.

<sup>12</sup>We had to enable SMW to import the Twelf sources of the Flyspeck formalization into wiki pages by an extension; due to an existing translation from Twelf to OMDoc (cf. [section 8.2.1](#)), none of that had to be done for SWiM.

<sup>13</sup>This requirement dates back to the time when this syntax had – unsuccessfully – been proposed for inclusion into MathML 3 [[ABC+o8](#), chapter 8.6] and thus also OpenMath 3.

**Document Metaphor:** Despite the database-like nature of CDs, the OpenMath community is more familiar with treating them as documents, as discussed in [section 6.2.6](#). In particular, that means that any content, including metadata, **MUST** be editable in a document interface.

**Maintenance Workflows:** Each of the CD maintenance workflows from [sections 6.1.2.1 to 6.1.2.3](#) **MUST** be supported. SWiM **SHOULD** address each of the problems pointed out in the descriptions of these workflows.

**Coexistence with Subversion:** SWiM **MUST** give access to the OpenMath CDs in their primary Subversion [[Apaal](#)] repository.<sup>14</sup> Editing a CD in SWiM **MUST NOT** disrupt the integrity of the files in the repository. Semantically irrelevant features such as XML comments (which are used despite the existence of *CDComment*) and the order of XML elements (cf. the “document metaphor” above) **MUST** be preserved.<sup>15</sup>

## 9.3 Architecture

This section explains the architecture and functional range of SWiM: the underlying IkeWiki system ([section 9.3.1](#)), mathematics-specific extensions of its storage backend ([section 9.3.2](#)), and mathematics-specific extensions of its user interface ([section 9.3.3](#)). Technical details can be found in [appendix C.4](#).

### 9.3.1 The IkeWiki Base and its Extension into SWiM

The original features of IkeWiki that made it a suitable base for SWiM have been mentioned above in [sections 9.2.1 and 9.2.2](#). This section summarizes further notable features of IkeWiki, as well as my extensions that turn IkeWiki into SWiM.

The following IkeWiki features are relevant in the context of this thesis; [appendix C.4.1](#) provides further background on ontology and reasoning support:

**Ontology Installation:** IkeWiki ships with a number of ontologies, including Dublin Core (DCMES and DCMI Terms), FOAF, SIOC, and ccREL, which can be preloaded into the database when installing the system.

**Permission Configuration:** Compared to other wikis, IkeWiki has a relatively powerful permission management. Permissions can be assigned to users and groups of users (named “roles”). There are dozens of actions that users can perform, each of them identified by a URI; examples include viewing the revision history of a page, editing metadata, writing a discussion post, or importing an external resource. These actions are grouped into sets such as “viewing”, “editing”, or “management”. Users or groups can be granted the permission to

<sup>14</sup>With regard to the first phase of development, the motivation for this requirement was to ensure user acceptance by giving them access to the same CDs that they already knew from the Subversion repository – and not, e.g., an out-of-sync copy. In the long run it was clear that SWiM would never be able to perform *all* desirable CD maintenance tasks. For example, search and replace operations across multiple documents are hard to realize in wikis, as discussed in [section 9.6.4.3](#).

<sup>15</sup>As SWiM processes CDs as XML documents and not as text files, the community request for even preserving the whitespace layout could not be met.

perform a certain action or all actions of a set – generally, or for a specific set of resources identified by a URI pattern. In the OpenMath CD installation of SWiM, user groups have been set up for visitors (allowed to comment on everything), CD editors (allowed to edit the CDs)<sup>16</sup>, and administrators (additionally allowed to edit special pages like the entry page).

**Reasoning:** RDFS reasoning is enabled by default in the triple store.

SWiM features the following extensions over IkeWiki:

**Ontologies:** On setup, the OMDoc and OpenMath CD ontologies as well as the mathematical extensions to the SIOC argumentation ontology are preloaded.

**Mathematical Services and Translations:** Mathematical knowledge items, represented OMDoc or in the OpenMath CD language, are represented as resources. Their XML representations are stored in IkeWiki’s relational wiki page database. RDF outlines in terms of the above-mentioned ontologies is transparently extracted from them and fed into the triple store, in order to provide each service with the representation format it understands. The following primitive services have been integrated:

**Editing:** Mathematical knowledge items are edited using the semantically extended document editor presented in [section 6.2.3](#) with formula editing plugin introduced in [section 6.2.4](#). Metadata can be edited in an RDF-based form or in the document editor, as explained in [section 6.2.6](#). The translation approach introduced in [section 8.3.3](#) enables both interfaces to access the same metadata from the RDF triple store.

**Publishing:** While IkeWiki does not expose its internal RDF graph as linked data, it can still be browsed in a linked data style inside the wiki environment, as explained in [section 6.4.1.2](#). The RDF graph includes links of all types supported by the OMDoc and OpenMath CD ontologies, most importantly whole→part and dependency links. Mathematical knowledge items are published using the machinery presented in [section 6.4.2.8](#). Knowledge items are displayed completely with their subparts. Published mathematical objects are annotated with parallel content markup. Presentation markup symbols are linked to the wiki page where they are declared.

**Argumentation:** Any mathematical knowledge item can be discussed, following the structure of the SIOC argumentation ontology and its extension by mathematical problem and solution types. The structure of these discussions is exploited for prototypical problem solving assistance (cf. [section 6.6](#)).

**Information Retrieval:** The RDF representation of the structural outline of the mathematical knowledge, covering everything except the inner structures of mathematical objects, can be queried using SPARQL (cf. [section 6.5.2](#)). This functionality is provided by IkeWiki; all that SWiM adds is the extraction of RDF from mathematical markup.

Integrating so many different services in a coherent environment required, independently of the choice of IkeWiki, harmonizing them with each other. Re-rendering wiki pages

---

<sup>16</sup>In the OpenMath Society, there is the official position of a “CD editor”. That person oversees changes to the CDs in general but is not the only person who changes them.

**Article** | **Discuss** | **Metadata** | **Context** | **Edit** | **Annotate** | **History**

---

# arith1

Identifier: cd:arith1  
 Types:

omo:ContentDictionary - omo:SignatureDictionary - omo:ContentDictionary u  
 omo:ContentDictionaryGroup - omo:OpenMathConcept - rdfs:Resource

**CD Base:** <http://www.openmath.org/cd>

**Date:** 2008-10-02

**Version:** 3

**Review Date:** 2006-03-30

**Status:** draft

This CD defines symbols for common arithmetic functions.

## Symbol Definition (lcm)

**Role:** application

**Title:** Least Common Multiple

**Description:** This n-ary operator is used to construct an expression which represents the least common multiple of its arguments. If no argument is provided, the lcm is 1. If one argument is provided, the lcm is that argument. The least common multiple of x and 1 is x.

**Pragmatic MathML:**

```
<lcm />
```

type : MathMLType

**Property:**

$\text{lcm}(a,b) = a*b/\text{gcd}(a,b)$
XML (OpenMath)
Content: MathML
Prefix form
Presentation MathML

$\text{lcm}(a,b) = \frac{(ab)}{\text{gcd}(a,b)}$

**Property:** for all integers a,b | There does not exist a c>0 such that c/a is an Integer and c/b is an Integer and  $\text{lcm}(a,b) < c$ .

XML (OpenMath)
Content: MathML
Prefix form
Presentation MathML

$\forall a,b(c \in \mathbb{Z}) b \in \mathbb{Z} \rightarrow \exists c,c > 0, c/a \in \mathbb{N}, c/b \in \mathbb{N} \wedge \text{lcm}(a,b) < c$

**User**

- User Page
- Administrator
- Preferences
- Logout
- Theme:

[tundra] [series]

**Navigation**

- IkeWiki Help
- Recent Changes

**Search**

**Edit**

- Create Resource
- Create Class
- Create Property
- Create Multimedia
- Create Template
- Delete Resource
- Add Relation
- Remove Relation

**System**

- Manage Action Sets
- Manage Users
- Manage Roles
- Manage Languages
- Namespaces
- Flush Caches
- Rebuild Index
- Restart System

**Tools**

- Export
- Import
- Print View
- Permalink
- Refresh

**References**

- outgoing
  - has author
    - Forum:11C81A927490
  - contains Symbolic definition
    - arith1-abs
    - arith1-big gcd
    - arith1-big lcm
    - arith1-divide
    - arith1-gcd
    - arith1-lcm
    - arith1-minus
    - arith1-plus
    - arith1-power
    - arith1-product
    - arith1-root
    - arith1-sum
    - arith1-times
    - arith1-unary minus
  - has font
    - type
      - ContentDictionary
      - OpenMathConcept
      - Resource
      - Incoming
      - untyped
- Socialise
  - Digg this
  - Post to del.icio.us
  - Post to Furl
  - Post to Mignolla
  - Post to Yahoo
  - Permalink

Figure 9.1: The *arith1* OpenMath CD in SWiM; published view in the center, RDF links on the right

affected by changes made to notation definitions is a prime example, which is detailed in [section 9.3.2.4](#).

**Subversion Client:** The Subversion client required for maintaining the official and contributed OpenMath CDs is introduced in detail in [section 9.3.2.2](#).

**Subscription to Discussions:** Registered users can subscribe to discussion forums by e-mail, as detailed in [section 9.4.3](#).

### 9.3.2 Storage Backend

This section describes the functionality that SWiM adds to the storage backend of IkeWiki. [Appendix C.4.2](#) provides further technical background about the implementation.

#### 9.3.2.1 Document Translation and Storage

SWiM relies on IkeWiki's storage backend for storing mathematical knowledge items. Most of the services that SWiM integrates operate on small units of knowledge, as pointed out in [section 8.3.1](#). As many other semantic wikis, IkeWiki assumes that each resource is represented by a wiki page; it cannot treat fragments of pages as resources of their own. Therefore, any relevant mathematical knowledge item has to be represented on a wiki page of its own. SWiM considers mathematical properties (of OMDoc statements or OpenMath symbol definitions) the smallest relevant knowledge items that can reasonably form a self-contained page.

Whenever a wiki page is saved inside SWiM, or whenever an external document is imported into SWiM (from a file, or from a connected Subversion repository, as explained below in [section 9.3.2.2](#)), several translations make sure that each integrated service is provided with the representation format and granularity that it understands. These translations yield fine-grained wiki pages corresponding to knowledge items (cf. [section 8.3.2](#)), an RDF outline of the mathematical knowledge (cf. [section 8.1.2](#)), and a metadata representation that is both accessible to RDF queries and editing forms and to the document editor and the publishing process (cf. [section 8.3.3](#)). These translations are reversed when exporting a wiki page to the file system or a Subversion repository and thus preserve the original file layout. [Appendix C.4.2.1](#) explains the exact translation procedure.

#### 9.3.2.2 A Client for Versioned Repositories

While the integration of SWiM with Subversion repositories has primarily been motivated by the requirements for maintaining the official and contributed OpenMath CDs, which are hosted in such a repository (cf. [section 9.2.3](#)), Subversion has also frequently been used for other collections of mathematical knowledge. Examples include the sources of the Flyspeck project (cf. [section 9.2.2](#)) and MICHAEL KOHLHASE's lecture notes (cf. [section 6.1.2.4](#)).

SWiM can be used as a Subversion client, where the original IkeWiki database acts as a working copy for the files in the Subversion repository. Each of the namespaces that IkeWiki/SWiM uses to group pages can be defined as a Subversion working copy; [table 9.1](#) shows the configuration of the OpenMath CD installation of SWiM. On every read access to a wiki page whose original



Table 9.1: Subversion collections accessible in the OpenMath CD installation of SWiM

Prefix	Subversion URL	Description
cd	<a href="http://svn.openmath.org/OpenMath/OpenMath3/cd/MathML/*.ocd">http://svn.openmath.org/OpenMath/OpenMath3/cd/MathML/*.ocd</a> <sup>a</sup>	OpenMath/MathML 3 draft CDs
ntn	<a href="http://svn.openmath.org/OpenMath/OpenMath3/cd/MathML/*.ntn">http://svn.openmath.org/OpenMath/OpenMath3/cd/MathML/*.ntn</a> <sup>a</sup>	OpenMath 3 notation definitions (unofficial)
cd2	<a href="http://svn.openmath.org/OpenMath/www/cdfiles2/cd/">http://svn.openmath.org/OpenMath/www/cdfiles2/cd/</a>	official OpenMath 2 CDs
cd2contrib	<a href="http://svn.openmath.org/OpenMath/www/cdfiles2/contrib/cd/">http://svn.openmath.org/OpenMath/www/cdfiles2/contrib/cd/</a>	contributed OpenMath 2 CDs

<sup>a</sup> In the transition from OpenMath 2 to the next version, the OpenMath Subversion repository is currently (spring 2011) undergoing a heavy restructuring. All of these URLs were valid at the time of submitting this thesis.

source is in a Subversion repository, SWiM tries to retrieve its latest revision. Conversely, SWiM commits every change made to a page or its metadata to the repository. Both data transfers are subject to the translations mentioned in [section 9.3.2.1](#).

Listing 9.1: Log message for a revision of the description of the *transcl#sin* symbol

```
r1234 | clang | 2009-05-11 13:06:41 +0200 (Mon, 11 May 2009) | 2 lines
[Administrator@SWiM] replaced metadata field dc:description
Actually changed fragment cd:transcl+sin
```

The log messages that SWiM creates when committing to the repository describe changes as closely as possible, as shown in [listing 9.1](#), even though, from the repository’s point of view, just “something” has been changed in a file. The log message includes the identifier of the exact fragment that has been changed<sup>17</sup>, and a description of the change that has been made to that fragment. In the document editor, the author can give a custom summary, which would then be used in the first line of the log message. In the metadata editing form, the user cannot freely choose an editing summary; therefore the log merely mentions the metadata field that has been edited (cf. [listing 9.1](#)).

### 9.3.2.3 Testing the Subversion Client and Document Translation

A test in the OpenMath CD repository confirmed that the Subversion client introduced above is operational and that the translations explained in [section 9.3.2.1](#) work correctly and do not disrupt the integrity of the CD files in the repository. To each CD in the OpenMath/MathML 3 draft collection, we applied a “null edit”, i.e. we opened it in the document editor and saved it without changes<sup>18</sup>. The expected result was that the CD files were not changed – except for whitespace changes owed to XML parsing and serialization. That was confirmed by manually inspecting the

<sup>17</sup>The naming of CDs and parts thereof that is used in SWiM varies from OpenMath conventions and instead reflects the design of SWiM (*nsprefix:localname* document URIs, and names of fragments of imported documents concatenated with “+”) but is still close enough to OpenMath to be recognizable.

<sup>18</sup>Null edits are a common means of enforcing maintenance actions in a wiki [[Meta](#)].

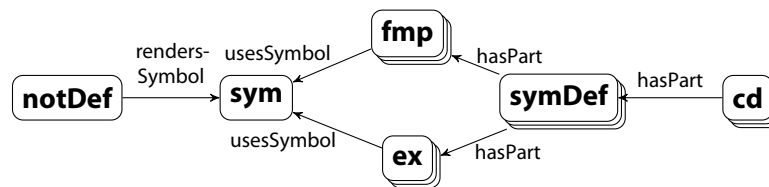


Figure 9.2: Finding pages (depicted as stacks of nodes) affected by changes to a notation definition. Both *sym* and the *symDefs* are instances of the *omo:SymbolDefinition* class.

XML diff of each such edit. This action also made sure that the first *actual* change made to a CD in SWiM would yield a readable text-based Subversion diff.

#### 9.3.2.4 Efficiently Publishing Formulæ when Notation Definitions are Changeable

Rendering the mathematical knowledge items stored in SWiM to a human-comprehensible presentation is expensive – due to the inherent complexity of rendering semantic mathematical markup (cf. [section 6.4.2](#)), but also for technical reasons of the IkeWiki environment and the services it integrates. Therefore, XHTML+MathML documents should be cached and only regenerated when their original semantic markup changes – or when any notation definition changes that has been used for rendering the document. Any change to the<sup>19</sup> notation definition of a symbol  $\sigma$  requires all the cached presentation markup generated from mathematical objects containing  $\sigma$  to be invalidated and re-rendered upon the next request. Doing this as efficiently as possible requires (i) determining when a notation definition has been added, deleted, or changed, and (ii) identifying the minimal set of documents affected by that change. With regard to [step \(i\)](#), SWiM currently assumes that every edit changes a notation definition. This assumption is reasonable, as each notation definition constitutes a wiki page of its own, which a user explicitly has to open in order to change it. [Step \(ii\)](#) is performed by querying the RDF triple store for all knowledge items that use the symbol rendered by the notation definition in question. This includes all units that include affected knowledge items as fragments, e.g. the OpenMath CD that includes an affected *FMP*. [Figure 9.2](#) shows the relevant structures in the case of OpenMath CDs.

Rendering a mathematical object can be considered a special case of inline query processing. An inline query on a semantic wiki page usually consists of a predicate  $p: \text{Page} \rightarrow \mathbb{B}$ , a specification of the information that is desired for every page satisfying  $p$  (e.g. its title), and a style for formatting the result. A mathematical object on a SWiM wiki page can be considered a query for the notation definitions of the symbols used in the object, where for every symbol only the most appropriate rendering is included in the result set and the result is “formatted” by rendering the symbols according to the rendering specifications in the result set. In this setting, one can determine whether a change to a page (here: a notation definition) affects the result set of a query (here: a

<sup>19</sup>The current implementation assumes one notation definition per symbol in the knowledge base. Determining the minimal set of documents affected by a change to one out of multiple notation definitions for a symbol is non-trivial, as discussed in [section 3.2.2.6](#).

This CD defines symbols for common arithmetic functions.

## Symbol Definition (lcm)

[\[open this\]](#)

**Role:**

application

**Title:**

Least Common Multiple

**Description:**

This n-ary operator is used to construct an expression which represents the least common multiple

Figure 9.3: “Open this” link to a fragment of a larger unit

mathematical object) by checking whether the object contains a particular fixed symbol, which requires linear time w. r. t. the size of the object.<sup>20</sup>

### 9.3.3 User Interface

This section describes two features that SWiM adds to the user interface of IkeWiki, beyond its mere extension by an editor and a publication process for semantic mathematical markup. [Appendix C.4.3](#) provides further technical background about the implementation.

#### 9.3.3.1 Giving Local Access to the Editor

SWiM displays large units of knowledge in cohesion for convenience of reading. For example, an OpenMath CD is displayed with all of its symbol definitions, even though they are conceptually separate knowledge items and even physically reside on wiki pages of their own, as explained in [section 9.3.2.1](#). Nevertheless, SWiM tries to give access to the *editor* as locally as possible by enabling authors to open the smallest knowledge item containing the passage they intend to change as a wiki page of its own, as shown in [figure 9.3](#), and editing that page. In combination with the above-mentioned Subversion client, local editing allows SWiM to generate log messages that refer exactly to the knowledge item that has been edited.

#### 9.3.3.2 Argumentative Discussions

SWiM extends the SIOC-based discussion forums of IkeWiki by support for the SIOC argumentation module and its mathematical extensions and a very simple semi-automatic problem solving assistant, as presented in [section 6.6](#). In the current implementation, the extension of the user interface to the generic flow of argumentation has been hard-coded, whereas the domain-specific issue and idea types applicable in a concrete situation are determined dynamically. That allows privileged users to customize the domain-specific part of the argumentation ontology to the requirements of the community, using the ontology editor built into IkeWiki. Support for some

<sup>20</sup>For general queries, this problem is far more complex, as the satisfiability problem for propositional boolean expressions is  $\mathcal{NP}$ -complete. In database research, the area of problems touched on here is known as “materialized view maintenance” [GM99].

actions that are typical solution patterns, such as deleting a knowledge item, had been available in IkeWiki before, whereas I have implemented an assistant for creating a knowledge item related to the one being discussed as a first proof of concept for additional assistance.

To demonstrate the system, I consider the situation that there is a definition, of which it is not clear whether it is useful. The user Alice wants to report that issue. She opens the discussion page for the definition, reports a new issue, and thus starts the discussion thread depicted in [figure 9.4](#). As a specific type of that issue, she can select any type that is applicable to definitions. Afterwards, she realizes that her statement might not have been entirely clear, and appends an elaboration. Bob does not agree that there is actually a problem with the definition and voices his position. Cecil has the idea that the problem could be solved by giving an example; he contributes his idea by clicking the “Idea” reply button in the issue post and selecting an idea type that is applicable to definitions whose utility is unclear. Dan argues from his previous experience that examples are useful.

Now assume that Alice replies to the idea with another agreement and that, after that, Eric, a moderator of the knowledge base visits the theorem: By then, SWiM will have identified the idea to provide an example for the definition as the best one to resolve the issue and display a message that proposes this, offering a link to start a semi-automatic assistant ([Figure 9.5](#) shows a similar case). If Eric decides to provide the example and clicks on the link, a new example page, pointing to the original knowledge item, will be created, and he can fill out the template (cf. [figure 9.6](#)). SWiM is not yet capable of closing a discussion thread by posting an auto-generated decision statement; therefore, that was done manually here.

[Figure 9.7](#) shows the complete discussion as an RDF graph, where the argumentative structure forms an overlay graph to the physical thread structure. [Listing C.2](#) on page 383 provides an example of how to query such graphs.

## 9.4 How SWiM Supports OpenMath CD Maintenance Workflows

This section explains how the three OpenMath CD maintenance workflows introduced in [section 6.1.2](#) are realized in SWiM.

### 9.4.1 Quickly Fixing Minor Errors

By integrating a document editor as well as editors for metadata and mathematical objects, SWiM offers dedicated support for editing all major structures of OpenMath CDs: the document-like structure (i.e. the CD top level, the symbol definitions, and their properties), metadata of such structural units (e.g. their informal descriptions or the date of revision), and OpenMath objects inside *FMPs* and examples.

SWiM realizes the “minor fixes” workflow introduced in [section 6.1.2.1](#) as follows: First of all, manually updating the working copy is no longer necessary. In the published view of the CD, the author has to open the exact fragment of that contains the error on its own wiki page – unless the error is in a top-level metadata field. SWiM embeds links to all such fragments into the published documents, as explained in [section 9.3.3.1](#). Alternatively, one can use the linked data

Article
Discuss
Metadata
Context
Edit
Annotate
History

### Discussion about SampleDefinition

**Issue (UnclearWhetherUseful): Is this definition of any use?**
Alice
  
Mon, 27 Oct 2008 01:05:07 +0100

Can it be applied for anything relevant?

Idea
general idea
Elaboration
Argument
Position
Decision
Reply

**Decision: Re: Is this definition of any use?**
Administrator
  
Mon, 27 Oct 2008 02:02:04 +0100

It seems that I should give a rocket science example for applying this definition. I will do so.

Reply

**Position: Re: Is this definition of any use?**
Bob
  
Mon, 27 Oct 2008 01:23:03 +0100

Oh, come on, don't complain, mathematics is the science of abstract nonsense, after all.

Reply

**Elaboration: Re: Is this definition of any use?**
Alice
  
Mon, 27 Oct 2008 01:13:50 +0100

I mean, I cannot imagine any application in science or technology where this might be helpful.

Argument
Reply

**Idea (ProvideExample): Re: Is this definition of any use?**
Cecil
  
Mon, 27 Oct 2008 01:27:43 +0100

Why don't you provide an example that shows how to apply this to the domain of rocket science?

Elaboration
Argument
Position
Decision
Reply

**Evaluation: Re: Re: Is this definition of any use?**
Dan
  
Mon, 27 Oct 2008 01:59:23 +0100

My students have never understood definitions of that kind easily. But as they all like rocket science, they have always found examples from that domain helpful.

Position
Reply

**Position: Re: Re: Is this definition of any use?**
Alice
  
Mon, 27 Oct 2008 02:00:30 +0100

Yes, please, I want such an example right now!

Reply

New Issue

general issue
general issue
UnclearWhetherUseful
Reinvention
InappropriateForDomain
UncommonStyle
Overspecified
Underspecified
TooManySubparts
Incomprehensible

New Comment

Figure 9.4: A complete argumentative discussion thread (mind the chronological order when reading!)

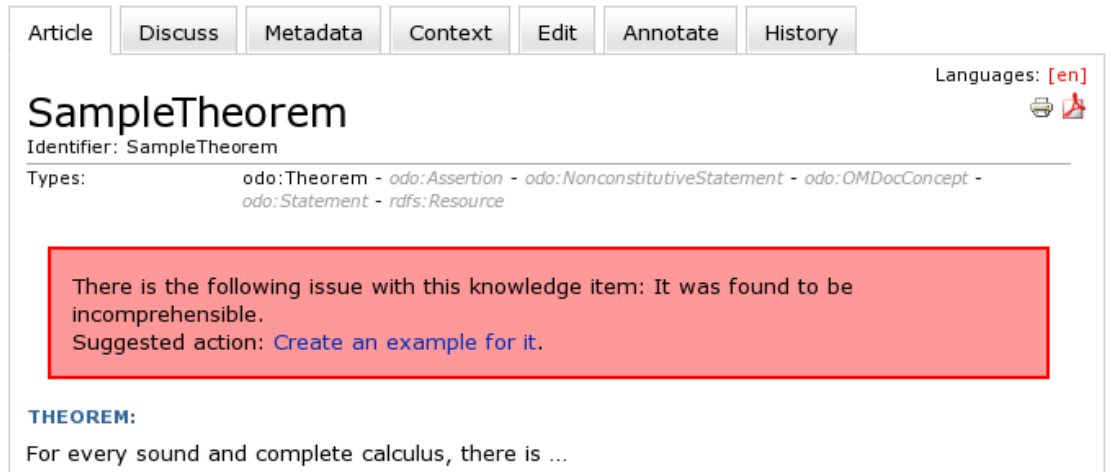


Figure 9.5: Warning about an issue and the offer to solve it

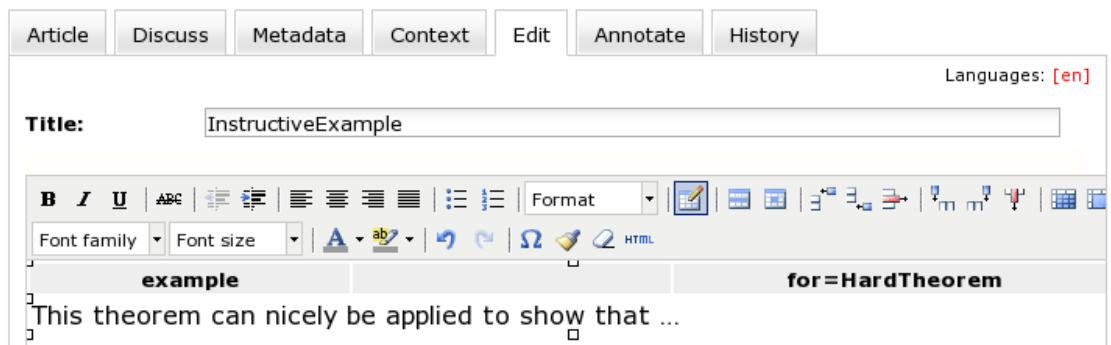


Figure 9.6: Editing the newly created example

style navigation tree introduced in [section 6.4.1.2](#) and follow the *hasPart* links.<sup>21</sup> The document editor, with its integrated formula editor, can be used for text paragraphs (in *CMs* or examples), for formulae, and for metadata. In this editor, one can provide a log message describing the change made ("Summary" field in [figure 6.2](#) on page 191). Note that, in contrast to other wiki systems, IkeWiki does not offer a check box by which authors could distinguish a minor from a major edit; instead, we assume that major edits are only made after a discussion (see below). On confirming the edit, SWiM appends a reference to the fragment that has actually been changed to the log message, as shown in the last line of [listing 9.1](#), and commits the whole CD to the repository. Metadata can alternatively be edited using a form. There, SWiM generates the complete log message and makes it refer to the metadata field that has been changed, as shown in [listing 9.1](#).

<sup>21</sup>Getting from a CD to a mathematical property or example would take two steps.

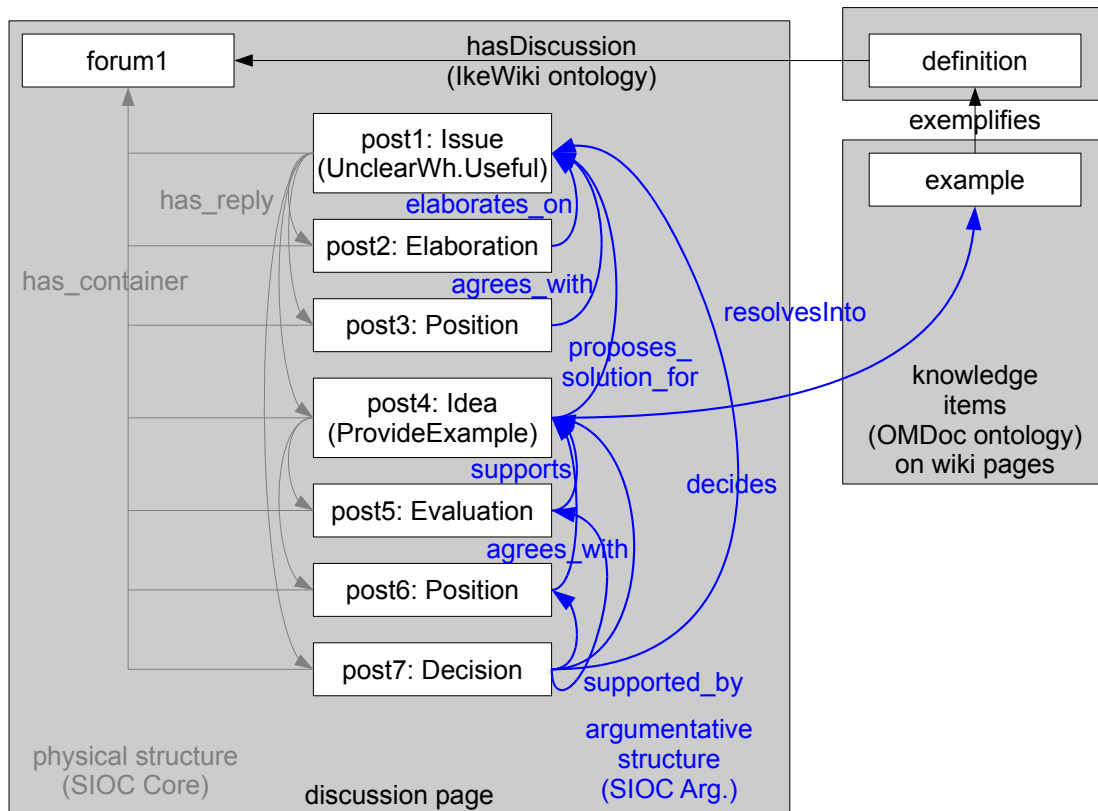


Figure 9.7: RDF graph of the sample discussion (cf. figure 9.4)

### 9.4.2 Fixing and Verifying Notations

SWiM renders OpenMath CDs into published documents using the service described in [section 6.4.2.8](#). The notation definitions used for rendering symbols in mathematical objects are editable in SWiM. SWiM improves the workflow for fixing and verifying notations, which has been introduced in [section 6.1.2.2](#), in the following ways: A developer can directly follow the link from the occurrence of a rendered symbol to its *CDDefinition* (cf. [section 6.4.2.8](#)). From there, the notation definition is one more click away via the navigation tree.<sup>22</sup> When a notation definition has been changed, exactly those published documents are re-rendered that are affected by that change, as explained in [section 9.3.2.4](#). For immediately verifying a notation definition after changing it, one can also check the preview rendered from it, as explained in [section 6.4.2.8](#).

### 9.4.3 Peer Review and Preparing Major Revisions by Discussion

SWiM supports discussions about issues with OpenMath CDs, as introduced in [section 6.1.2.3](#), in the following ways: By offering a discussion forum for each knowledge item (i.e. for each CD,

<sup>22</sup>SWiM does not currently provide even more direct access from a rendered symbol to the notation definition that has been used for rendering it, as explained in [section 6.4.2.4](#).



You are currently not watching this discussion forum.

☐ watch it ☒ automatically watch discussions once I posted a comment

Figure 9.8: Configuring the subscription to a discussion forum (displayed below the posts)

#### Ongoing Discussions

■ **Unresolved issues (by date of issue, oldest issues first):** [FrontPage](#) (2008-09-04) — [quant1](#) (2008-09-04) — [quant1](#) (2008-09-04) — [s data1](#) (2008-09-04) — [s data1](#) (2008-09-04) — [s data1+median](#) (2008-09-04) — [transc1](#) (2008-09-04) — [transc1](#) (2008-09-04) — [transc1](#) (2008-09-04) — [quant1](#) (2008-09-05) — [set1](#) (2008-09-06) — [linalg1](#) (2008-09-13) — [linalg1](#) (2008-09-13) — [linalg2](#) (2008-09-13) — [arith1](#) (2008-09-20) — [arith1](#) (2008-09-20) — [limit1+null](#) (2008-09-20) — [limit1+tendsto](#) (2008-09-20) — [linalg1](#) (2008-09-20) — [logic1](#) (2008-09-20) —

■ **CDs being discussed (by date of latest discussion post; most recent first):** [plangeo2](#) (2009-09-14) — [list1](#) (2009-03-11) — [list1](#) (2009-03-11) — [logic1](#) (2008-11-05) — [arith1](#) (2008-10-29) — [arith1](#) (2008-10-29) — [arith1](#) (2008-10-29) — [arith1](#) (2008-10-29) — [fns1](#) (2008-09-21) — [relation1](#) (2008-09-21) — [set1](#) (2008-09-21) — [set1](#) (2008-09-21) —

■ **Symbols being discussed:** [fns1+identity](#) (2008-11-05) — [fns1+lambda](#) (2008-11-05) — [fns1+left compose](#) (2008-11-05) — [fns1+left compose](#) (2008-11-05) — [fns1+left inverse](#) (2008-11-05) — [fns1+range](#) (2008-11-05) — [fns1+right inverse](#) (2008-11-05) — [nums1+NaN](#) (2008-11-05) — [nums1+based integer](#) (2008-11-05) — [nums1+based integer](#) (2008-11-05) — [nums1+e](#) (2008-11-05) — [nums1+e](#) (2008-11-05) — [nums1+gamma](#) (2008-11-05) — [nums1+gamma](#) (2008-11-05) — [nums1+pi](#) (2008-11-05) — [nums1+pi](#) (2008-11-05) — [arith1+divide](#) (2008-10-29) — [arith1+plus](#) (2008-10-29) — [arith1+power](#) (2008-10-29) — [s data1+mean](#) (2008-09-21) —

■ Any other OpenMath concept being discussed: [interval1+interval+MMLexample0](#) (2008-09-27) — [limit1+limit+limit.ex1](#) (2008-09-27) —

■ Any non-OpenMath wiki page being discussed: [FrontPage](#) (2008-09-04) —

Figure 9.9: Results of queries for discussions about OpenMath CD knowledge items (not all results shown)

symbol definition, mathematical property, and example), SWiM allows for discussing problems locally, and for quickly switching between a knowledge item and the discussions about it. Users can indicate the argumentative type of their discussion posts. The argumentation ontology that defines the vocabulary of these types and the discussion workflow has been introduced in [section 6.6](#), its usage in SWiM has been described in [section 9.3.3.2](#).<sup>23</sup> User can set up an e-mail subscription per discussion forum; additionally, they can request to be automatically subscribed to any discussion forum as soon as they post a comment there (cf. [figure 9.8](#)).

The structure of the discussions and their subjects can be queried from the RDF graph. On the entry page of the OpenMath CD installation of SWiM, the following inline queries have been set up in order to draw attention to ongoing discussions, particularly to unresolved issues:

- knowledge items of any type having unresolved issues, oldest issues being listed first (cf. [listing C.2](#) on page 383)<sup>24</sup>
- CDs having any kind of discussion, most recent discussions being listed first
- symbols having any kind of discussion
- any other OpenMath concept (e.g. a mathematical property or an example) having any kind of discussion ([listing C.3](#))
- any non-OpenMath wiki page (e.g. the entry page) having any kind of discussion

Any experienced user can enter additional inline queries on any page he may edit.

<sup>23</sup>The domain-specific extensions described in [section 3.6.2](#) have not yet been deployed in the OpenMath wiki.

<sup>24</sup>Due to the technical restriction that IkeWiki cannot link to discussion posts, one has to query for knowledge items being discussed instead of querying for exact discussion posts. Therefore, the user first has to follow the link to a knowledge item and then open the discussion forum himself. Custom labels for links to query results, e.g. with the *dc: title* of a knowledge item, are not supported either; instead, the wiki page name is displayed.

## 9.5 Related Work

This section provides an overview of the wide range of systems related to SWiM. Not only mathematical wikis in the narrow sense are considered, but also other kinds of web collaboration systems and integrated development environments. Systems that have merely been used for collecting mathematical knowledge without utilizing any of its structures, such as MediaWiki for Wikipedia, are not considered.

### 9.5.1 PlatΩ, Lurch, jEditOQMath, and WIRIS: Editors with Integrated Validation

PlatΩ extends the  $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$  scientific editor towards semantic markup, which is interactively verified by the Ωmega proof assistant, and for which custom notations can be defined [WABo6; AFN+07; DSWo8]. A validation component has not yet been integrated into SWiM. Both systems are comparable in that they minimize the amount of data that is recomputed after a change to a notation definition. Changing a notation definition in SWiM only affects documents that have been published using that notation definition. In PlatΩ, it also affects parser rules, as notation definitions are also used for parsing presentational input into a semantic representation.

Lurch, “a word processor with the ability to check the steps of your work in many areas of mathematics, from calculus to logic” [Lur], comes with an extensible library of OpenMath-encoded mathematical topics, each with varying validation support; in elementary algebra documents, for example, each derivation step can be validated.

The jEditOQMath OMDoc editor (cf. section 6.2.1.1 and [Lib10b]), validates documents and is integrated with the ActiveMath e-learning environment. Validation comprises the XML grammar and the integrity of theory imports. From the editor, one can trigger a build process that publishes a document in the ActiveMath environment, so that it can be previewed. While the editing component of SWiM does not validate, the way from the editor to the preview is shorter there, as it only requires saving the current page.

A final, simple example of an editor with built-in validation is the WIRIS formula editor, which validates the types of OpenMath objects against their STS type signatures [MEC+06]. In contrast to the PlatΩ-extended  $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$  and Lurch, this is not a standalone development environment but a component suitable for integration into web environments.

### 9.5.2 PlanetMath/Noösphere: Automatic Linking and other Services

PlanetMath [Plab], the free mathematics encyclopedia mentioned in section 1.4.2, is driven by the Noösphere wiki system specially developed for that purpose [Kro03]. It is not a semantic wiki, as the articles are authored in presentational  $\text{\LaTeX}$ . However, it uses MSC metadata (cf. section 2.1.7.4) for search and navigation. Similarly to SWiM’s support for arguing about problems, users can file errata and addenda. Another remarkable feature is automatic linking, which recognizes named entities in the article content and links them to those articles that have them as labels (titles, defined concepts, or synonyms). The automatic linker has been generalized factored into a reusable component [GKX09]. Noösphere supports threaded comments, and ratings in the four dimensions of correctness, clarity, pedagogy, and language. Porting PlanetMath to an entirely new system built from components presented in this thesis is currently in progress; see section 11.3.1.

### 9.5.3 OpenMath CD Manager: Maintaining Fixed Structures

The OpenMath CD manager [Hea09], which was released in fall 2009 but has not been used by the OpenMath community so far, is the only system besides SWiM that aims at supporting the *maintenance* of OpenMath CDs. It is capable of importing CDs from the file system – but not from a Subversion repository – into a relational database and makes them editable via web forms. CDs are split into fragments of the same granularity as in SWiM, so that, for example, each mathematical property is accessible as one knowledge item in the database. Similarly to SWiM, the OpenMath CD Manager features an elaborate user permission management and assumes that not all users may edit the CDs. Its most outstanding feature is the possibility for non-privileged users to *suggest* changes or additions, which a moderator can then review and finally approve or reject. The flow of an argumentative discussion in SWiM is more complex but allows for incorporating opinions from more collaborators than just one user and one moderator. Secondly, knowledge items of any granularity – above objects – can be argued on, whereas the suggestion facility of the CD Manager is limited to definitions, CDs, and CD groups. A clear advantage of the CD Manager is that a suggestion can directly be turned into an actual knowledge item, whereas the content of a SWiM discussion post is unstructured. It may contain OpenMath objects that could be copied and pasted when editing the respective knowledge item, but SWiM would not recognize that automatically. In a brief review of the OpenMath CD installation of SWiM, DAVID HEATH criticizes its poor performance, and the unclear navigation. Performance was not a primary goal when implementing SWiM, whereas HEATH conducted explicit performance optimizations – which is, arguably, easier in an environment that is restricted to a single representation format. The comprehensibility of SWiM’s user interface will be discussed in detail in [section 10.5](#).

### 9.5.4 Connexions/Rhaptos: Structured Semantic Markup Editing

The Connexions open courseware repository mentioned in [section 1.4.2](#) is driven by the Rhaptos system. Rhaptos integrates publication services, such as the notation selection facility mentioned in [section 2.1.5](#), and editors for metadata (cf. [section 2.4.8.5](#)) and structured documents. The latter editor is capable of validating XML (cf. [section 6.3.3.3](#)) and giving local access to fragments of semantic markup, as discussed in [section 6.2.7.2](#). This “in-place” editing and the “section editing” links offered by many wiki systems, including MediaWiki, served as an inspiration for the links that give local access to fragments in SWiM (cf. [section 9.3.3.1](#)). There is a one-dimensional rating facility; detailed comments about knowledge items can merely be given by e-mail.

### 9.5.5 ProofWiki, Mizar Wiki, and Logiweb: Formalized Mathematics

ProofWiki<sup>25</sup> is a prototype of a wiki for managing a formalized library [Prod; CK07]. The semantic structures of the content are, however, only used by the integrated Coq proof assistant, not to facilitate browsing or editing. Human-readable descriptive texts are written in non-semantic L<sup>A</sup>T<sub>E</sub>X. ProofWiki is no longer under development, but some of its ideas will survive in the wiki frontend for the MML mentioned in [section 1.4.2](#), whose development is currently in progress.

<sup>25</sup>not to be confused with the same named site mentioned in [section 1.4.2](#), which is merely a collection of mathematical proofs driven by a non-semantic MediaWiki

Logiweb is a system for publishing verified mathematical content on the Web [Gru07].  $\text{\TeX}$  and PDF output generated from the built-in formal language are the main publication targets. In contrast to ProofWiki and the Mizar wiki, Logiweb is not committed to a particular logical foundation. Logiweb is highly customizable: In the same way as mathematical documents, users can write their own notation parsers, renderers, proof checkers, and tactics for proof checkers. In contrast, only the ontologies – for the structures of mathematical knowledge as well as argumentative discussions about problems – are customizable in SWiM (cf. section 9.6.4.1 for an elaboration of that aspect). In contrast to wikis, which allow for modifying the content of a page, Logiweb adopts the immutability paradigm of pure functional programming to ensure that a document that has once been verified and published remains valid. The document itself cannot be changed any more, but an author can derive a copy from it. A new revision of a wiki page could analogously be considered a modification of a copy of the immutable previous revision, but the “current latest version” of a page, which is always accessible from the same URL, is mutable (compare the discussion in section 3.4.4.2).

### 9.5.6 ASciencePad and Mathematica-users.org: Computation and Graphing

ASciencePad is a mathematical extension of TiddlyWiki, resulting in a single-user wiki suitable for taking personal scientific notes [Jip]. It renders linear  $\text{\TeX}$ -like input as Presentation MathML, which section 6.2.7.3 reviews from an editing point of view, and integrates services for numerically evaluating expressions and for graphing.

The [mathematica-users.org](http://mathematica-users.org) community site is powered by a MediaWiki rewrite that integrates webMathematica, the web frontend to the Mathematica CAS [Bar]. Thus, it offers most of the numeric and symbolic computation, graphing, and other services of Mathematica. It supports inline Mathematica code in wiki pages as well as uploaded Mathematica notebooks.

### 9.5.7 Lekapidia, Cicero, and SharedHCONE: Wikis with Argumentative Discussions

Argumentation ontologies have been used in two semantic wiki systems so far. In the Lekapidia case study, the DILIGENT argumentation ontology (cf. section 2.1.8.2) was preloaded into an installation of the coefficientMakna semantic wiki [TSL+07]. In that wiki, the authors replayed the collaborative engineering of a simple dessert recipe ontology, which had earlier been engineered following the DILIGENT methodology. In contrast to SWiM, issues were rather raised about the ontology as a whole than about individual entities or axioms. The authors found out that using a wiki “*significantly reduces the effort to capture the arguments in a structured way*” [TSL+07].

Cicero is an SMW extension for argumentation according to a modified DILIGENT ontology<sup>26</sup> [DEM+08; DGG+08]. It has also been integrated into the NeOn toolkit [Neo], an integrated development environment for networked (i.e. versioned, modular, and interdependent) ontologies. In contrast to SWiM, Cicero has not been designed for arguing *about* knowledge items, but for solving problems in projects in general. One wiki page corresponds to one project, issue, or solution proposal (= idea). Arguments are represented as subsections of a solution proposal page.

<sup>26</sup>Cicero’s version of DILIGENT is not currently available as a reusable ontology, but rather hard-coded into the wiki extension.

Cicero offers versatile options for voting and deciding, as outlined in [section 6.6.2](#), whereas the decision support service integrated into SWiM only counts positive and negative positions so far (cf. [section 6.6.3](#)). In the SharedHCONE specialization of Cicero, argumentation has been used as a device for teaching domain ontologies as well as an ontology engineering methodology to learners [KPV+09].

## 9.6 Conclusion and Future Work

The integrated collaboration environment presented in this chapter combines the production of mathematical knowledge with its consumption. This has been achieved by merging the two large strands of integration introduced in chapters 7 and 8 – interactive documents on the frontend, and a knowledge base on the backend side – into a coherent environment. Overall, the SWiM prototype has served three purposes discussed in the following subsections: It has proved the feasibility of integrating heterogeneous services for effectively supporting mathematical knowledge management workflows ([section 9.6.1](#)), presented an improvement over the state of the art of [semantic] wikis ([section 9.6.2](#)), and served as an incubator for developing new services and system components ([section 9.6.3](#)). Retrospectively, the latter has turned out to be particularly important: I have discontinued the development of SWiM, as its IkeWiki foundation is no longer maintained, but its individual components – several of which chapters 6 to 8 describe – will continue to exist in future collaborative MKM environments, such as those discussed in sections [9.6.4.4](#) and [11.3.1](#).

### 9.6.1 Mathematical Services Integrated into a Collaboration Environment

By integrating diverse but complementary services, SWiM supports workflows that occur in realistic scenarios of collaboratively managing collections of mathematical knowledge. Fixing local errors requires editing fragments of semantic markup or metadata fields and committing the changes to the shared repository with a meaningful description. In particular, fixing an error in a notation definition for some symbol requires quick access to the semantic markup fragment that caused the error, and verifying – by checking a re-rendered document containing that symbol – whether the fix performed in the editor solved the problem. Preparing major revisions requires discussing problems and possible solutions, making further collaborators aware of ongoing discussions, e.g. by putting an auto-generated list of discussions on the project homepage, deciding on a solution to implement, and implementing that solution.

The IkeWiki semantic wiki served as a framework for integrating services for editing, publishing, and discussing mathematical knowledge, and for retrieving information from collections of mathematical knowledge. SWiM makes such a collection interactively browsable and gives local access to editors – for documents, formulæ, and metadata – and discussion facilities; the latter have been enhanced by mathematics-specific issue reporting and decision support. Transparent translations performed in the backend provide each integrated service with a representation of the knowledge that it understands, even when the data originate from an external repository. Interactive browsing is partly enabled by links created on rendering knowledge items, but mainly by visualizing the neighborhood of the currently viewed knowledge item in the RDF graph that the backend automatically extracts from the original XML markup of the mathematical knowledge items. Information from the RDF graph also controls the discussion facility in that the supply

of available problem types is governed by the type of the knowledge item to be discussed. For integrating those services that work on XML markup, the core of IkeWiki had to be extended so as to support alternative representation languages besides the original HTML-like wiki markup. By the same approach, any other semantic CMS is likely to be extensible towards mathematics. One cannot generally estimate whether, for any such system, that would require more or less work than for IkeWiki. In the case of IkeWiki, integrating the mathematical services was a task beyond the extension paths that IkeWiki provided for; particularly the introduction of OpenMath and OMDoc as new page formats required heavy modifications to the storage and rendering components.

While SWiM proves the feasibility of integrating a set of heterogeneous services, the integration is still fragile in reality. For example, using the document editor to fix an error in a metadata field of a resource from an external repository involves eight translation steps, as described in [section 9.3.2.1](#) and [appendix C.4.2.1](#):

1. obtaining the latest version of the nearest exportable (and thus also importable) parent document (cf. [section 8.3.2.1](#)) of the resource from the repository and importing it into SWiM's storage backend, where it is split into fragments of a suitable granularity,
2. extracting RDF from each fragment,
3. replacing the values of the metadata fields in the semantic markup by pointers to the RDF triple store,
4. transforming the semantic markup to a HTML representation that the document editor can display, filling in the metadata values from the RDF triple store,
5. transforming the HTML source saved from the editor back to semantic markup,
6. redoing [step 2](#) ...
7. ... and [step 3](#) for the saved page,
8. determining the nearest exportable parent of the resource, reassembling all of its fragments into one document, filling in the metadata values from the RDF triple store, and committing the result to the repository.

From the point of view of this particular task, these steps – any of which could fail (and did fail, in the development phase; more thorough unit testing is likely to improve that) – seems ridiculously complicated. However, in the complete setup of SWiM, each step is required in order to accommodate all integrated services, for example those that access or query metadata in RDF form. A welcome side effect of this complex integration effort was, however, that it helped to validate the CD files. Concretely, three experimental contributed CDs could not be imported from the repository into SWiM. However, validating these CDs against the XML schema would have detected two of the errors and a stronger schema would also have caught the third.



### 9.6.2 Improvements over the State of the Art of [Semantic] Wikis

SWiM delivers a number of improvements over the state of the art of semantic and non-semantic wikis:

- support for *complex semantic markup* – gained by turning the built-in HTML editor into an editor for semantic markup and by integrating an XML→RDF translation
- *exploitation of dependency information* for optimizing internal maintenance processes – here exemplified for re-rendering pages only when notation definitions depended upon have changed
- integration of *external repositories* (here exemplified for Subversion), for which the wiki acts as a working copy – achieved by integrating a respective client and translating the files retrieved from the repository to a representation understood by the services in the wiki
- *argumentative discussions* about problems with wiki pages (here: with the mathematical knowledge items represented by these wiki pages), and assistance with making decisions and solving problems

### 9.6.3 Incubator for New Services and Integration Approaches

This thesis has first presented individual services, then ways of integrating them in documents and knowledge bases, and ultimately an integrated collaboration environment. Historically, these developments were made in reverse order. SWiM served as the “object to think with”<sup>27</sup> about collaboration, and as an incubator for experimenting with new services and integration approaches<sup>28</sup>, which were then factored out into independent reusable components. Two examples for that are the interactive definition lookup facility and the transparent under-the-hood XML→RDF translation.

The functionality that was to become the definition lookup service of the JOBAD library (cf. [section 7.5.1](#)) was inspired by SWiM’s ability to link all symbols in mathematical objects to the wiki pages of their declaration.<sup>29</sup> In the course of developing SWiM, it became clear that (i) without a dedicated plugin API, IkeWiki/SWiM would not be an attractive target for third-party developers contributing services, (ii) a definition lookup facility is not only desirable in SWiM, but also in other environments, such as MMT and TNTBase, and that (iii) documents published in SWiM would benefit from a larger number of assistive services – including services that would require information that SWiM as a sole backend would never be able to provide (e.g. computations, as demonstrated in [sections 7.6.2 and 7.6.3](#)). That gave, from a SWiM point of view, the motivation for designing the JOBAD architecture, which has by now been deployed on top of the MMT and TNTBase backends, with a lightweight proxy enabling access to further information sources.

<sup>27</sup>a term coined by SEYMOUR PAPERT [[Pap80](#)], and applied to semantic software by ANDREA KOHLHASE [[Koho8a](#)]

<sup>28</sup>The role of semantic wikis as “Petri dishes” for prototyping semantic web technologies has first been pointed out by SEBASTIAN SCHAFFERT and MAX VÖLKEL [[SV08](#)] and has been a leitmotif of the workshop series on semantic wikis ever since [[Semb](#)].

<sup>29</sup>This facility was in turn a reimplement of a feature that the OMDoc 1.2 XSLT stylesheets for rendering mathematical objects have already had, but which was not configurable and assumed a file system environment.



Sections 9.6.4.4 and 11.3.1 discuss the actual integration of JOBAD services into the potential successors of SWiM.

The XML→RDF translation in SWiM started as a hard-coded Java implementation that extracted RDF triples from an OMDoc document and directly fed them into IkeWiki's triple store. Preparing SWiM for maintaining the OpenMath CDs required similar support for the OpenMath CD format. Therefore, in order to facilitate the effort of maintaining this functionality inside SWiM, I developed the Krextor XML→RDF library (cf. [section 8.1](#)), having a generic translation core and specializations to OMDoc and OpenMath CDs; support for further input and output formats was added soon after. The availability of this independent translation library facilitated both unit testing, which no longer required the full SWiM environment, and the integration of an OMDoc→RDF translation into other systems – here, concretely, the TNTBase-powered repository of MICHAEL KOHLHASE's lecture notes (cf. [section 6.1.2.4](#)). Conversely, enhancements that have been made to the Krextor library outside of SWiM could now be made available inside SWiM with little effort. That would, for example, allow for publishing RDFa-annotated documents with SWiM, and for editing OWL ontologies in SWiM, as discussed below in [section 9.6.4.1](#).

## 9.6.4 Future Work

This section discusses conceptual innovations and improvements that could be realized in SWiM or a successor system, but also technical aspects of a possible reimplementaion. Improving OpenMath CD maintenance and related tasks is skipped here but covered in [section 10.7](#) instead, as the evaluation of the CD maintenance support covered in that chapter provides a better background for such a discussion.

### 9.6.4.1 Application to Ontology Engineering, System Ontology Customizations

[Chapter 4](#) introduces OMDoc as an expressive language for formalizing and documenting heterogeneous and modular ontologies. SWiM is capable of editing OMDoc documents, including notation definitions that determine how they will be published. Integrating translations of OWL ontologies implemented in OMDoc from and to their standard RDF representation (cf. [sections 8.1.3](#) and [8.2.2](#)) would allow, e.g., for authoring the integrated documentation of an ontology in SWiM while performing other edits on the same ontology with a specialized OWL editor such as Protégé.

The OMDoc→RDF translation should not just be performed in an export filter, but there is also a great potential in feeding the resulting RDF triples into the triple store built into SWiM. That obviously makes an ontology browsable and queryable inside SWiM. Additionally, some ontologies – the SIOC argumentation ontology and subsets of the OMDoc and OpenMath CD ontologies – have the role of *system ontologies* for SWiM, as they influence its operational behavior. Making them editable provides a community with an entirely new way of tailoring SWiM to their needs.<sup>30</sup> The usability evaluation of SWiM, presented in [chapter 10](#), has identified concrete situations where that would help. [Section 10.7.3](#) discusses the impact of customizable system ontologies on usability.

<sup>30</sup> IkeWiki already features an editor for RDFS and OWL ontologies, but the ontologies relevant for collaboration on mathematical knowledge cannot completely be implemented in OWL, as discussed in [section 4.1](#).

#### 9.6.4.2 Improving Change Management

By exploiting dependency information from the RDF graph to avoid unnecessary, expensive re-renderings of pages (cf. [section 9.3.2.4](#)), SWiM gives a first proof of concept of change management. A backend with more powerful change management capabilities<sup>31</sup>, combined with an adequate user interface, could help to generalize this to other types of knowledge items, for example:

**Detecting and Reporting Changes:** A more intelligent differencing algorithm can help a system to distinguish changes that affect the semantics from those that don't (cf. [Mühob, chapter 5]), and thus only proceed to further steps if the semantics of a knowledge item has changed. It can also improve the way differences are presented and explained to the user, compared to the current state of line-wise text diff in Subversion and purely syntactic XML diff in IkeWiki. Concrete situations that could use better change detection are log message generation for metadata edits (cf. [section 9.3.2.2](#)) and finding out whether a notation definition has really been changed (cf. [section 9.3.2.4](#)).

**Change Impact Analysis:** The notation-specific change management workflow explained in [section 9.3.2.4](#) is currently hard-coded into SWiM; a more sophisticated backend could automatically take into account *all* dependencies defined in the underlying ontology.

**Adjusting to Changes:** Whereas changes to notation definitions are propagated non-interactively by re-rendering all affected documents, managing other kinds of changes may require user interaction. Changes to logical or functional structures, such as the definition of a symbol on which other symbols depend, typically rather affect the well-formedness of depending items; therefore, the system has to assist the user in keeping the knowledge collection valid. When *B* depends on *A* and the user Alice is about to change the semantics of *A*, she could be warned that that action might break the well-formedness of *B*, or precluded from changing *A* altogether. Conversely, if Alice insists on changing *A*, Bob, the next user to edit *B*, could be advised on how to adapt *B* to accommodate for the changes in *A*. Given that SWiM stores old revisions of knowledge items, an alternative might be that all links relevant w.r.t. dependency are not pointed to the latest version of the target knowledge item – which may change –, but to a fixed revision, and that the system would assist users in semi-automatically updating such links, always taking well-formedness into account. We have sketched this workflow in more detail in [LKo8]; it constitutes a compromise between the flowing nature of a wiki and Logiweb's immutability paradigm mentioned in [section 9.5.5](#).

#### 9.6.4.3 Improving Argumentation and Problem Solving

SWiM's proof-of-concept problem solving assistance focuses on individual knowledge items; more complex tasks such as refactoring have not yet been considered. Tasks like, for example, splitting one theory into two, or rewriting all axioms in a theory that involve description logic to first-order logic, not only require a discussion to have more than one subject – and a forum

---

<sup>31</sup>It is planned to extend the TNTBase database into that direction, by incorporating the locutor change impact analysis mechanism [AM10; Mühob] (personal communication with VYACHESLAV ZHOLUDEV, June 2010).

user interface that supports such discussions<sup>32</sup> –, but also new user interfaces for assisting with workflows that are no longer tied to individual wiki pages. Support for actions involving multiple pages is still rare in wikis; inspiration can be found in a few existing plugin, such as the Replace Text extension to MediaWiki, which globally searches and replaces text patterns [KL]. Conversely, SWiM currently lacks support for discussing knowledge items below the level of wiki pages, such as a single proof step, or a subterm of a mathematical object. This has partially been explored in related environments, such as the panta rhei browser discussed in [section 2.1.8.1](#), or the discussion and feedback services developed for JOBAD (cf. [section 7.7](#)).

Another potential that SWiM leaves unused so far is the problem solving *experience* represented in archived argumentative discussion threads (cf. [section 2.1.8.2](#)). Besides utilizing them for educational purposes, as the related SharedHCONE system does (cf. [section 9.5.7](#)), SWiM might use case-based reasoning to make participants of ongoing discussions aware of steps that other users have taken in previous discussions of similar problems.

#### 9.6.4.4 Potential Reimplementation on top of KiWi, TNTBase, and JOBAD

At the time of choosing it as a foundation for SWiM, IkeWiki was a good choice, as argued in [section 9.2.1](#). While IkeWiki has now been discontinued, the first stable version 1.0 of its successor KiWi [SEG+09] has been released in October 2010. This section discusses one possible direction of “reviving” SWiM: a reimplementation on top of KiWi, tentatively called SWiM<sup>K</sup>. Actually, we have, however, started pursuing another alternative path. Due to its even larger reuse of technology presented in this thesis, it will be discussed in the final chapter (cf. [section 11.3.1](#)).

The following enhancements and improvements that KiWi offers over IkeWiki are attractive for SWiM<sup>K</sup>:

- the ability to *annotate fragments* of wiki pages (cf. [section 9.1.2](#)). This may better suit the fine-grained structures of mathematical knowledge, but it might still not be adequate to the ubiquity of fine-grained markup.
- an *RDFa editing* plugin for the TinyMCE editor. This facilitates annotating mathematical knowledge items with arbitrary metadata
- the ability to *rate wiki pages*. This is a low-impact complement to argumentative discussions, as discussed in [section 6.6.2](#).
- a notion of *transactions* that will allow for committing several related changes at once [SEG+09] – for example a set of changes resulting from a refactoring, as discussed above in [section 9.6.4.3](#)
- *versioning of metadata* [SEG+09], whereas IkeWiki only versions pages. This is not so crucial for SWiM<sup>K</sup>, as the metadata are originally embedded into pages as semantic markup – and thus versioned anyway – and only extracted into the RDF triple store for compatibility with

<sup>32</sup>More such situations, where multi-subject discussions would have been desirable, occurred during the evaluation of SWiM for maintaining OpenMath CDs (cf. [section 10.5.5](#)). DANIEL SUTHERS and JUN XU made similar observations when designing the Kūkākūkā e-learning environment for “*artifact-centered discourse*” [SX02]; however, they refrained from supporting multiple subjects to prevent discussions from digressing from their original topic.

RDF-based services. However, not all semantic markup languages are capable of embedding all desirable metadata; the OpenMath CD language, for example, does, in contrast to OMDoc, not support RDFa, but still one might want to express metadata such as the author of a CD.

- a customizable *rule-based querying and reasoning engine* for the KWQL query language (cf. [section 9.1.3](#) and [\[BW10\]](#)), plus a visual query editor and a user interface for faceted search [\[SEG+09\]](#). This possibly makes queries more accessible to non-experts than IkeWiki's inline SPARQL queries were, and it may enable more reasoning potential in the wiki, if a relevant subset of the OMDoc and OpenMath CD ontologies can be implemented in KiWi rules.
- *linked data compliance* of the published wiki pages [\[SEG+09\]](#). This facilitates integration with external clients or with JOBAD client services that are loosely integrated into published wiki pages but do not have special knowledge about KiWi's storage backend.
- a “*community equity*” mechanism [\[Ora; BEK+09\]](#), which measures the social weight of user contributions. This is likely to be useful in all collaborative settings and can possibly be extended by a valuation of mathematics-specific types of contributions, such as the further formalization of an existing knowledge item.
- a *dashboard* that gives every registered user a personalized overview of recent changes at a glance [\[SEG+09\]](#)
- *recommendation of related content* based on tags [\[DD09\]](#). This could possibly be specialized to a service that recommends prerequisites to learners, as I have envisaged for SWiM in [\[Lano7b\]](#).

A mere port of SWiM to KiWi would not yet improve the connection to file-based repositories. Replacing the XML document storage of KiWi by the fully Subversion-compatible TNTBase database would give SWiM<sup>K</sup> full access to the history of the repository. As TNTBase also enables fine-grained access to XML fragments (without versioning them, however; cf. [section 8.3.4.1](#)), it would also do away with the necessity of importing files from a repository into a SWiM “working copy” and splitting them into suitable fragments. In the IkeWiki-based SWiM, the latter mechanism slows down the performance of edits, but the physical nature of the split-off fragments also entails usability problems. While browsing and editing *existing* fragments only requires understanding how to navigate along whole→part links, which the usability evaluation of SWiM for OpenMath CDs confirms to be feasible (cf. [section 10.5](#)), *adding* a new fragment, e.g. adding a symbol definition to a CD, is non-trivial. Moreover, adding a complete CD is not currently supported at all. Both are important use cases in an OpenMath context; authors need to create new symbols or CDs when existing CDs do not cover the mathematical concepts of interest sufficiently deeply or rigorously [\[DL08; AEB07\]](#). TNTBase would also facilitate the integration of alternative storage backends. There is, for example, a post-commit plugin that feeds extracted RDF triples into a local installation of the powerful OpenLink Virtuoso engine (cf. [sections 6.5.2.2](#) and [8.1.5](#)). A close integration of the wiki frontend with a repository backend, however, also entails technical challenges, one of them being a unified permission management.

Finally, integrating JOBAD into SWiM<sup>K</sup> would increase the supply of assistive services, including functionality provided by web service backends other than those of SWiM<sup>K</sup> itself. As an example,

where the combination of a TNTBase repository and JOBAD user interface extensions could provide a valuable new service with low impact on the SWiM<sup>K</sup> codebase, consider formula search. This functionality is missing from the old SWiM; there are only RDF queries, which merely cover occurrences of symbols in mathematical objects (cf. [section 3.2.2.4](#)), and full-text search<sup>33</sup>. TNTBase could feed the index of a formula search engine, such as MathWebSearch (cf. [section 6.5.1](#)), and a JOBAD client service could provide the user interface, as envisaged in [section 7.7](#).

## Acknowledgments

Parts of the review of the state of the art of [semantic] wikis given in [sections 9.1.1](#) and [9.1.2](#) are loosely based on my master's thesis [[Lano6a](#)] and an updated technical report derived from the former [[Lano7a](#)]. The review of the state of the art of argumentative discussions in wikis is derived from my contribution to [[LHCo8](#)]. The justification for developing SWiM on top of IkeWiki, which is summarized in [sections 9.2.1](#) and [9.2.2](#), was originally given in [[Lano7a](#)] and in [[LMRo8](#)]. The requirements for managing a large-scale formalization effort with SWiM, as described in the latter publication, were established together with SEAN McLAUGHLIN and FLORIAN RABE. The requirements for maintaining OpenMath CDs with SWiM (cf. [section 9.2.3](#)) were contributed by core members of the OpenMath community, including DAVID CARLISLE, JAMES DAVENPORT, MICHAEL KOHLHASE, PAUL LIBBRECHT, and CHRIS ROWLEY. Parts of the description of the SWiM architecture in [section 9.3](#) are based on earlier publications [[Lano8a](#); [Lano9b](#); [LHCo8](#)]. JAKOB ÜCKER helped with testing SWiM's Subversion client and the translations it performs (cf. [section 9.3.2.3](#)).

---

<sup>33</sup>The Lucene [[Apac](#)] full-text index is not currently populated with the text content of wiki pages that contain semantic markup, but that would be easy to realize.



## Usability Evaluation of an Integrated Environment for Maintaining Semiformal Collections

This thesis has so far focused on *enabling* the integration of heterogeneous services for managing semiformal mathematical knowledge – by representing the latter in a mutually comprehensible way and translating between different representations as needed. An environment that integrates such services and exposes them to end users has to be *usable*. Methods for evaluating the primitive services introduced in [chapter 6](#) are known in principle, but integrating such services into a coherent environment, which should assist with complex workflows, creates a new level of complexity. This chapter explores the challenges in making integrated environments usable.

I have conducted a small-scale usability evaluation of the support that the SWiM wiki introduced in [chapter 9](#) offers with three workflows that typically occur when managing collections of semiformal mathematical knowledge: quickly fixing minor errors, changing the presentation of something without changing its semantics, and peer review and preparing major revisions by discussion. Concretely, this has been done in the context of maintaining the official and contributed OpenMath CDs (cf. [section 2.4.3](#)), a realistic and practically relevant collection.<sup>1</sup> The evaluation has been done in three complementary steps: analyzing user-generated content w.r.t. its annotations, surveying the OpenMath community about the perceived utility of the wiki and their satisfaction with it, and conducting supervised experiments with test persons who had not used the system before, in order to assess its learnability and effectiveness.

[Section 10.1](#) provides a quick overview of how the OpenMath wiki has been prepared for the evaluation. [Section 10.2](#) explains the evaluation hypotheses and method, also reviewing methods that have been employed for evaluating existing semantic web or MKM systems. [Sections 10.3](#) to [10.5](#) report on the three evaluation steps. [Section 10.6](#) summarizes the results of the evaluation and interprets their impact on maintaining OpenMath CDs as well as the usability of integrated en-

---

<sup>1</sup>This chapter henceforth refers to the OpenMath CD installation of SWiM as “the OpenMath wiki”.



vironments in general. [Section 10.7](#) concludes and gives an outlook to further possible evaluations and means of improving usability.

## 10.1 Preparation and Setup

The evaluation of SWiM's support for the OpenMath CD maintenance workflows has been prepared as follows: Particular requirements for enabling SWiM to manage OpenMath CDs have been established together with the OpenMath community, as listed in [section 9.2.3](#). [Section 9.3](#) explains how SWiM has been designed to meet these and other requirements, and how its installation at <http://wiki.openmath.org> has been configured, particularly including the connection to the Subversion repository hosting the CDs. [Section 9.4](#) explains how support for the three concrete CD maintenance workflows has been realized in SWiM.

## 10.2 Evaluation Hypotheses and Method

This section defines usability and explains how I have evaluated the usability of SWiM for maintaining OpenMath CDs. Neither has usability evaluation played an important role in previous MKM research, nor has a specific methodology for evaluating the maintenance of knowledge collections in semantic wikis been developed so far. Deeper research on usability and evaluation methodologies has been conducted in the related field of digital libraries [[FTA+07](#); [Kru09](#)]. Not only have digital libraries been used for mathematical knowledge before (cf. [section 1.4.3.1](#)), but they also conceptually intersect with semantic wikis w.r.t. data storage, information retrieval, document presentation, navigation, and possibly annotation and commenting. For evaluating SWiM, I have therefore combined individual techniques for evaluating digital libraries, semantic wikis, and MKM services. [Section 10.2.1](#) defines usability and locates it within an overall model for evaluating digital libraries, [section 10.2.2](#) reviews evaluations of related systems from the fields of digital libraries, semantic wikis, and MKM. [Section 10.2.3](#) states my hypotheses about the usability of the OpenMath wiki, and [section 10.2.4](#) explains the methods I employed to prove them.

### 10.2.1 The Interaction Triptych Model and a Definition of Usability

The interaction triptych model by NORBERT FUHR et al., shown in [figure 10.1](#), comprises the components *user*, *content*, and *system*, connected via the three axes of *performance* [of the system managing the content], *usability* [of the system for the user], and *usefulness* [of the retrieved content for the user], along which a digital library can be evaluated [[FTA+07](#)]. This model is applicable to semantic wikis, except that its usefulness axis primarily considers content that the user consumes, but not the usefulness of content that the user *produces*, i.e. contributes to the library. FUHR et al. acknowledge that “*in order to fulfill all requirements on it, a DL [digital library] system must be the result of the integration of a number of different components*” [[FTA+07](#)]<sup>2</sup> and that “*the individual evaluation of such components can be carried out following evaluation standards already available in the literature*” [[FTA+07](#)], whereas “*an important issue is the evaluation of how*

---

<sup>2</sup>Here, “component” solely refers to components of the *system*.

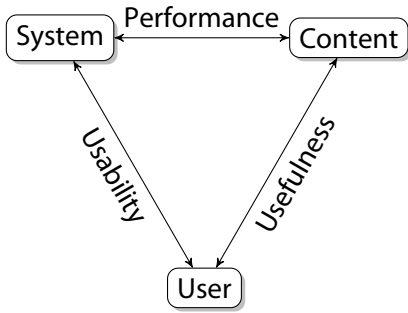


Figure 10.1: The Interaction Triptych Model [FTA+07]

ISO 9241-11	SHNEIDERMAN	NIELSEN
Efficiency	Speed of performance Time to learn	Efficiency Learnability
Effectiveness	Retention overtime Rate of errors by users	Memorability Errors/Safety
Satisfaction	Subjective satisfaction	Satisfaction

Table 10.1: Definitions of usability in ISO 9241-11 [Iso], by BEN SHNEIDERMAN [SP10], and JAKOB NIELSEN [Nie93]; comparison cited from [WVE99]

the individual components interoperate inside a DL” [FTA+07]. However, they do not provide guidelines on how to evaluate the integration.

In terms of the interaction triptych model, my evaluation of SWiM’s support for maintaining OpenMath CDs focused on the usability axis, taking a user’s point of view. Usability is commonly defined as “the extent to which a product can be used by specified users to achieve specific goals with effectiveness, efficiency and satisfaction in a specified context of use” [Iso], where effectiveness is defined as the “accuracy and completeness with which users achieve goals” [Iso], efficiency by the “resources expended in relation to [that]” [Iso], and (subjective) satisfaction as the “comfort and acceptability of use by end users” [Iso]. JAKOB NIELSEN and, similarly, BEN SHNEIDERMAN, have broken down effectiveness and efficiency, as cited in table 10.1. Some researchers have added utility as a further goal of usability [SRP07].

Usability evaluation must not confine itself to the user interface. FUHR et al. emphasize that “usability studies not only the surface representation of a system, but also its underlying structures” [FTA+07]. Similarly, JACCO VAN OSSENBRUGGEN et al. observed “strong dependencies between the user interface quality, the data and the underlying search and inference software” [OAH08] of semantic web applications. These aspects are of particular relevance here, as SWiM serves as a framework for integrating different services that access the content of a repository in an XML or RDF representation, and a considerable effort had been invested into designing the vocabularies = ontologies for the latter.

### 10.2.2 Usability Evaluations of Related Systems

Common usability evaluation techniques include assigning tasks to test users, analyzing the quality of the content created during these tasks (if any), and obtaining additional feedback – e.g. about users’ subjective satisfaction – from questionnaires or interviews. Analyzing the quality of created content is a special case of indirectly observing users by an *interaction log* [SRP07] – an evaluation approach that is generally suitable for interactive systems. This section briefly reviews how such techniques have been applied in usability evaluations of systems related to the OpenMath wiki – social/semantic digital libraries, semantic wikis, and other MKM environments.

### 10.2.2.1 Social and Semantic Digital Libraries

Building on the interaction triptych model, SEBASTIAN RYSZARD KRUK has evaluated the usability of JeromeDL – a semantic digital library that integrates services for collaborative faceted navigation and for social semantic collaborative filtering, as well as a number of other social and semantic services – in comparison to non-semantic systems [Kru09]. Besides the immediate quality and accuracy of the answers provided by these services and the users’ satisfaction with them – evaluated by assigning complex question-answering tasks to users, analyzing the results retrieved by individual searches w.r.t. the standard information retrieval metrics of recall, precision, etc., and assessing satisfaction with a questionnaire –, KRUK has evaluated the impact of these services on long-term memory retention by asking the test subjects some of the original questions once more, without offering help from the digital library. While accomplishing the overall tasks required combining different semantic services – as with the CD maintenance workflows in the OpenMath wiki –, their *integration* and any usability issues arising from that were not in the focus of the evaluation.

JILL FREYNE et al. have conducted a preliminary usability evaluation of services for social search and social navigation/browsing on top of a digital library [FFB+07]. In the first phase, a control group of users used the system with all social cues disabled in order to populate the “community wisdom” database. With social cues based on that information being enabled, the experimental group of users then had to find as many articles about a given topic as possible within a limited time. These users then had to answer a questionnaire, giving subjective feedback on how useful they found the individual social cues. Data collected during searches of other users turned out to be most useful for browsing, and vice versa, which proved the benefit of integration.

### 10.2.2.2 Semantic Wikis

This section briefly reviews usability evaluations of semantic wikis – first two that covered use cases similar to those that are of interest in this thesis, then two others. None of the evaluated systems does, however, integrate heterogeneous services to an extent comparable to SWiM.

CHRYSOVALANTO KOUSETTI et al. have evaluated the usability of an SMW extension that recommends types for classifying wiki pages in order to foster convergence in collaborative ontology engineering [KMH08]. Both in a wiki installation seeded with some initial structured knowledge and in one without, they analyzed the cohesion of the knowledge graph that test users had created over several days, and the extent to which types had been reused, and they obtained additional user feedback via structured interviews. KOUSETTI’s setting is in principle relevant here – recall the structural similarities with ontologies pointed out in [section 4.3.1](#)! However, with the official and contributed OpenMath CDs, created by a small community and having few topical intersections, the specific aspect of divergent evolution has never been an issue so far. FREDERIK PFISTERER et al. have evaluated the usability of an SMW extension for annotating unstructured text and an assistant for formulating inline queries [PNJ+08]. For both features, test users were assigned concrete tasks they had to accomplish – as many as possible within a limited time. The researchers obtained their results from analyzing the correctness of the annotations and queries created, and from a questionnaire. Annotating unstructured text and formulating inline queries are relevant

tasks within the larger context of creating, formalizing, and organizing mathematical knowledge, including OpenMath CDs, but my evaluation has not particularly focused on them.

Notable evaluations of other aspects of semantic wikis have been conducted by PETER HAASE et al. [HHM+09] and by TOBIAS KUHN [Kuh09]. HAASE et al. have evaluated the usability of a search extension for SMW by asking test users to answer given questions within a limited time [HHM+09]. The researchers obtained their results from analyzing the system's log files and from a questionnaire. KUHN has evaluated the ability of non-experts to create a formal knowledge base in a wiki using controlled natural language [Kuh09]. He analyzed the ratio of correct vs. incorrect sentences created, as well as the time a user spent per sentence.

### 10.2.2.3 MKM Services and Systems

In the MKM domain, usability evaluation is still rare. Among primitive services, content markup formula editors have most frequently been evaluated for usability [MEC+06; KKK+08; KLS+10]. Evaluation methods included performance testing – for example how many key presses or clicks users need for creating a given formula – and questionnaires assessing subjective usability. The pedagogical support of the ActiveMath integrated e-learning environment has been evaluated for usability [MWA03]. However, that publication does not describe the method of evaluation, but rather discusses some qualitative results. In particular, the integration of different services in ActiveMath (cf. section 7.1) was not in the focus of that evaluation. WILLIAM H. BILLINGSLEY has evaluated the usability of a graphical notation for mathematical objects and proofs, consisting of composable tiles, for students submitting solutions to exercises [Bilo8]. Test subjects were asked to prove a given set of theorems; BILLINGSLEY analyzed their solution attempts and classified the mostly unstructured feedback they had given by features of the software and aspects of the mathematical domain studied. Furthermore, he evaluated the notation against a cognitive dimensions of notations questionnaire (cf. section 8.1.5).

### 10.2.3 Hypotheses about the Usability of the OpenMath Wiki

The goal of evaluating SWiM's usability was to find out whether users with basic knowledge of OpenMath CDs can successfully accomplish maintenance workflows using SWiM, whether they understand the knowledge model that SWiM assumes in order to support these workflows, and to study the impact of the integration of heterogeneous primitive services, which enables workflow support, on usability. The evaluation particularly focused on the three typical maintenance workflows introduced in sections 6.1.2.1 to 6.1.2.3 and SWiM's assistance with them (cf. section 9.4).

A list of concrete hypotheses follows, ordered by workflow, where the term “effective” refers to the usability criterion of effectiveness (cf. section 10.2.1), and the terms “informing”, “understanding”, and “comprehensible” refer to learnability.

1. **Quickly Fixing Minor Errors:** When a user has spotted a minor error in [some part of] a CD, (a) the knowledge model and the user interface effectively support him in fixing this error locally by editing exactly the affected knowledge (sub)item or metadata field. (b) The user interface informs the user of the granularities at which he can edit a CD, and (c) it makes the exact change retraceable afterwards, in a comprehensible way.

2. **Fixing and Verifying Notations:** When a user has identified a mis-rendered symbol in some mathematical object, (a) the system utilizes the connection of the human-readable rendering of the object to the underlying semantic structures in such a way that it effectively enables the user to navigate in a straightforward way to the exact place where the underlying error has to be fixed, i.e. to the notation definition of the symbol. (b) The user interface enables the user to understand that the knowledge model treats the appearance of a symbol separately from its (formal) definition, and how both of these are connected. (c) The user interface effectively enables the user to fix the error by editing that notation definition and (d) to verify immediately whether that change fixed the problem.
3. **Peer Review and Preparing Major Revisions by Discussion:** When a user has identified a problem with a knowledge item, or wants to contribute to an ongoing discussion about a knowledge item, the system supports him in verbalizing his concern in a focused way. Concretely, (a) the user interface and the underlying knowledge model allow him to write a discussion post that is associated exactly to the knowledge item in question. (b) The knowledge model captures the argumentation primitives that users most commonly employ in discussions in the respective domain. (c) The user interface informs the user, in a comprehensible way, of what primitives are available in the current situation, and (d) effectively supports him in choosing the right one.

The following additional “meta-hypothesis” covers the system as a whole:

4. The assistance offered by the system (a) satisfies its users and (b) is considered useful by the OpenMath community.

#### 10.2.4 Techniques Applied for Testing the Hypotheses

To prove or refute the hypotheses given above, I combined different techniques known from the evaluation of digital libraries, semantic wikis, and MKM services (cf. [section 10.2.2](#)):

**Content Analysis:** Quantitatively analyzing the classification of discussion posts generated by five expert users helped to test hypotheses [3a](#) and [3b](#) from a knowledge model point of view. Sufficient amounts of content for testing the other hypotheses that way, e.g. by analyzing revision histories, had not been created.

**Community Survey:** To gather general feedback from the OpenMath community on all four hypotheses, I developed a questionnaire, which eleven people answered. As that encouraged few users to run through the three workflows, the most useful feedback gathered in the survey concerns [hypothesis 4b](#).

**Supervised Experiments with Test Users:** Finally, to test all hypotheses from both a knowledge model and a user interface point of view, particularly focusing on learnability, I conducted supervised experiments with 14 test subjects who had not used the system before. While running through each of the three workflows covered by hypotheses [1](#) to [3](#), as well as using further features of the system, they thought aloud, and afterwards gave additional free-form feedback.

Table 10.2: Matrix of evaluations carried out vs. hypotheses they help to test

▼ tests ►	Minor Fixes			Fixing Notations				Discussion				Satisf./Utility	
	1a	1b	1c	2a	2b	2c	2d	3a	3b	3c	3d	4a	4b
Content Analysis	(not enough content available)							++ <sup>a</sup>	++ <sup>a</sup>	(not evaluated)			
Community Survey	○	– <sup>b</sup>	○	○	– <sup>b</sup>	○	○	○	○	– <sup>b</sup>	○	○	++
User Experiments	++	++	++	++	++	++	++	+ <sup>c</sup>	+ <sup>c</sup>	+ <sup>c</sup>	+ <sup>c</sup>	++	– <sup>c</sup>

<sup>a</sup> This evaluation only covered the knowledge model, not the user interface.

<sup>b</sup> Individual survey participants commented on learnability, but I did not generally consider a questionnaire useful for assessing learnability.

<sup>c</sup> In contrast to those users who contributed the discussion posts for the content analysis, the experiment test subjects had little experience with OpenMath CDs, and thus with real OpenMath-related discussions.

Sections 10.3 to 10.5 summarize the results from each evaluation carried out. The different coverage of the hypotheses by the evaluations – depicted in table 10.2 – justify an arrangement by evaluation technique. Section 10.6.1 gives an overall summary of the evaluation results by hypothesis.

## 10.3 Quantitative Content Analysis of Argumentative Discussions

We have evaluated hypotheses 3a and 3b, stating that the knowledge model of the OpenMath wiki<sup>3</sup> supports focused discussions, by analyzing discussion posts w.r.t. the granularity of the subject they referred to and their argumentative type.

Initially, we manually imported a legacy corpus of relevant e-mail conversations into the wiki, structuring them as appropriately as possible. These e-mails had been written by four OpenMath experts in an early phase of preparing the OpenMath CDs for MathML 3 (cf. section 2.4.2). This resulted in 66 posts – a seed for attracting further, user-contributed posts.<sup>4</sup> Three experts from the original group plus one additional one contributed 24 further posts in the wiki. Except for noticing the per-type reply buttons and having received a brief introduction e-mail, these users were not familiar with the argumentation model.

A breakdown of these figures by argumentative type and by subject granularity follows:

**by type:** 69 posts fit into one of the types from the argumentation ontology, mainly *Issue* (48) and *Idea* (10). Only counting the 24 posts contributed by the users themselves, the result is slightly less convincing; for 9 posts the users were not sure how to classify them. The post type missed in most cases was not strictly argumentative: the *question* – either a direct question about some concept from a CD, or a follow-up question on an argumentative post, such as “what do you mean by this issue description?”. Further posts that could not be uniquely classified both raised an issue and proposed a solution (= idea) in the same sentence;

<sup>3</sup>i.e. the SIOC argumentation module, without the mathematics-specific extensions yet

<sup>4</sup>Seeding is a common wiki pattern [Mad+], which has also been employed to prepare evaluations of wikis, e.g. the one by KOUSETTI et al. mentioned in section 10.2.2.2.



possible ways of modeling that are discussed in [section 3.6.3.2](#) (model) and [section 6.6.5](#) (user interface).

**by granularity:** 36 posts – but only posts taken from the e-mail corpus – had individual symbols as their subject; the remaining 54 posts – including all user-contributed posts – were made on the CD level. This shows that either the users did not find it intuitive, or not necessary, to access subparts of a CD when they saw a complete CD in the browser, or that it was not possible to identify individual symbols a post referred to. The latter is the case with certain posts that argue on design issues of a CD in general, sometimes naming certain individual symbols as examples. A few other posts from the e-mail corpus referred to *two* closely related symbols each; we handled them by filing a copy with each affected symbol.

This analysis does not tell whether the problems that users encountered, e.g., with expressing multiple argumentative statements at once, root in the model, or in the user interface. The two following evaluation steps therefore complement the content analysis. The overall results obtained for hypotheses [3a](#) and [3b](#) are summarized in [section 10.6.1](#).

## 10.4 Community Survey

To gather feedback from a larger number of OpenMath community members on the utility of the services offered by the OpenMath wiki, including its support for all three maintenance workflows, for themselves and for the community as a whole, I developed a questionnaire, which eleven people answered.<sup>5</sup> In addition to testing [hypothesis 4b](#), this also yielded feedback on the other hypotheses. A summary of relevant results follows; full details are listed in [appendix D.2](#).

### 10.4.1 OpenMath CD Experience and Practices

More than half of the survey participants had, as active members of the [om@openmath.org](mailto:om@openmath.org) mailing list, been involved into the initial requirements analysis and deployment of the OpenMath wiki and were therefore somewhat familiar with it. The others were members of the wider OpenMath community, who were familiar with maintaining CDs, but to whom the wiki was mostly new.

Looking up information about CDs, symbols, mathematical properties, etc., was reported as the most frequent task when working with CDs, followed by discussing and reviewing. About two thirds of the participants work on the official OpenMath CDs, followed by CDs developed by themselves or their colleagues. More than half of the participants know what in CD file to look up information most of the time; browsing a human-friendly presentation of the CDs is an even more frequently reported way. Although the wiki offers that service, it has rarely been used. Automated search methods are employed rarely to never. However, a majority of users had *tried* the wiki for browsing CDs and acknowledged the relevance of custom knowledge base queries.

Most participants create or edit CDs with general-purpose text or XML editors; however, some have made their favorite editor aware of the OpenMath XML schema. There were no significant

---

<sup>5</sup>Four of the fifteen actual participants had only answered the first two, very general questions.



differences between creating and editing CDs.<sup>6</sup> Personal communication and mailing lists are the preferred means of communicating about CDs. A few community members – who also participated in the survey – have used the wiki for that (cf. the content analysis in [section 10.3](#)), whereas further technical means have hardly been adopted, as discussed in [section 6.1.2.3](#).

### 10.4.2 Utility of CD Maintenance Workflow Supports

The participants received a one-page description of SWiM’s assistance with each of the three CD maintenance workflows described in [section 9.4](#) – quickly fixing minor errors, fixing and verifying notations, and peer review and preparing major revisions by discussion –, including screenshots and instructions. In contrast to the supervised usability experiments with test subjects (cf. [section 10.5](#)), the participants were not assigned *concrete* tasks. For each feature set supporting a workflow, they first had to judge on a five-point Likert scale (cf. [Opp92], cited in [SRP07]) its relevance for the OpenMath community and for themselves, and its perceived potentials of time saving, additional possibilities of working, and enjoyment. Then, they were asked for how many times they had already carried out the workflow in the wiki themselves, and how well every single step of it had worked for them.

The feature sets were judged rather favorably, with the perceived relevance for OpenMath ranking highest and the perceived time saving potential ranking lowest. The significance of these answers is, however, limited, as less than half of the nine respondents to these questions had actually used these features. The most common reason for not using them was that not being personally involved into such workflows.<sup>7</sup> For those three to five participants who had carried out [parts of] the workflows, the system had worked “moderately well” on average, even “quite well” for posting comments. More useful feedback resulted from free individual comments. Feedback concerning the wiki’s utility for the community ([hypothesis 4b](#)) is discussed here; feedback on hypotheses 1 to 3 on the learnability and effectiveness of the maintenance workflow support as well as [hypothesis 4a](#) on satisfaction is reviewed together with the results of the supervised usability experiments with test subjects in [section 10.5](#).

Worrying that the ease of performing minor edits might attract low-quality contributions, one participant suggested a moderation facility to review and then approve, revise, or reject changes.<sup>8</sup>

<sup>6</sup>I had asked two separate questions, as one could imagine specialized tools, such as converters or interactive assistants, for creating CDs. Conversely, some editing tools – including SWiM in its current state of development, as discussed in [section 9.6.4.4](#) – only allow for changing existing CDs.

<sup>7</sup>This unexpectedly low turnout roots in a change of the OpenMath 3 development agenda. The whole OpenMath wiki case study and its preparation, starting with studying the CD maintenance workflows, was based on the assumption that, in parallel to the work on MathML 3 [ABC+10], the OpenMath 2 standard would evolve into OpenMath 3, entailing a complete revision of the official CDs by those people who were members both of the W3C Math working group and the closer OpenMath community. Until late 2008, when the OpenMath wiki became operational, the OpenMath 3/MathML 3 draft CD collection was under active development, all three workflows studied here being carried out regularly. Then, the attention of the developers shifted towards finalizing the specification of MathML 3, whose semantics would eventually be based on the formal core of the almost unchanged official OpenMath 2 CDs, whereas the informal comments and notation definitions of the symbols would not be revised inside the CDs but in the MathML 3 specification, as the comment cited in [appendix D.2.5.1](#) explains. In late 2010, the agenda towards OpenMath 3 is still unclear.

<sup>8</sup>In personal communication, that participant explained that this comment referred to the OpenMath CD manager reviewed in [section 9.5.3](#), which offers such functionality.

Another participant perceived a general discrepancy between the “*fluid editorial style that tends to be adopted in a wiki*” and the strict editorial control required for the official OpenMath CDs, which are rather consistently edited by few central editors than growing organically. Indeed, typical wiki communities tend to be more optimistic about the behavior of contributors and the quality of their contributions than the core OpenMath community; compare Wikipedia’s principles “*assume good faith [of collaborators]*” [Wikioe] and “*be bold [to contribute]*” [Wiki0f]. Traditionally, wiki systems have enabled the maintainers of a site to cope with a large number of changes by providing an overview of recent changes and making it easy to revert unwanted changes, as pointed out in [section 9.1.1](#). More recent systems offer more preemptive control mechanisms, including MediaWiki’s flagged revision extension mentioned in [section 6.3.2](#) or IkeWiki’s powerful permission management. All respondents considered the latter feature relevant; it is actually in effect in the OpenMath wiki in a strict configuration explained in [section 9.3.1](#). As that exactly reflects the traditional practice of managing and reviewing the OpenMath CDs, I consider it sufficient until more advanced moderation support will be introduced eventually.

Regarding argumentative discussions, three participants stated to prefer mailing lists. One respondent pointed out that discussions started on a mailing list, initially focusing on a topic not related to CDs, might eventually “*tangentially [touch] on CD issues*”, but then, the debaters would not want to switch into the different environment of a wiki.

### 10.4.3 Feedback on Other Wiki Features, and Wishes

The relevance of further miscellaneous SWiM features was generally rated high,<sup>9</sup> exceptions being the possibility to create custom queries (only 6 out of 9 respondents agreed), document-based CD editing (6/9), and form-based metadata editing (5/9). The disdain of custom queries might be related to a lack of awareness of the possibilities, even though all participants but one considered the sample queries prepared in the wiki relevant – a reproduction of the XSLT-generated symbol list at [openmath.org](http://openmath.org) [Oped]<sup>10</sup> and another similar query. In the requirements gathering phase, it had already become apparent that the OpenMath community prefers document-based editing of metadata to form-based editing (cf. [section 6.2.6](#)). About reasons for considering the document-based editor irrelevant, one can merely conjecture – taking into account the top relevance rating of the Subversion integration – that most users prefer to continue using traditional text or XML file editors while benefiting from other features that only the wiki can offer.

When asked to prioritize additional features or improvements of existing ones, the participants preferred usable support for adding new content – such as CDs or symbols –, dedicated support for a CD review workflow, and type-checking of mathematical expressions. Facilities for editing signature dictionaries<sup>11</sup>, formula search, and linking to discussion posts also ranked high.

<sup>9</sup> An additional question for how well these features had worked did not yield useful feedback, as too few users had used them.

<sup>10</sup> I have actually reproduced the symbol list in the wiki using a SPARQL query. This query is much more concise than the XSLT code that generates the traditional symbol list, and any wiki user would have had the permission to create or modify it. The only obvious disadvantage is IkeWiki’s suboptimal presentation of the query results.

<sup>11</sup> Support for signature dictionaries is prepared in the OpenMath CD ontology and in the SWiM editor, but the OpenMath signature dictionaries have not been in the focus of this case study and therefore have not yet been made available in the wiki.

#### 10.4.4 Conclusion on Utility

The OpenMath community generally confirms the utility of SWiM for managing OpenMath CDs ([hypothesis 4b](#)); this result is significant insofar as the survey participants have actually used the system. Experienced users even requested support for more complex workflows, such as reviewing and approving concrete changes proposed for a CD. When tools for certain tasks already exist, experienced users have usually got used to them; examples include editing CDs with general-purpose XML editors, and browsing the static CD pages on [openmath.org](http://openmath.org). The bar for convincing users of new approaches to these tasks is therefore set high. Although most existing tools merely accomplish single, primitive tasks in isolation from each other instead of integrated workflows, as discussed in [section 6.1](#), one might conclude from the replies about OpenMath experience and practices (cf. [section 10.4.1](#)) that experienced collaborators to a relatively maintainable collection do not need integrated support with complex workflows. However, one participant suggested fully integrating the unique features of the wiki, particularly the argumentative discussions, into the familiar environment of the existing CD pages, instead of making them available as yet another, separate system. Conceptually, this is exactly what the integration techniques introduced in [chapters 7 and 8](#) enable, SWiM just being one possible integration platform; practically, it would require a complete overhaul of the [openmath.org](http://openmath.org) site.<sup>12</sup>

### 10.5 Supervised Usability Experiments with Test Users

The community survey covered in the previous section has mainly yielded feedback about the (perceived) utility of the OpenMath wiki but little detailed feedback on the learnability of the user interface and its underlying knowledge model and its effectiveness in supporting the three CD maintenance workflows ([hypotheses 1 to 3](#)) and satisfaction ([hypothesis 4a](#)). Thus, I have conducted complementary hands-on experiments with 14 test users, who were directly observed in a controlled environment [[SRP07](#), [chapters 7.6 and 12.2](#)]. [Section 10.5.1](#) introduces the setting and evaluation method, [section 10.5.2](#) presents overall figures about the user feedback gathered and interprets them with regard to learnability. [Sections 10.5.3 to 10.5.5](#) discuss notable observations regarding [hypotheses 1 to 3](#) made when walking through the three maintenance workflows, whereas [section 10.5.6](#) reviews feedback related to the general (in)coherence of integration. Feedback referring to [hypothesis 4a](#) is reproduced in these sections in the context of its occurrence. [Appendix D.3](#) provides the logs of the experiments in full detail.

#### 10.5.1 Setting and Evaluation Method

The set of test subjects was disjoint with the participants of the community survey. None of the test subjects had used the OpenMath wiki before; some had seen or used SWiM in other settings. All users were familiar with one of OMDoc, OpenMath, or semantic wikis in general, few of them with more of these topics. Thus, they did not have the same level of background knowledge about CD management workflows as a member of the closer OpenMath community.

<sup>12</sup>I have actually provided preliminary links from the static XHTML renderings of the OpenMath 3/MathML 3 draft CDs to the wiki. Each CD and symbol is linked to its corresponding wiki page, asking visitors to discuss these topics there. However, these links do not replace a full integration of the wiki's *services* into the CD pages.

Concerning hypotheses 1 to 3, each test user was asked to perform the following concrete tasks:

1. **Quickly Fixing Minor Errors:** Change the description of a given symbol in a given sample CD – the *swim* symbol in the *swimtest* CD –, first using the document editor, then using the metadata editing form. Comment on the comprehensibility of the Subversion log. The experiment started on the top level of the CD.<sup>13</sup>
2. **Fixing and Verifying Notations:** Starting from a rendered object – the first and only mathematical property of the *swimtest#swim* symbol –, change the rendering of a given symbol (*swimtest#testop*) in that object.
3. **Peer Review and Preparing Major Revisions by Discussion:** Conduct a short discussion with yourself or other users on a discussion page of your choice.<sup>14</sup> Comment on the inline queries for topics with discussions on the entry page (described in [section 9.4.3](#)). Due to the limited experience of the test subjects with OpenMath CDs, this experiment did not cover an actual review or revision.

The experiments were carried out in the live OpenMath wiki to give the users the opportunity to browse the real CDs. The procedure for each task was as follows:

1. The user read the same one-page description of the respective SWiM feature that the community survey participants had been given (cf. [section 10.4.2](#)).
2. I gave additional explanations, e.g. about the OpenMath background, as appropriate, when people were not familiar with it, in a way adequate to their previous experience.<sup>15</sup>
3. The user tried to accomplish the task. I first let them explore the system and figure out what to do themselves, but when they got stuck or obviously tried a wrong approach, I gave hints pointing to the intended way of solving the problem.<sup>16</sup>
4. I asked the users to think aloud [[Nie93](#); [BRoo](#)]: What am I trying to do, how am I trying to accomplish it, what knowledge model do I have in mind, what do I think about the interface I'm using, how would I expect it to be. Previous research has proven thinking aloud suitable to “*reveal users' problems with respect to information processing*” [[KUo8](#)].
5. As appropriate, I asked the users questions about their perception and understanding of the user interface, roughly following the questionnaire from the community survey (cf. [section 10.4.2](#)). Particularly when users had severe problems or had performed an

<sup>13</sup>Here, and in the following task, each test user found the page as the previous user had left it behind. Comparing two successive experiments, the only differences were in the description text of the *swimtest#swim* and in the rendering of the *swimtest#testop* symbol. As there was no *conceptual* difference and the users were told that what they saw was just sample content, it can be assumed that these differences did not significantly influence the experiment.

<sup>14</sup>In most cases, there was already an existing discussion, started by real users or by previous test users.

<sup>15</sup>For example, to a user familiar with OMDoc, I introduced OpenMath CDs and symbol definitions as “similar to theories and symbols, but less formal”. To a user familiar with semantic web ontologies, I introduced them as “similar to ontologies and their entities (classes, properties, individuals), but more mathematics-oriented”.

<sup>16</sup>I considered that legitimate and not distorting the results, as the objective was not to measure success rates or performance.

unexpected action, I asked them for their comprehension of the underlying knowledge model. Thus, the experiments combined user observation with a semi-structured interview [FF94] (cited in [SRP07]).

Depending on how deeply users delved into the system and how much feedback they gave, including feedback beyond the tasks, one session took between one hour and one hour and a half.

From their previous knowledge and from the brief introduction to OpenMath CDs and the one-page feature descriptions they had received, the users could be assumed to have a basic understanding of the domain of knowledge and its structures – that a CD contains symbols, that CDs and symbols have metadata, that definitions of symbols can contain mathematical objects, which are again composed of symbols in content markup, that a symbol in an object has a presentation markup notation, and that for each wiki page there is a discussion page. I had neither made explicit that the wiki treats CDs, symbol definitions, mathematical properties of symbols, and notation definitions as knowledge items that correspond to wiki pages, nor that these knowledge items are connected via typed links, which are primarily accessible via the navigation tree.

Following general figures about the collected feedback statements in the following section – including actions as well as verbalizations –, a walk through the tasks with a discussion of relevant user feedback is given. Any figures are not meant to be interpreted strictly quantitatively; however, a problem explicitly verbalized by a large number of users can rightly be considered more serious than an irritation observed from few users. The latter are only reproduced here as far as I consider them relevant. Suggestions for improvement that are likely to result from later rationalizations of why something had not worked for a user ([Nie93, chapter 6.8]) have been filtered out.

## 10.5.2 Overall Figures

The 516 distinct feedback statements given by the users were classified using a customization of LINDA VAN RENS's scheme for categorizing think-aloud protocols [Ren97] (cited in [SRP07, chapter 8.4.2]) and then analyzed.

### 10.5.2.1 Types of Positive and Negative Feedback

Overall, the users gave more negative than positive feedback (306 vs. 224 statements) and made 182 suggestions for improvements beyond mere bug fixing.<sup>17</sup> However, the method of data gathering does not support the quantitative verdict that the overall system is rather badly usable. Positive feedback was almost balanced between *explicit positive statements* (93) and *actions or think-aloud verbalizations showing evidence that the user understood how to accomplish a task and succeeded in doing so* (95). The remainder was *evidence showing that the user understood a design concept or the knowledge model behind the user interface* (36).

Major types of negative feedback were *explicit statements* (61), *verbalizations showing evidence of confusion or uncertainty* about an aspect of the interface (52), *explicit statements about expectations not met* (51), *actions or verbalizations showing evidence that the user did not understand how to accomplish a certain task* (44), and *verbalizations showing evidence of dissatisfaction with an aspect*

<sup>17</sup>59 statements were classified into more than one category.



Figure 10.2: Types of positive (light) and negative (dark) feedback from test users

of the interface (43). In 19 cases, verbalizations showed *evidence that the user was surprised at the outcome of an action*. There were another 18 cases when a user wanted to perform an action that would have been right, but *unexpectedly encountered a known bug or unknown problem* in doing so, and finally 18 cases where the user's verbalizations showed *evidence of not understanding a design concept or the knowledge model* behind the interface.

### 10.5.2.2 The Importance of Understanding Concepts

Understanding or not understanding design concepts and the structure of the knowledge seems to be a marginal issue according to the figures given above, but a clearer inspection of the records indicates the opposite: Users who understood a concept were able to successfully accomplish tasks for which that concept was relevant (13 out of 14 single cases<sup>18</sup>). Conversely, users who did not understand a concept had difficulties in accomplishing related tasks (7 out of 7 single cases). An account of a few striking cases follows:

**Dependency Background→Easy Navigation:** A user with background knowledge about dependency relations in structured documents quickly understood the types of links between wiki pages and the way they were presented in the navigation tree. Having edited a notation definition of a symbol, the user navigated back to the original mathematical object with superior ease, exclusively relying on the navigation tree (cf. [appendix D.3.14](#)).

**Semantic Markup→Notation Editing:** A user with background knowledge about content vs. presentation markup, first opened a mathematical object in the document editor when asked to edit the notation definition of a symbol. Then, the user realized that the formula editor can only edit content markup, whereas the given task was to change the *presentation* of a symbol. Therefore, he or she left the editor and tried something else (cf. [appendix D.3.10](#)).

<sup>18</sup>The other cases where users had understood a concept were not followed by a relevant task.



**Ontology Misunderstood→Navigation Difficulties:** One user had a completely wrong understanding of the ontology when trying to navigate from a symbol to its notation definition. He or she expected a notation definition to be a *part of* a symbol and also thought that in the ontology there was a class for each operator, and then a subclass or instance for each of its renderings or occurrences. Therefore, the user found it hard to make any use of the links that were available in the navigation tree (cf. [appendix D.3.8](#)).

### 10.5.3 Quickly Fixing Minor Errors

#### 10.5.3.1 Navigating to the Editor for a Subpart of a CD

Getting from the top level of a CD to the editor for a symbol definition in order to change its description was no problem for most users; more than two thirds (9/14) immediately used the right “open this” link (cf. [section 9.3.3.1](#)) to open the symbol definition on its own wiki page and then edited that page. Such links were familiar to five users with MediaWiki experience (cf. [section 9.5.4](#)); however, three users had expected an “edit this” link directly leading to the *editor*. Five opted for in-place editing (cf. [section 6.2.7.2](#));<sup>19</sup> while these were mostly users who had not initially used the right “open this” link, I consider in-place editing a suitable complementary feature, which would not interfere with the “open/edit this” mechanism. Most of those who did not open the right part opened a different one instead. Three opened the mathematical property that the symbol had, which happened to be just below the description text; two did so because its “open this” link was closest to the cursor position. Three other users<sup>20</sup> directly opened the editor at the CD top level,<sup>21</sup> and three had trouble finding the “edit” tab.<sup>22</sup>

#### 10.5.3.2 The Document Editor

In the document editor, three users immediately spotted the “description” metadata field; there was no one who did not find it.<sup>23</sup> Few users gave a summary of their changes; four did not notice the summary field at all. As a remedy, they suggested a more instructive label (e.g. “change summary” instead of merely “summary”) or an initial placeholder text (e.g. “enter summary here”). Saving one’s edit was no problem in most cases.

One user explicitly remarked that “minor edit” does not necessarily mean “no semantic impact” (cf. the terminology in [section 6.1.2.1](#)). The “description” field, in particular, is semantically constitutive for an OpenMath symbol (cf. [section 2.4.3](#)). While the system could check whether such a constitutive metadata – actually, rather *data!* – field or a less important one has been

<sup>19</sup>for example a text box appearing on double clicking the description text

<sup>20</sup>including one user who was familiar with MediaWiki’s section editing but for that same reason had expected an “edit this” link for the section instead of “open this”

<sup>21</sup>I consider editing the top-level page an alternative that should be supported in any case. In the current implementation, the editor for a parent knowledge item merely displays the source code of XInclude links to split-off children (cf. [section 8.3.2.2](#)) but does not expose them as clickable links. The only way of getting to the editor for a subpart when already in the edit tab would be via the navigation tree, which is not quite obvious.

<sup>22</sup>Several users got distracted by IkeWiki’s ontology editing toolbox (cf. [section 10.5.6](#)). One user reported being most used to the wiki interface of Trac [[Traa](#)], where the “edit” button is at the bottom of the page.

<sup>23</sup>Four tried to edit the *labels* of metadata fields, which, albeit useful, should have been forbidden, as the translation employed for making metadata editable did not support it.



changed, it cannot generally distinguish a semantic change from a non-semantic one. With an additional check box in the editor, responsible users could indicate that.

### 10.5.3.3 The Metadata Editor

For editing the description of a symbol, most users first chose the document editor<sup>24</sup>; afterwards, I asked them to use the metadata editing form as well. Two users, having realized that a description is a metadatum, went into the metadata editing form first. Most users found that field easily; however, one was confused by additional metadata fields shown in the form that did not correspond to metadata present in the CD, such as *dc:creator* for the (wiki) user who had made the last change. One user missed the possibility to summarize a metadata edit; looking at the Subversion logs of metadata edits, others requested the same (see below). Another user missed metadata edits in the revision history of a page.<sup>25</sup> Most of the problems encountered in the metadata editing form were, however, due to shortcomings of its unchanged IkeWiki user interface.<sup>26</sup>

### 10.5.3.4 Understanding the Revision Log Messages

Eight out of the fourteen test users considered the revision log messages generated for the CD file helpful and immediately recognized their own changes.<sup>27</sup> Five wondered what a (no comment) message meant and only realized after an explanation that it reflected that they had not given a summary in the document editor.<sup>28</sup> Two users commented that the log only shows *that* – but not *how* – a *metadata* field had been changed; summaries for metadata edits (see above) could help to remedy that. The users generally understood that the wiki supported editing at a finer granularity than the file level, and that the revision log messages tried to reflect that.

One user remarked that fine-grained change tracking might not always be desirable, for example when applying a large change set with a common purpose, such as “revised according to the decisions of the last project meeting”. In a working copy of a repository, one would do that by first making all these changes locally, and then committing all affected files in one transaction. Thus, a suitable collaboration environment would also have to support transactions (cf. [section 9.6.4.4](#)), or *subsequent* annotation of changes already committed.

Several improvements to the way the log messages indicate granularity were suggested. Three users got confused by the technical terminology used for referring to the changed subpart (actually changed fragment ... and SWiM-internal URIs such as *cd:arith1+plus*; cf. [listing 9.1](#)) and suggested less technical descriptions, such as “changed example 1 for the symbol *plus* from the CD

<sup>24</sup>most likely because it is accessible via a tab labeled “edit”

<sup>25</sup>IkeWiki does not version metadata, as explained in [section 9.6.4.4](#). Besides versioning metadata in the database, a system has to display their revision history. The Semantic History SMW extension provides such a view of recent changes to metadata [[BDM09](#)].

<sup>26</sup>The most common problems were that the metadata fields were perceived as read-only, as there were only “delete” buttons (reported by three users), that they had to be opened for editing by double clicking into them, and that there was no “save” button, but, instead, one had to hit Return or click into another field.

<sup>27</sup>All test users were familiar with Subversion.

<sup>28</sup>The same had occurred to one participant in the community survey, who figured out the latter himself/herself.

*arith1*". Another user pointed out that the names of the metadata fields were different from the XML element names in OpenMath CDs and the labels in the published document.<sup>29</sup>

### 10.5.3.5 Navigating Back to the CD Toplevel

Having edited a metadata field of a symbol, some users wanted to navigate back to the CD.<sup>30</sup> Five users asked how to generally navigate to the parent page<sup>31</sup>, three would have expected breadcrumbs linking to previously visited pages.<sup>32</sup> Having navigated to the parent via the incoming *hasPart* link in the navigation tree – mostly after being instructed so –, two users suggested to unfold the most relevant relations in this tree by default – including a relation pointing to the parent.<sup>33</sup> For easily navigating to children of a complex resource, such as a CD, two users had expected a table of contents on top of the page, as one often finds in (non-semantic) wikis.

## 10.5.4 Fixing and Verifying Notations

### 10.5.4.1 Rendered Objects

This task started on a page where the users looked at a rendered object; some commented on that. Three users immediately understood that the buttons above a mathematical object (cf. [figure 9.1](#) on page 291, and [appendix C.1.2.1](#) for how they are generated) give access to views of the same object in different representations. Two users found the distinction of opened from closed views by button color (green vs. gray) helpful. One wondered whether the content markup representations of an object, as displayed in these views, were directly editable.<sup>34</sup> Three would have preferred a larger font for rendered objects, in order to be able to hit symbols with the mouse more easily.

### 10.5.4.2 Navigating to the Definition of a Symbol

The users had received instructions that they could navigate from a rendered symbol to its definition in a CD with the middle mouse button (cf. [appendix C.1.2.1](#)). Individual users suggested making that more obvious by displaying a hand cursor over symbols, which Firefox does not currently do on its own, highlighting symbols on hover, or other context-sensitive help. Two users asked how to open symbols with an invisible rendering, such as “invisible times”, which is not currently possible.<sup>35</sup> After middle clicking on a symbol, four users did not notice that the link target had been

<sup>29</sup>The revision log displays the RDF names, e.g. *dc:description*. This does not differ much from *Description*, but in other cases there would be a considerable difference, e.g. the OpenMath *Name* element being mapped to *dc:identifier*.

<sup>30</sup>This was not part of the task, but it should actually have been.

<sup>31</sup>One community survey participant reported the same problem.

<sup>32</sup>IkeWiki does offer breadcrumbs in its “history” toolbox. No one noticed them, probably as they are displayed in the lower left corner of the browser page. Secondly, a bug in the implementation of the “open this” links prevented them from contributing to the navigation history.

<sup>33</sup>Such an inverse relation of *hasPart* is not currently displayed in the navigation tree. [Section 10.6.3](#) discusses how that could be done.

<sup>34</sup>They are not, but it would be a good idea.

<sup>35</sup>Certain visible renderings also cause problems. Presentation MathML elements such as *m:mfrac*, which both renders the fraction stroke and contains its arguments as children, or the *m:mtable* rendering of the binomial coefficient shown in [listing 2.6](#), can carry an *@href* attribute, but Firefox propagates the link to all children of the fraction, overriding any nested links. The MathML specification does not define a semantics for nested links but advises

opened in a new browser tab.<sup>36</sup> Two users explicitly requested navigating to a symbol's definition by a left click.<sup>37</sup>

### 10.5.4.3 Navigating to the Notation Definition for a Symbol

Four users explicitly communicated that they had understood that the newly opened tab displays the definition of the symbol they had just clicked on. Getting from there to its notation definition proved easier. Six users found and used the incoming *rendersSymbol* link in the navigation tree without further help;<sup>38</sup> some found it because of its name, or because the link target had “notation” in its URI.<sup>39</sup> To three users it was generally unclear in what situations to use the navigation tree. Suggested improvements, each by three users, included: an *outgoing* link from the symbol definition to the notation definition, more intuitive link type labels,<sup>40</sup> and bringing the navigation tree more to the user's attention.

Some users tried – partly successfully – different ways of navigating from the original mathematical object to the notation definition of the symbol in question. Four first opened the mathematical property that contained the object. Two of them then tried to edit that page but realized that the formula editor is for content markup, whereas the task was to change the *presentation* of a symbol. One user studied the *usesSymbol* link going out of the mathematical property and identified the right symbol among them.<sup>41</sup> One even entered the rendering of the symbol as a character into the “search” box of the wiki.<sup>42</sup>

### 10.5.4.4 Preview of a Notation Definition

Five users gave explicit positive feedback when first viewing a rendered notation definition (cf. figure 6.10 on page 221). It was commonly criticized (by five users) that the source code of the rendering was not shown. One user additionally missed the content markup source of the expression rendered for the preview, i.e. the instance of the prototype with placeholder arguments. Probably due to such reasons, it was not clear to one user *why* the symbol was rendered as shown in the rendering preview column. Besides that, several users suggested making the presentation more intuitive, for example by explaining the “prototype” and “rendering” technical terms (two users)

---

against using them [ABC+10, section 6.4.3]. Any workable solution would require an in-browser script to interpret the link, be it given via *@href* or as an annotation.

<sup>36</sup>The middle click has an unfortunate double functionality here: It is the only way of opening a Presentation MathML link in Firefox, and on Unix-like systems, as on the computer used for the experiments, it opens the target of a link in a new tab. The latter was not clear to users of other operating systems.

<sup>37</sup>Meanwhile, similar functionality has been realized as a JOBAD service (cf. chapter 7) in the Logic Atlas [KMR].

<sup>38</sup>Some may have remembered it from the workflow description they read before the experiment.

<sup>39</sup>Such pseudo-XPath URIs are generated by default when splitting an imported CD (cf. section 8.3.2.2).

<sup>40</sup>IkeWiki uses *rdfs:labels* if available in the respective ontology, but none had been provided for the OpenMath CD ontology at the time of the experiment.

<sup>41</sup>That user was able to interpret the well-known names of the other symbols in the object, such as *arith1#times*, and concluded that the test symbol in question was none of them.

<sup>42</sup>That did not work, as the full-text index is not currently populated with the text content of wiki pages that contain semantic markup (cf. section 9.6.4.4). If it were, it would also have been questionable how to rank the results returned by a search for symbols used in mathematical objects: For example, should the notation definition be prioritized, or a document with many occurrences of the symbol?

as well as further elements of the notation definition markup language (one user), and making it more obvious that  $arg_1, \dots, arg_n$  are sample arguments in the preview of an  $n$ -ary operator, each of whose arguments is matched by `<expr name="arg"/>` (three users). One user requested more realistic previews by showing real mathematical objects from the knowledge base in which the symbol occurs. Finally, minor layout improvements were suggested; related to the semantics of the notation definitions were writing arguments  $arg_1, \dots, arg_n$  with subscripts (one user) and providing a third operand for an  $n$ -ary infix operator, so that the exact spacing with realistic arguments on both sides of the operator would become apparent (e.g.  $arg_1 \circ \circ arg_2 \circ \circ \dots \circ \circ arg_n$ ; one user).

#### 10.5.4.5 Editing a Notation Definition

In the editor, six users instantly figured out where to go for changing the operator symbol; however, two reported not having understood anything else in the editor. Overall, nine users complained about the extensive use of HTML tables, mainly arguing that they occupied too much space; one suggested making them foldable. One user who was familiar with the notation definition language (cf. [section 2.4.5.2](#)) would have preferred direct XML source editing altogether. Four users explicitly appreciated the two-column prototype|rendering arrangement (cf. [section 6.2.3](#)), whereas it confused two others. As they had seen a preview of the *rendering* – instead of its source – in the right column of the preview, they were not sure whether they would now have to *edit* the right column. Seven users wondered how to enter Unicode symbols. Only three discovered the  $\Omega$  tool button for inserting special characters. Suggested alternatives included editor support for XML entities such as `&compfn`; or `&#x2218`; (two users) or L<sup>A</sup>T<sub>E</sub>X macros such as `\circ` (one user). One user suggested logging what symbols were used most frequently when defining notations.<sup>43</sup>

#### 10.5.4.6 The Formula Editor

Some users tried the Sentido formula editor, either during this experiment, or during “quickly fixing minor errors”. The tool button for opening the Sentido dialog was not obvious to two users, even though it appears pressed when the cursor is inside a formula.<sup>44</sup> One user wondered how to reuse the symbol, whose notation he or she had just changed, in the formula editor.<sup>45</sup> One user requested a more direct access to the formula editor, without first passing through the document editor. Two users made notable comments about editing linear syntax: One suggested a multi-line input box, particularly for large formulæ. Another one wanted to be able to enter function symbols like the  $f$  in  $f(x)$  more easily ad hoc, i.e. without first defining them in a CD.<sup>46</sup>

<sup>43</sup>This would indeed make sense in a very large and dynamic knowledge base, where many new non-standard symbols are introduced, but less so for the official OpenMath CDs.

<sup>44</sup>This was a deliberate design decision (personal communication with ALBERTO GONZÁLEZ PALOMO, 2008-07-02).

<sup>45</sup>Arbitrary symbols can be used via the *OMS* construct of the linear syntax, but Sentido’s symbol palette is not currently extensible by new symbols defined in the wiki, as discussed in [section 6.2.8](#).

<sup>46</sup>By default, Sentido interprets such an input as “ $f$  invisible times  $x$ ”. Sentido allows for marking  $f$  as a “function variable”, but this possibility was not immediately obvious to the user.

#### 10.5.4.7 Support with the Overall Workflow

Three users explicitly approved of the support with the complete workflow. Four liked the instant preview after editing a notation definition. Three found the navigation from a mathematical object to the notation definition editor too cumbersome. Several users requested a more direct access to the editor. Two had expected that middle clicking on a rendered symbol would directly lead to the notation definition, not to the symbol definition. Two others expected a notation definition of a symbol to be directly editable within the editor for the symbol's definition. Most users (eight) wished for a local “change notation” button or context menu entry accessible from any rendered object; another one suggested integrating something similar into the (content markup) formula editor – which would indeed well contribute to the design goal of facilitating local minor fixes. However, as one user also pointed out correctly, changing a notation definition of a symbol potentially affects all other mathematical objects using that symbol as well, so one should be warned about the global consequences of such a local edit.

### 10.5.5 Peer Review and Preparing Major Revisions by Discussion

In the discussion forum, the users merely had to find their way and optionally write some posts. This section focuses on feedback related to the argumentation model and OpenMath CD maintenance. Frequent annoyances caused by IkeWiki's limited discussion forum interface<sup>47</sup> could be avoided with a more sophisticated forum system, such as the one described in [section 11.3.1](#).

#### 10.5.5.1 Understanding the User Interface

Three users familiarized themselves with the argumentation model by studying existing discussion threads. Four noticed the tooltips of the post and reply buttons and read some of them. Some users commented on the icons indicating the argumentative type of a post. Three found them helpful for getting an overview without reading lengthy explanations. However, some of the icons were not clear; three users suggested a legend. Two users wondered what “thumbs up” vs. “thumbs down” on a *Justification* post meant and requested additional text labels (“supporting” vs. “challenging”), so that understanding the post type would not exclusively rely on the icon. Conversely, two users who had found out that one can post challenging and supporting arguments did not initially understand that such posts would only be distinguishable from their icons.

One user explicitly expressed satisfaction with the supply of post and reply types; the availability of untyped posts was positively commented on twice. One user explicitly confirmed having understood that the availability of reply buttons depends on the argumentative type of a post, whereas another one explicitly admitted not to understand that. Three users expected the chosen post type to be displayed in the post composition dialog; they would actually have preferred to first compose their post and *then* think about its type before submitting. One user remarked that the default reply subject (“Re: (subject of the original post)”) does not make sense for an *Idea* replying to an *Issue*, as the title of a proposed solution is reasonably different from the title of the

---

<sup>47</sup>Six users complained about posts on the same level not being sorted by time. As the posts occupy a lot of space, three users suggested to make them collapsible; one user additionally opted for a reasonable preset, e.g. collapsing old threads by default. Other features commonly known from web discussion forums but missing in IkeWiki/SWiM were also requested, such as quoting other posts, deleting comments, and re-editing posts (one user each).

problem it addresses. Another user simply changed the subject manually for each of his or her posts. One user suggested that an administrator should be able to change the argumentative type of an existing post, in case it had a wrong or missing type.

#### 10.5.5.2 Writing a Discussion Post

Inside the editing dialog, two users expressed interest in writing mathematical objects that used symbols or notations not available in the formula editor – a realistic scenario when arguing about symbols or notations that do not yet exist in the knowledge base. A presentation markup editor would have been necessary for that purpose. Three users suggested an easier interface for votes, with ▲/▼ buttons on each post, instead of having to state one's position as a “heavyweight” post, and to display the outcome as a number.<sup>48</sup> Another two users did not mind the current interface for posting *Positions* for its consistency with the other argumentative types.

#### 10.5.5.3 Alternative Accesses to the Discussion Facility

Two users suggested alternative ways of accessing discussion posts: One suggested to shortcut the access to discussions about subparts of larger knowledge units by adding a link to the discussion forum of a subpart next to the respective “open this” link in order to save one click;<sup>49</sup> that would be a first step towards a closer integration of the discussion forums into the published CDs, as discussed in [section 10.4.4](#). The remarkable action of another user questioned the strict separation of discussion forums for parent and child knowledge items: The user intended to comment on a symbol but posted that comment in the discussion forum of the CD page. One justification was the inconvenience in navigating to the symbol definition via “open this”, but the actual reason was that the user had taken the potential interaction of one symbol with other symbols in the same CD into account.<sup>50</sup> As a partial remedy, the user suggested listing all threads about subitems of a CD in the CD-level discussion forum. Decoupling discussion threads from forums that are attached to a fixed knowledge item might be beneficial; the system could allow for freely associating posts to an arbitrary number of knowledge items, on whose discussion pages they would show up.

#### 10.5.5.4 The Argumentation Model: Understanding and Suggested Enhancements

Some users did not understand the argumentation model from the information given on the interface, whereas others did not find it appropriate for expressing what they wanted. Two users did not understand the difference between an *Argument* and a *Position*. On the other hand, three users made explicit that they understood it. Two users did not understand the difference between a challenging and a supporting argument. Another user did not understand the difference between an idea and a supporting argument, and one user used an *Elaboration* for what was actually an

---

<sup>48</sup>This is known from many web discussion environments, including MathOverflow (cf. [section 6.6.4](#)), and also supported in the Planetary prototype based on technology introduced in this thesis (cf. [section 11.3.1](#)).

<sup>49</sup>Currently, one has to “open this” and then open the “discussion” tab for that page.

<sup>50</sup>During the population of the discussion forums with old e-mail posts, as reported in [section 10.3](#), similar cases of discussions covering two distinct symbols had occurred.



*Example.* Three users wanted to reply to a decision, saying, for example, that they did not like it, but missed an appropriate argumentative type for doing so.<sup>51</sup>

Individual users wanted to perform further actions the argumentation model does not provide for:<sup>52</sup> creating posts with more than one argumentative type, as discussed in [section 10.3](#), replying to an example with an argument or an elaboration, replying to an idea with another idea, posting a typed reply on an untyped post, and posting a counter-argument to an argument. Another user pointed out a new issue with a knowledge item but merely posted an untyped comment, as he or she had found the description for an *Issue* (“something that is wrong”) too strong and was actually not sure whether really something was *wrong*. He or she would have preferred a weaker post type, such as “new feature request”, as known from bug tracking systems. Three users found it helpful to be forced to make up their mind; one characterized that as “convention over configuration”, another one anticipated the community to adapt the given supply of argumentative types.

#### 10.5.5.5 E-Mail Subscription and Notification

Most users approved of the e-mail subscription to discussion forums. Three users explicitly appreciated the option for automatic subscription; however, half of all users criticized the incoherent user interface.<sup>53</sup> Two asked for a subscription by argumentative *types*; for example, one might only be interested in decisions and (objective) arguments but not in (subjective) positions. Five users explicitly approved of the content of the notification e-mails; one argued that more content is not necessary, as people would have to go to the wiki in any case for participating in a discussion. Four users nevertheless requested more information in the e-mails, at least the subject of the post, for disambiguation. One user suggested including the argumentative type. Three users pointed out that it is not clear what exact post an e-mail refers to.<sup>54</sup>

#### 10.5.5.6 Inline Queries

Seven users found the possibility to write their own queries helpful; individual reasons included the ability to draw attention to certain topics and supporting administrative tasks. Four users explicitly mentioned that one would have to know SPARQL in order to write queries. Three users also realized that one would have to know the ontologies used by the system; one asked for their documentation. Two users suggested an interactive query composer. Several users suggested additional queries for: (i) resolved issues (for experience management and e-learning) – as opposed to the list of *unresolved* issues currently offered –, (ii) symbols with at least two notation definitions, with one of which there is an issue, (iii) most frequently used CDs, (iv) users who reused symbols from CDs that I wrote, and (v) discussions that are expiring now<sup>55</sup>. One user requested a general

<sup>51</sup>[Section 6.6.2](#) discusses the possibility to reopen discussion threads after a decision.

<sup>52</sup>Note, however, that the DILIGENT developers deliberately restricted the supply of types in order to keep reasoning feasible (cf. [TPS+05]), and to keep discussions more focused, as discussed in [section 3.6.1](#).

<sup>53</sup>Four users correctly remarked that the configuration interface for the global auto-watch option does not belong into the discussion forum user interface but into the user’s preference pane. Two were not sure what “auto-watch” meant; one of them thought it would only refer to replies to his/her posts.

<sup>54</sup>This could partly be remedied by including the subject, but a completely working solution would require a link to the exact post, which IkeWiki does not support.

<sup>55</sup>The Cicero system reviewed in [section 9.5.7](#) supports deadlines for discussions.



overview of most discussed, most active, and best rated topics, as in other forum systems. One user suggested providing a query template for watching a whole CD (including all of its subitems), which users would be able to copy and adapt to the CDs they are interested in. The selection of queries on the entry page of the OpenMath wiki (cf. [section 9.4.3](#)) yielded some negative feedback. Only one user said that these queries provided exactly the desired information.

Multiple aspects of the presentation of query results were criticized. Two users got confused by the result sets not being disjoint and requested a clearer arrangement. Four users did not find it obvious what a listed discussion was about and suggested displaying the subjects of the discussion posts.<sup>56</sup> Three found the display of result items confusing.<sup>57</sup> One user suggested displaying the dates of the most recent discussion posts; similarly, another user suggested sorting query results by date.<sup>58</sup> Another user suggested to display links to symbol pages using rendered symbols.<sup>59</sup> Here, as well as in the community survey, the absence of direct links to discussion pages – instead of knowledge items – was criticized. Finally, four users suggested to accommodate the conflict between a large number of query results and the frequently requested more informative presentation of results by displaying a limited but expandable number of query results. One user requested the full expansion of the result list to be shown on a searchable or filterable page; another user suggested faceted browsing, such refining a list of unresolved issues by the type of their subject, e.g. CD or symbol.<sup>60</sup>

### 10.5.6 Feedback on Incoherent Integration

Some users would generally have expected an integrated work environment to feature a more coherent user interface and more uniform interaction patterns for related tasks. Concretely, the users encountered a number of problems that could have been avoided by a better adaptation of the individual integrated services as well as the integrating environment. Independently from the integration, there were obvious usability issues with single components, most notably IkeWiki's metadata editor, but these are not in the focus of this discussion, as standard user-centered design and usability evaluation methods would likely have helped to avoid them.

Problems with the integration of individual services included the links to symbol definitions in the “prefix” view of mathematical object not being adapted to SWiM's URI format (reported by three users) – which would have provided an alternative to the non-intuitive middle clicking on rendered symbols –, the metadata extension of the document editor not supporting line breaks in long metadata values, such as symbol descriptions (three users), and the preview of a notation definition looking different from a rendered object in a CD (one user).

Features of the integrating IkeWiki environment caused problems where they conflicted with SWiM's different design but had not been disabled, or where they had not been properly adapted

<sup>56</sup>That would require much more space on the entry page, as there can be multiple unresolved issues per knowledge item, and different subjects in each such thread.

<sup>57</sup>Two users complained about the dashes separating the links to the discussions in the result set; however, that was the best output that could be achieved with IkeWiki's limited formatting support.

<sup>58</sup>That has been implemented meanwhile (cf. [section 9.4.3](#)).

<sup>59</sup>That is feasible in SPARQL if the rendering of each symbol in the “constant” role (cf. [section 2.4.5.3](#)) is made available in the RDF graph.

<sup>60</sup>KiWi, for example, would support that (cf. [section 9.6.4.4](#)).

to mathematical documents. Conceptual design differences comprised multilinguality of documents<sup>61</sup> (two users) and metadata (another two), as well as different annotation paradigms – semantic markup in SWiM vs. plain HTML and a separate annotation user interface in IkeWiki<sup>62</sup> (one user), and, related to that, fields displayed in the metadata editing form that did not correspond to OpenMath CD metadata (another one). A case where an existing IkeWiki facility had not been adapted to mathematical content was full-text search for symbol renderings (discussed above in [section 10.5.4](#)). Finally, three participants of the community survey complained about SWiM's poor speed and reactivity, which was most likely to blame on rendered pages not being cached – a functionality that IkeWiki neither had nor really needed, but that SWiM would have needed due to the complexity of its publication process, as discussed in [section 9.3.2.4](#).

In other situations, users found the user interface coherent. One user would have preferred, as many others, a quicker access to the notation definition editor, but then pointed out that he or she also appreciated the consistency in always having to open the “edit” tab, and compared the tab bar to the menu bar of Mac OS, which is always in the same place as well.

### 10.5.7 Discussion of the Evaluation Setup and Method

This section critically discusses the evaluation setup and method employed in the supervised experiments with test users, before [section 10.6](#) summarizes the evaluation results.

More than a quarter of the negative feedback – 82 out of 306 statements, referring to 39 distinct problems – referred to small problems that, if known, could have been avoided without fundamentally changing the design of the system or the knowledge model. Fixing them would have required an estimated net amount of two more weeks of work on the implementation. Typical problems include, classified by usability aspect:

**Learnability:** distracting user interface elements that were not relevant for the OpenMath CD case study and could therefore have been disabled, a non-intuitive terminology in the repository log messages, missing labels for link types in the ontology, missing explanation of the icons representing argumentative types of discussion posts, and the incoherent user interface for setting up the e-mail subscription to discussion forums

**Effectiveness:** too small fonts for mathematical objects, two irritatingly similar objects on two different sample pages to be visited during the experiments, a bad presentation of inline query results, and various shortcomings of the discussion forum layout

**Satisfaction:** incoherent colors throughout the user interface resulting from the integration of different components, and additional information requested by some users not being available in discussion notification e-mails

<sup>61</sup>IkeWiki offers basic support for multilingual content (cf. [figure 9.1](#) on page 291), which two users wanted to try, but OpenMath CDs are only written in one language and do not even allow to express what language that is.

<sup>62</sup>In contrast to SWiM, a page in IkeWiki consist of HTML-like presentation markup; the document editor does not support semantic markup. Instead, one has to add page and link types in the “annotation” tab and metadata in the “metadata” form, resulting in standoff RDF triples. However, the “annotation” tab does not have any functionality for semantic markup. Secondly, the *vocabulary* for annotations is maintained with a built-in ontology editor. This editor is accessible via a toolbox titled “edit”, which distracted five test users from finding the document or metadata editor. While that box has an unfortunate title even from an IkeWiki point of view, it plays a much less important role in SWiM and should therefore be removed from the user's sight.

The vast majority of these feedback statements (69 out of 82) was about problems I had not been aware of before starting the user experiments. They had first been pointed out to me by test users. This suggests that an iterative evaluation approach might have yielded clearer feedback about the *actual* issues of the conceptual model and the user interface design.

Choosing two or three members of the overall group of test users for the first round and then fixing the avoidable problems *they* had encountered would have reduced the noise in the subsequent experiments. General usability research corroborates that: JAKOB NIELSEN points out that it is usually sufficient to evaluate with five test users in order to discover most usability problems. Additional test persons should rather be asked to participate in a later evaluation round [Nieoo]. However, when a user encountered a known problem, I acknowledged it and gave instructions on how to continue with the given task, as advised in [BRoo]. The overall collected feedback shows no evidence that little annoyances due to avoidable problems significantly distorted the results. Instead, subsequent test users usually gave a considerable amount of fresh feedback statements.

## 10.6 Evaluation Results and their Interpretation

This section summarizes and interprets the results obtained from the evaluations described in sections 10.3 to 10.5. Section 10.6.1 summarizes the results of evaluating the usability of the OpenMath wiki. Sections 10.6.2 and 10.6.3 abstract from the OpenMath wiki and discuss the impact of the general lessons learned from this evaluation on designing integrated semantic work environments.

### 10.6.1 Usability of the OpenMath Wiki

Based on an analysis of three typical CD maintenance workflows and on requirements gathered from the community, I have specialized the SWiM semantic wiki into a tool for maintaining the official and contributed OpenMath CDs. The hypotheses about its usability established in section 10.2.3 largely hold, as the following summary shows:

1. **Quickly Fixing Minor Errors:** (a) The knowledge model, with its hierarchy of CDs and their subparts and a separate treatment of metadata, and the document editing interface – or, alternatively, the metadata editing form – effectively supported a majority of test subjects of the supervised experiment as well as participants of the community survey<sup>63</sup> in fixing errors locally (cf. sections 10.4.2 and 10.5.3). However, the metadata editing form integrated into the system exposed some problems that restricted effectivity. Further, general observations that apply to all effectivity-related hypotheses are summarized below. (b) Most experiment test subjects understood at what granularities they could edit a CD (cf. section 10.5.3). However, previous knowledge of related knowledge models and of related user interfaces was of advantage particularly for navigation (cf. section 10.5.2.2), and the user interface could not entirely compensate for a lack of such previous knowledge. (c) Most users found that the system's revision logs made changes retraceable in a comprehensible way; however, the user interface for summarizing changes as well as the display of changes for non-experts could be improved (cf. section 10.5.3.4).

<sup>63</sup>In the following, this will be abbreviated to “most users”.

2. **Fixing and Verifying Notations:** (a) Most users were effective able to navigate from a mis-rendered mathematical object to the editor for the responsible notation definition (cf. sections 10.4.2, 10.5.4.2 and 10.5.4.3). In the underlying knowledge model, these are linked via the (formal) definition of a symbol. However, getting from a symbol in a rendered object to its definition would hardly have been possible without instructions due to the non-intuitive user interface (cf. section 10.5.4.2). (b) Moving on to the notation definition was, once more, easier for users with previous knowledge of related models (cf. section 10.5.2.2); the user interface should make the connection more explicit for non-experts. (c) Most users were effectively able to edit a notation definition (cf. sections 10.4.2 and 10.5.4.5); however, a less cluttered display of the content in the document editor, as well as an easier access to special characters would help to further improve that. (d) Most users found the preview of a notation definition comprehensible, and helpful for verifying their edits (cf. sections 10.4.2, 10.5.4.4 and 10.5.4.7).
3. **Peer Review and Preparing Major Revisions by Discussion:** (a) The argumentation model supports problems with single knowledge items, or posts following up on such problem statements, to be associated exactly with that knowledge item. The content analysis (cf. section 10.3) and, to a lesser extent, the supervised experiment (cf. section 10.5.5.3) show that that covers most but not all practical situations. Concerning this and the following sub-hypothesis, both the content analysis and the supervised experiment once more highlight the conflict between a restricted argumentation model suitable for querying and reasoning and a model that fully captures discussions as they occur in practice. (b) The content analysis (cf. section 10.3), the supervised experiment (cf. sections 10.5.5.2 and 10.5.5.4), and, to a lesser extent, the community survey (cf. section 10.4.2) show that the knowledge model captures most of the argumentation primitives that occur in practice. However, a “question” type was missed most (cf. section 10.3), and a better documentation of the existing argumentative types – and possibly best practices of how to construct common argumentative structures using them – (cf. sections 10.5.5.1 and 10.5.5.4) might have helped to utilize them better. (c) Most users understood how to create the desired discussion post (cf. sections 10.5.5.1 and 10.5.5.4), but suggested an even more self-explanatory user interface. The content analysis possibly indicates that narrowing down the subject of a discussion post to a knowledge subitem is less learnable than narrowing down the argumentative type (cf. section 10.3). (d) They were effectively able to write a post but requested more flexibility in doing so (cf. sections 10.5.5.1 and 10.5.5.2).
4. (a) Besides issues with specific system components mentioned below, the most severe general **dissatisfaction** with the system was caused by easily fixable bugs (cf. section 10.5.7), as well as missing adaptations of integrated components and the integrating environment – including performance speedups –, most of which would also be easy to realize (cf. section 10.5.6). Additionally, test users frequently requested certain tasks to be simpler to perform and additional familiar features to be supported. (b) The community has given the system a fair **utility** rating; the content analysis of discussion posts confirms that for that particular feature set (cf. section 10.3). However, while acknowledging the added value of individual services offered by the wiki, the community did not fully approve of SWiM taking over

workflows completely but opted for integrating the wiki services more closely with the existing infrastructure – not just with the Subversion repository but also with the static website and the mailing lists (cf. [section 10.4.4](#)). Further fundamental reservations against the utility of a wiki for maintaining OpenMath CDs can, nevertheless, be addressed with a suitable configuration of the system (cf. [section 10.4.2](#)).

Regarding the learnability (cf. hypotheses [1b](#) to [3c](#)) and effectiveness (cf. hypotheses [1a](#), [2a](#) and [2c](#) to [3d](#)) of the system, the following general observations were made:

**Learnability:** Users with previous knowledge of related knowledge models or user interfaces easily learned the steps necessary for accomplishing workflows, whereas less experienced users were more frequently taken in by misconceptions and requested more obvious instructions to be given by the system itself (cf. [section 10.5.2.2](#)). The feedback gathered in the supervised experiments gives evidence that many misconceptions about the design and knowledge model could have been avoided by a more self-explanatory user interface. [Section 10.6.3](#) discusses possibilities for that.

**Effectiveness** was occasionally hampered by bugs, which could be fixed with only minor additional investment (cf. [section 10.5.7](#)). Key reasons for users being completely unable to accomplish a task without further instructions were a lack of understanding – or a different conception! – of design concepts and the knowledge model, as observed in [section 10.5.2.2](#), and an incoherent and unsupportive user interface – which, again, contributed to misunderstandings. These findings suggest that improving learnability will also improve effectiveness.

## 10.6.2 The Challenge of Integrating Heterogeneous Services

While the main contribution of this thesis is *enabling* the integration of heterogeneous services for managing mathematical knowledge – by representing the latter in a mutually comprehensible way and translating between different representations as required, the experiments show that making them *usable* also requires harmonizing their ranges of functions, their user interfaces, and their interaction patterns. This applies to individual services as well as the integrating environment. The test users of the OpenMath wiki not only expected a more coherent integration of the individual services, but, realizing that these services were integrated into a wiki with discussion forums, they also had expectations from previous encounters with such systems, not all of which the IkeWiki/SWiM environment met to their satisfaction.

## 10.6.3 Semantically Transparent User Interfaces for Integrated Environments

Understanding the design concepts and the knowledge model behind the user interface has proven a critical prerequisite to successfully accomplishing tasks (cf. [section 10.5.2.2](#)). Users had problems when they had not understood certain concepts in the first place (e.g. that notation definitions and symbols are distinct resources linked to each other as  $\text{Symbol} \xleftarrow{\text{rendersSymbol}} \text{NotationDefinition}$ ) and the user interface did not support them in learning them, but also when they had actually understood them but the user interface did not intuitively reflect their understanding (e.g. when they had understood the former concept but did not find the navigation tree usable).

These experiences show that the user interface of the OpenMath wiki is not yet sufficiently *semantically transparent*, i.e. in too many situations it neither “enables a user to access its semantic objects [in this thesis: knowledge items] and their relations via the corresponding UI objects” [KKo9a], nor does it “[allow] the user full access to its intention” [KKo9a]. The need for semantic transparency is not new, as ANDREA KOHLHASE and MICHAEL KOHLHASE point out in [KKo9a], but it becomes more apparent in the heterogeneous integration setting of the OpenMath wiki, and the usability experiments give concrete evidence *where* more semantic transparency is needed.

From specific situations in the OpenMath wiki, the following general lessons for making integrated systems semantically transparent can be learned:

**Self-Explanatory User Interface:** The user interface **SHOULD** explain the interactions it offers.  
— For example, the “open this” links in the OpenMath wiki should visually indicate *what* they open, as some users were not sure about that.

**Familiar and Consistent Terminology:** The user interface **SHOULD** use the terminology the user is most familiar with, regardless of the underlying representation format, and across different representation formats such as XML vs. RDF.<sup>64</sup> — For example, the users of the OpenMath wiki did not consider it helpful that the revision log used URIs and terms from SWiM’s internal storage layout instead of reusing the more familiar general OpenMath terminology (“fragment *cd:arith1+plus*” vs. “the *plus* symbol from the *arith1* CD”). Similarly, a user with an OpenMath background but no semantic web background does not have to or not even want to know that SWiM represents the structures of CDs in RDF. For example, the metadata editor and the revision log messages should not only have displayed the RDF names of metadata fields, but also the OpenMath XML names.<sup>65</sup>

**Explanations for Technical Terms:** Indispensable technical terminology **SHOULD** be explained comprehensibly to non-experts. — For example, in the OpenMath wiki, it is certainly legitimate to expose the prototype of a pattern-based notation definition under that name in the published view and editor of a notation definition. However, as not all users are familiar with that particular notation definition language<sup>66</sup>, or with notation definition languages in general, the system should additionally explain – e.g. in a tooltip – that this is a content markup pattern to be matched.

**Viewing Structures in Multiple Frames:** The interface **SHOULD NOT** be restricted to one view on the underlying structures, as a user might not be familiar with that particular view. As an approach to “*individualize[d] semantic transparency*” [KKo9a], KOHLHASE and KOHLHASE have studied user interfaces that support the practice of *framing*, i.e. viewing objects in terms of structures already understood, and concluded that “*framing-aware interactions allow users*

<sup>64</sup>In the same direction, ANUPRIYA ANKOLEKAR et al. observed that semantic web applications still “*tend to burden people cognitively with their own internal semantic models and ontologies*” [AKT+08] and opted for new “*ways for easing people into working with the semantic models underlying their software and tools*” [AKT+08].

<sup>65</sup>However, considering more advanced use cases, I do not consider it helpful to completely hide the RDF names. For example, experienced users interested in writing their own queries or consuming and processing linked data from the knowledge base should know the RDF names. Therefore, it seems most reasonable to expose both names.

<sup>66</sup>It is debatable whether the term “prototype” was a good choice at all, as other notation definition mechanisms do not use it. On the other hand, there is no established standard term.



to choose the right level of abstraction of explanations” [KKo9c]. – In the OpenMath wiki, the usability experiments showed that the navigation tree lacks semantic transparency in that it only displays the connection between a CD and a symbol in the whole→part direction – i.e. the *hasPart* property in the OpenMath CD ontology –, but that, when viewing the symbol, users would have been more familiar with the part→whole frame. An analogous case was that of the link between a notation definition and the symbol it renders. SWiM’s “transparency gap” w.r.t. this aspect of framing does not root in the ontology – which does declare inverses for most properties – but in the reasoner, which only handles RDFS (cf. [section 9.3.1](#)) and thus does not understand inverse properties.

Note, however, that it is not necessarily easy to design a fully semantically transparent user interface. Design choices that make a user interface learnable and effective for simple tasks may restrict the view on more complex underlying structures. For example, in the OpenMath wiki it has sometimes been found necessary that a discussion post should refer to more than one knowledge item (cf. [sections 10.3](#) and [10.5.5.3](#)). The SIOC argumentation model does not preclude that, but the user interface currently does. Free associations of discussion posts to knowledge items would add complexity to the user interface and thus pose another potential challenge to usability.

## 10.7 Conclusion and Future Work

In order to obtain initial guidelines for integrating heterogeneous services for managing mathematical knowledge into *usable* overall environments, I have evaluated the usability of the OpenMath wiki – particularly its support with three typical knowledge collection maintenance workflows. The results of the three complementary evaluation steps – analyzing user-generated content w.r.t. its annotations, surveying the OpenMath community about the perceived utility of the wiki and their satisfaction with it, and conducting supervised experiments with test persons who had not used the system before, in order to assess its learnability and effectiveness – will not only help to develop an improved OpenMath wiki; they also yielded general insights into the integration of heterogeneous services into a coherent environment, and into the design of more semantically transparent user interfaces for semantic MKM services.

[Section 10.7.1](#) reviews to what extent the overall contribution of this thesis has been evaluated, [section 10.7.2](#) discusses other aspects of the OpenMath wiki that could be evaluated, and [section 10.7.3](#) discusses customizable system ontologies as a means of achieving the semantic transparency demanded in [section 10.6.3](#).

### 10.7.1 Coverage of the Evaluation

The evaluation this chapter reports on has particularly focused on three typical OpenMath CD maintenance workflows introduced in [section 6.1.2](#) – quickly fixing minor errors, fixing and verifying notations, and peer review and preparing major revisions by discussion – but also touched on managing a project. Of the structural dimensions of mathematical knowledge introduced in [section 2.1](#) and represented by the ontologies of [chapter 3](#) it has thus covered logical/functional structures – to the extent the OpenMath CD language supports them –, the notation of symbols, administrative metadata, and discussions about knowledge items. This permits a generalization of



the results obtained to other structural dimensions and representation languages and ontologies that cover them, as much as their knowledge model intersects with that of OpenMath CDs. In particular that means that the results are fully applicable to OMDoc, modulo syntactic and structural differences between OMDoc and the OpenMath CD language. Of the primitive services introduced in [chapter 6](#), the evaluation has covered visual editing, publishing for humans, information retrieval by multi-dimensional queries, and arguing about problems with knowledge items. Of the integration techniques introduced in [chapters 7 and 8](#), it has covered the transparent translation of semantic markup to RDF and the transparent translation between a fine-grained representation of knowledge items and files in a repository. Moreover, it has pointed out additional situations where collaboration services should be integrated more directly into the user interface in the manner of [chapter 7](#) – for example for speeding up the notation editing workflow (cf. [section 10.5.4.7](#)).

### 10.7.2 Evaluating Collaboration and Other Axes and Components

This section outlines further aspects of the SWiM integrated collaboration environment – and the services it integrates – that could be evaluated in future: first its usability in more specific collaboration scenarios, and secondly other components and axes in terms of the interaction triptych model (cf. [section 10.2.1](#)), which would complement the usability evaluation.

#### 10.7.2.1 Usability of the Support with Specific Collaboration Scenarios

The knowledge collection maintenance workflows could be evaluated in more depth. Deficiencies of the evaluations conducted so far were that the domain experts from the OpenMath community gave little detailed feedback about the workflows, whereas the supervised experiments were conducted with non-experts, and that the collaboration services evaluated – except the argumentative discussions – have not yet been used in production scenarios. Aspects that could be evaluated with experts in a production setting include the retraceability of minor fixes for *other* collaborators, and the usefulness of the results of queries for ongoing discussions for collaborators who want to engage in them.

A particular aspect of the argumentative discussions that has not yet been deployed in the OpenMath wiki and therefore not evaluated is the problem-solving assistance introduced in [section 6.6.3](#).<sup>67</sup> Here, one could prepare a faulty knowledge collection and let one group of users collaborate on it with assistance enabled, another one without. That would help to test hypotheses such as that the problem-solving assistance helps users to get more problems solved at all, or within a limited time, or with fewer – potentially unfocused – discussion posts.

Any future re-evaluation of usability could assess learnability and satisfaction more precisely by asking users competency questions after they have used the system, and by using a standardized questionnaire for subjective feedback. The System Usability Scale (SUS [[Bro96](#)]), for example, even has two questions covering good integration (5) vs. inconsistency (6), plus two questions (9 and 10) focusing on learnability [[LS09](#)].

---

<sup>67</sup>I discussed the idea with some of the experiment test subjects. Some of them considered the approach introduced in [sections 6.6.3 and 9.3.3.2](#) too intrusive. Instead of directly offering a potential volunteer to put the “winning” solution into practice, two users suggested to link to the corresponding discussion threads instead to allow collaborators to make a more informed decision before taking action.

### 10.7.2.2 Evaluations Along Further Axes

On the performance axis of the interaction triptych model, the development of SWiM and its deployment for OpenMath has so far focused on proving the feasibility of integration: enhancing a semantic wiki with a fine-grained representation of knowledge on top of an existing file-based repository without disrupting file-based workflows, and enabling possibilities that had not been available before, such as semantics-oriented queries and problem-centered discussions. The only systematic evaluation that I have so far conducted along the performance axis was checking the effectivity of the integration, namely whether editing in the wiki frontend does not break the files in the repository (cf. [section 9.3.2.3](#)). A thorough performance evaluation would be in order after the technical upgrades suggested in [section 9.6.4.4](#), which are likely to improve performance.

Usefulness of the content was a side issue insofar as the target audience knew the OpenMath CDs very well anyway and had no problems finding the information they needed without a wiki – even without any tool support, as a survey confirmed (cf. [appendix D.2.1.5](#)). Therefore, providing useful content had not per se been a requirement for the OpenMath wiki. In any case, the wiki provides the same content as the static XHTML renderings of the CDs at [openmath.org](http://openmath.org), i.e. it gives access to all official and contributed CDs and presents them in the same layout.

### 10.7.3 Customizable and Documented System Ontologies as a Means of Appropriating Environments and Achieving Semantic Transparency

One practical approach towards more semantically transparent user interfaces is grounding them on ontologies. One example – navigation in the OpenMath wiki – has been discussed in [section 10.6.3](#). Another one – guiding the flow of a discussion based on an ontology of domain-specific problem and solution types – has been introduced in [section 6.6.3](#). This is merely the foundation, as the interfaces realized so far in SWiM do not yet obtain and display human-comprehensible *explanations* from these system ontologies<sup>68</sup>. In the same direction, KOHLHASE and KOHLHASE have developed an extension that makes spreadsheets semantically transparent by displaying informal help texts from a domain ontology related to the data in the spreadsheet, where the aggregation of the help texts is controlled by formal rules in that domain ontology as well as a system ontology of general spreadsheet concepts [[KK09b](#); [KK09c](#)].

Now, once the user interface of a system is sufficiently adaptable via its underlying ontologies – while even staying inside the system, by the considerations of [section 9.6.4.1](#) –, this offers the user community a great opportunity to *appropriate* it<sup>69</sup>. While such *end-user modifiability* should “*not transfer responsibility for good system design to users*” [[FG90](#)], it allows the users “*to carry out a constrained design process to modify it*” [[FG90](#)], where it “*does not satisfy [their] needs and [...] taste*” [[FG90](#)]. Concretely, where the community misses, e.g., a certain flow of discussing problems, it can adapt the argumentation ontology. In the OpenMath wiki, adding a “question” type would have been such a case (cf. [section 10.3](#)). Where novice users miss a certain frame that views structures in a way they are familiar with, or where they miss explanations for existing user interface elements, experienced users can add the required formalizations or documentations to

<sup>68</sup>see [section 9.6.4.1](#) for the terminology

<sup>69</sup>speaking in terms of [[KK09b](#)]

the respective system ontologies.<sup>70</sup> The users of the OpenMath wiki frequently requested intuitive labels for ontology-based navigation links, a better explanation of the argumentation possibilities, and an overview of all ontologies available for writing custom queries – all of which could be addressed by such means.

## Acknowledgments

JAKOB ÜCKER seeded the OpenMath wiki with discussion posts from e-mails, classified them by subject and by argumentative type (cf. [section 10.3](#)). Furthermore, I would like to thank the users of the OpenMath wiki: those who contributed further discussion posts, the participants of the community survey, and the test users of the usability experiments.

---

<sup>70</sup>To a limited extent, this also holds for XML schemata, as remarked in [section 2.3.2.1](#).

## **Part IV**

# **Conclusion and Future Work**



## Conclusion and Future Work

This thesis has presented an integration approach that enables effective collaboration on semiformal mathematical knowledge. It provides an analysis of the structures of mathematical knowledge as well as typical workflows performed on them. I have developed an interoperability layer that covers all structural dimensions of mathematical knowledge; it remains compatible with existing representations of mathematical knowledge, which mathematical services understand, but its semantic web nature also enables integration with related non-mathematical knowledge. Developing software support for workflows in mathematical knowledge management (MKM) requires decomposing them into simple tasks, most of which are supported by existing primitive services, which, however, often speak different languages. I have shown how such services can effectively be utilized in the frontend and the backend of an integrated collaboration environment, which transparently translates between the different languages understood by the services. Finally, I have evaluated the usability of the SWiM environment in three typical knowledge collection maintenance workflows, and, in doing so, gained insights into the general challenges for usability of environments that integrate heterogeneous knowledge-oriented services.

### 11.1 Retrospective Summary

Mathematics is a collaborative science whose results are applied throughout the STEM fields. While mathematicians have adopted the Web and the Web 2.0 as a collaboration medium, such applications neither support automated knowledge reuse, nor intelligent information retrieval, nor are they integrated with automated reasoning and computation. Semantic web technology, which has enabled comparable innovations in other fields but failed to be adopted by MKM researchers due to its immaturity at the time of the first attempts to do so, is now, with standardized query languages and in combination with web 2.0 technology, applicable to MKM problems.

None of the previously existing languages for representing mathematical knowledge has been found to satisfactorily cover all of its structure – logical/functional, rhetorical and document structures, information on presentation, metadata, and connections to the application environment and its users –, to support flexible degrees of formality and interlinking mathematical with non-mathematical knowledge, while being comprehensible to automated agents as well as humans.

The combination of the OMDoc semantic XML markup language, which particularly excels in its wide coverage of structures of mathematical knowledge and degrees of formality, with the RDF data model of the semantic web, which particularly excels in its interlinking capabilities and embeddability, satisfies these requirements. I have derived an ontology, i.e. an RDF vocabulary, from OMDoc's model of mathematical knowledge, and specified a translation from OMDoc's semantic markup to an RDF representation in terms of that ontology. Conversely, I have extended OMDoc by RDFa to enable the annotation of mathematical documents with non-mathematical information. The benefit of these two contributions is twofold: Mathematical markup languages have been opened up for an inclusion of structural information that is not strictly mathematical but likewise required for practical applications of mathematics. Secondly, markup languages without a specific mathematical vocabulary, such as DocBook, can now be used for mathematical applications by annotating documents with terms from the OMDoc ontology.

Realistic workflows in creating, formalizing, and organizing mathematical knowledge so that it can be understood, reused, and applied are not supported out of the box by mathematical services. They have to be decomposed into sequences of simple tasks, each of which is supported by a primitive service. Implementations of most of these primitive services for editing mathematical knowledge, publishing it – for human and machine audiences –, validating its formalization, and retrieving information from knowledge collections already exist; I have added a model of arguing about problems with mathematical knowledge items and a prototypical primitive service that uses that for problem solving assistance. However, each of these primitive services has its own native language, for example  $\text{\LaTeX}$  – possibly with semantic macros – for editing and paper publishing, XML for validation and web publishing, and RDF – with suitable ontologies – for other validation tasks and for information retrieval. Integrating primitive services in order to enable support for complex workflows therefore requires translation.

We have shown how to integrate services into the user-centered frontend and with the database backend of a collaborative work environment. The SWiM semantic wiki combines both perspectives into a coherent environment for producing and consuming mathematical knowledge, particularly proving the feasibility of integrating heterogeneous primitive services in order to support concrete knowledge management workflows. They have been integrated on top of the foundational infrastructure of an existing semantic wiki, transparently translating between different representations of the same knowledge in order (i) to feed the knowledge to each service in a granularity or representation it is capable of processing, (ii) to give semantic querying and reasoning engines access to knowledge structures encoded in markup languages, and (iii) to preserve semantic structures in published documents, so that embedded assistive services can utilize it.

SWiM has particularly been enabled to support three typical knowledge maintenance workflows in a realistic medium-sized collection of semiformal mathematical knowledge – the official and contributed OpenMath Content Dictionaries (CDs). Here, SWiM and the services it bundles had to be integrated into an existing pre-web-2.0 infrastructure relying on a file repository and e-mail communication. The SWiM frontend provides richer collaboration features than the traditional infrastructure without disrupting traditional workflows, such as editing CDs as files in a text editor. Making the CDs with their unidimensionally logical/functional mathematical knowledge available in SWiM adds the social domain of users discussing about them. The queries created in the OpenMath CD installation of SWiM demonstrate how the combination of different



dimensions of mathematical knowledge and its environment yields new information that is relevant for contributors to a project – a practical example of how semantic web technology makes MKM tools scale beyond pure mathematical knowledge into the environment where it is applied. For the three maintenance workflows, SWiM has proven usable in that users – given some initial understanding of the knowledge structures – were able to learn how to accomplish tasks, in that they could effectively accomplish them with minor assistance, and in that they perceived SWiM’s support useful for the OpenMath community.

Beyond OpenMath CD maintenance, the usability evaluation of SWiM helped to point out a general challenge in integrating heterogeneous knowledge-oriented services. The user interface of a semantic system should generally support non-expert users in learning the structures of the underlying knowledge by way of semantic transparency. However, achieving the latter involves more work in an environment whose services operate on *different* semantic representations of the same knowledge. The user should not have to care about these differences but see everything in a consistent terminology, possibly with access to explanations and framed into structures he is already familiar with.

The architecture for integrating services operating on mathematical knowledge has been designed not only with SWiM in mind but independently from concrete environments and markup languages. The translation steps (ii) and (iii) mentioned above have already successfully been applied in a different backend system, the TNTBase XML database, where they support learners and instructors in browsing and retrieving information from a lecture note repository [DKL+10b].

My implementation of the translation steps (ii) and (iii) consists of an abstract, generic part, allowing concrete implementations to support languages other than OMDoc and OpenMath. Drawing on the structural similarities of mathematical theories and semantic web ontologies, even the concrete implementation for OMDoc proved to be easily adaptable for OWL ontologies encoded in OMDoc. That allows for utilizing OMDoc and services originally developed for collaborating on mathematical documents for the formalization and fine-grained documentation of heterogeneous and modular ontologies and ultimately paves a path towards more semantic transparency: The user community of an environment like SWiM, whose central features browsing, querying/searching, and arguing are largely influenced by ontologies, can now appropriate the environment by customizing these ontologies, even without leaving the familiar environment.

## 11.2 Evaluation Against the Original Research Questions

This section assesses to what extent this thesis has addressed the initial research questions posed in [section 1.6](#):

**Knowledge: How can knowledge be made reusable and comprehensible across knowledge bases?** The mathematical knowledge representation and exchange language resulting from the combination of OMDoc and RDF, together with the translations between XML- and RDF-based representations of mathematical knowledge and our service for publishing semantically represented mathematical knowledge as human-comprehensible Web documents with fine-grained semantic annotations, enable reuse and support comprehension in multiple ways. Translations from OMDoc (plus MathML/OpenMath) to an increasing number of other languages, and vice versa, are available. With RDFa and the OMDoc

ontology, mathematical knowledge can alternatively be embedded into any other XML markup language. Fine-grained parallel mathematical markup and RDFa annotations allow for embedding assistive services into documents published on the Web in order to adapting the appearance of a document to the users' needs or provide them with context-sensitive information on demand.

**Workflows:** How can workflows be transferred from one knowledge base to another one?

The composition of different services, whose native languages are mutually comprehensible, enables machine support for complex workflows. Mutual comprehensibility can be achieved by translation, which exchange languages such as the one presented here facilitate. Furthermore, embedding fine-grained annotations into published documents can turn them into command centers giving access to computational services and, combined with information retrieval services, into starting points for exploring knowledge collections.

Particularly on the user interface side, however, poor support for MathML as the preferred language for human-comprehensible, semantically rich publications, is still a challenge, albeit less so than in the times of HELM (cf. [section 1.4.3.3](#)). Still, only one browser rendering engine<sup>1</sup> supported MathML to the extent required for interactive adaptation, and few contemporary HTML/JavaScript user interface toolkits support XHTML and thus non-HTML namespaces (including that of MathML). Better MathML support can, however, be expected with HTML 5 becoming more and more mainstream.

**Authoring:** How can expressive knowledge representations be authored collaboratively?

The translation between different representations of mathematical knowledge has enabled the combination of heterogeneous authoring interfaces, such as forms vs. documents for meta-data and a linear input syntax vs. a visual editor for formulæ, in order to cover all structures of mathematical knowledge. However, the usability evaluation of SWiM shows that the *coherence* of such an integration as well as the design of editing interfaces for complex, nested semantic markup still constitute challenges. Additionally, the above-mentioned technical restrictions w.r.t. MathML/XML also hold for most contemporary web editors.

**Usability:** How can environments that integrate heterogeneous services for producing and consuming mathematical knowledge be made usable? One answer to this sub-question has been determined from the usability evaluation of SWiM: semantic transparency. This has been discussed in [sections 10.6.3 and 11.1](#).

**Value:** How can the investment required to create a human- and machine-comprehensible representation of mathematical knowledge be lowered, while at the same time utilizing that knowledge to provide useful knowledge management services and applications?

Integrated semantic collaboration environments such as SWiM constitute an attempt to answer this question. Any enhancement of the formalization of a knowledge item made through its editing interface, such as giving it a more specific type or linking it to some other knowledge item, becomes instantly available to all other services, from publishing to information retrieval. Concrete examples include giving a discussion post a specific

---

<sup>1</sup>Gecko, employed in browsers such as the widely used Firefox

argumentative type, which makes SWiM guide the further flow of the discussion and update the project management overviews generated by inline queries, and creating or changing a notation definition of a symbol, which is instantly reflected in all published documents using that symbol. The editing services do not currently benefit from formalization enhancements, but it is easy to imagine how they could: For example, when editing a knowledge item and creating a typed link, the editor could auto-complete the target, offering all instances of classes in the range of the link type (assuming a closed world; cf. [section 2.3.4](#)); cf. [LKo8] for an elaboration.

Now that the methods and techniques presented in this thesis provide an effective and, to some extent, usable answer to the questions of reusing knowledge and transferring workflows for managing mathematical knowledge, their real value has to be proven in the larger field. To that end, the following section outlines future directions, along which the results obtained so far should be disseminated.

I conclude this assessment with a summary of how this thesis, beyond MKM, has addressed three of the general Web 2.0/Semantic Web challenges listed in [section 1.3.4](#):

1. **The need for more expressive ontologies for adequately representing complex knowledge** – I have developed new ontologies for representing the logical/functional structures of mathematical knowledge, and explained how the other structural dimensions of mathematical knowledge can be represented using existing ontologies.
3. **The shortage of intuitive user interfaces for semantically rich applications** is directly addressed – albeit not yet solved – by the above-mentioned findings on the usability of environments that integrate heterogeneous services.
5. **Knowledge mapping and integration** – I have asked this specifically for MKM in the “Knowledge” question; however, the solution presented in this thesis is applicable beyond MKM. As the RDF data model is not adequate to the complexity of mathematical objects (cf. the discussion in [section 2.5.1](#)), I argued that mathematical knowledge is best represented in a combination of semantic XML markup and RDF. This may not only be the case with mathematical objects; other domains where complex  $n$ -ary ordered structures have so far preferably been represented using semantic XML markup, or alternatively in XML and RDF, include chemistry – with the Chemical Markup Language CML [MRRo3; Ada09] – and music[al notation] – with MusicXML [Rec] and the Music Ontology [Raio8]. This thesis presents RDF graphs embedded into XML via RDFa and, conversely, the translation from XML to RDF, as well as the publication of both XML and RDF representations as linked data under the same URIs as approaches to making such a double tracked representation fully accessible to services.

## 11.3 Future Directions for e-Science

Future directions related to specific topics covered by this thesis have been discussed in the “future work” sections of the respective chapters. This section discusses directions towards the overall goal

of improving scientific collaboration. Aside from proofs of concept, small-scale experiments, and a small user group browsing and discussing the OpenMath CDs in SWiM, the scientific community addressed by this thesis has not yet applied the technology introduced here in practical settings. The latter would require further workflows of producing and consuming scientific knowledge to be supported; this section discusses possible directions and relevant related work.

### 11.3.1 The Planetary “eMath 3.0” Environment

Work on Planetary, a new environment that integrates many of the services and translations mentioned in this thesis and targets application scenarios ranging from collaboratively maintained knowledge collections to teaching is currently in progress [Plaa; DGK+10]. As this environment has already significantly reused results from the work presented, it seems plausible that this will turn into the ultimate collaboration platform for which this thesis aimed at delivering the conceptual and technological building blocks.

In terms of this thesis, the Planetary prototype in its current state can be considered a combination of the  $\text{\LaTeX}\rightarrow\text{XHTML}+\text{MathML}$  pillar of the publication architecture depicted in figure 6.9 – including some of the interactive JOBAD services for MathML (cf. chapter 7), more of which are to be added – with a web-based  $\text{\LaTeX}$  (soon  $\text{\LaTeX}$ ) editor and discussion forum (reusing the existing Vanilla web forum software [Van]), and thus an alternative to the SWiM semantic wiki. The discussion forums are currently in production use in MICHAEL KOHLHASE’s computer science lectures. There is an experimental installation using the  $\text{\LaTeX}$  articles of PlanetMath (cf. section 9.5.2), which is planned to replace the dated Noösphere system; in the longer run it is planned to introduce more semantics into PlanetMath by way of  $\text{\LaTeX}$ . Another such installation is being prepared for the arXiv corpus [SKG+10; Arx].

Relevant next steps on the Planetary development agenda are (re)introducing the  $\text{\LaTeX}\rightarrow\text{OMDoc}\rightarrow\text{RDF}$  translation described in section 8.1.2 and thus RDF(a)-based services and linked data publishing (cf. section 6.4), as well as supporting argumentative discussions about mathematical knowledge items (cf. section 6.6). Compared with the past, adding RDF(a) functionality would enable further navigation and retrieval possibilities and thus constitute a significant progress over both the panta rhei environment previously used in the above-mentioned lectures (cf. [Pan]) and the Noösphere software currently powering PlanetMath (cf. section 9.5.2); in the latter setting, RDF(a)-powered services could make the future links between the PlanetMath corpus and other mathematical linked datasets (cf. section 11.3.4) accessible to end users. In both application settings, feedback for improving the model and user interface for argumentation can be expected, as both user communities are larger than the core OpenMath community, which has evaluated this feature so far.

Looking into the future, this first step makes Planetary a suitable platform for enabling customizability and appropriation through community-modifiable system ontologies, as discussed in sections 9.6.4.1 and 10.7.3. Sufficiently expressive system ontologies, exposed not only via annotations embedded into (inter)active scientific documents but actually embedded into all user interface components, would reduce the need for system level programming when designing new services. The system ontologies as a high-level “API” between content authors and the system and service developers, could reasonably attract a growing community of service developers that merely have to provide client-side scripts that run queries against the “content commons” of

scientific documents and system ontologies and visualize the result. This separation of concerns seems likely to a long-term compatibility of the knowledge collection hosted by an instance of that system with future demands.

### 11.3.2 Better Support for Peer Review and Integrated Publishing

We have developed a model and user interface for a peer review workflow via argumentative discussions and implemented it in SWiM. Users of the OpenMath wiki suggested supporting further review workflows, for example allowing users to post not just typed natural language comments on problems with knowledge items and possible solutions, but to upload concrete suggested fixes (as in the OpenMath CD Manager reviewed in [section 9.5.3](#)), which are then reviewed and, in case of acceptance, applied. Combined with the editing and publishing services introduced in this thesis, an integrated environment could support the complete process of creating scientific publications. A proof of concept of supporting such an integrated authoring, review, and publishing workflow has previously been realized on top of a semantic wiki [[Scho9](#), chapter 9.2], albeit not specifically tailored to STEM.

### 11.3.3 Interactive Workbenches for Scientific Collaboration

With its focus on *managing* mathematical and scientific knowledge, this thesis has only touched on the possibility of a complete workbench for collaboratively obtaining new scientific results. While [chapter 7](#) has investigated ways of connecting documents to arbitrary mathematical services, including computational ones, I have focused on utilizing these services for the purpose of enriching or adapting the presentation of the document, but neither for free interaction nor as tools for gaining new knowledge.

*Active essays* “[combine] a written essay, program fragments, and the resulting live simulations into a single cohesive narrative” (ALAN KAY, cited in [[YWK09](#)]) for the purpose of teaching programming. The reader can execute program code embedded into a literate programming style document. Adessowiki realizes the same in a wiki for the purpose of collaborative software development, i.e. gaining new knowledge [[LMK+09](#)].

In mathematics, such a system would be a collaborative web counterpart to single-user desktop workbenches such as Mathematica [[Urla](#)]. Mathematica has actually been integrated into a wiki (cf. [section 9.5.6](#)); however, our technology would enable integrating a wider palette of mathematical services into a collaboration environment. Imagine a mathematician or engineer who “plays” with a computer algebra system (CAS) integrated into the *editor* of a work environment by entering a sequence of expressions, recognizes a pattern, and asserts that pattern as a theorem. A collaborator might take over and attempt to formalize that assertion and validate it with an integrated proof assistant.<sup>2</sup> Another collaborator might employ the CAS to generate graphs as examples for applying that theorem and contribute them to the knowledge collection. Finally, an instructor might

<sup>2</sup>This may be an ambitious goal, when the mathematical foundations of the given theorem have not yet been formalized, as one can learn from the Flyspeck experience (cf. sections [1.1](#), [6.1.2.5](#) and [9.2.2](#)). Therefore, an environment for collaborative formalization should not only give access to proof assistants but also offer help with identifying gaps in the overall formalization and retrieving existing axioms and theorems applicable in the current proof effort (cf. [[Anc09](#)]).

Table 11.1: Collaboratory types by resource and activity [BZO+07]; **bold**: those covered in this thesis

	Tools (instruments)	Information (data)	Knowledge (new findings)
Aggregating across distance (loose coupling, often asyn- chronously)	Shared Instrument	<b>Community Data System</b>	<b>Virtual Learning Community, Vir- tual Community of Practice</b>
Co-creating across distance (requires tighter coupling, of- ten synchronously)	Infrastructure	<b>Open Commu- nity Contribution System</b>	Distributed Research Center

<sup>a</sup> Bos et al. use the term “[community] infrastructure” in a larger scale context than this thesis; however, the results of this thesis could, in the long run, enable such a large-scale infrastructure, e.g. via the Web-of-Data path discussed below.

annotate the theorem and the example with educational metadata and bundle it into a course module.

Where the integrated environments presented in this thesis, as well as the others mentioned in this section, support a fixed set of workflows, myExperiment goes a step further in offering a collaborative repository for (re)using and repurposing workflows (there: pipelines of web services) in the experimental sciences [DRGS07]. Currently, the workflows are not created within the myExperiment environment, but in external workflow management systems, whereas some types of workflows can already be executed within myExperiment. The contents of the myExperiment repository – both data and workflows – have recently been published as linked data (cf. section 1.5.2 and [DRGA+10; BAB+10]) in order to leverage network effects for a more radical reuse of “research objects”.

Further inspiration for modes of scientific collaboration that future integrated work environments could support can be found in the “taxonomy of collaboratories” by NATHAN BOS et al. (cf. table 11.1 and [BZO+07]); this thesis has covered the *community data system* (“an information resource that is created, maintained, or improved by a geographically distributed community” [BZO+07]), the *open community contribution system* (“an open project that aggregates efforts of many geographically separate individuals toward a common research problem”), the *virtual community of practice* (“a network of individuals who share a research area and communicate about it online”), the *virtual learning community*, which “increase[s] the knowledge of participants [without] conduct[ing] original research [usually by] formal education [...] provided by a degree-granting institution”.

### 11.3.4 Integrating Mathematics into the Web of Data

Large collections of mathematical knowledge exist in non-RDF representations. The vocabularies and the techniques for translation and publication introduced in this thesis now enable us to contribute them to the Web of Data and fill the gap that existing linked datasets have left with



regard to mathematical foundations, as argued in [section 1.5.2](#). That will enable not only new applications in e-science – as outlined for scientific publications and mathematical models underlying experiments –, but also in statistics, enterprises, and other domains where mathematics is applied. I conclude with an agenda towards publishing relevant mathematical datasets as linked open data.

The probably most foundational dataset needed to get mathematics on the Web of Data started is the official OpenMath CDs. An initial publication is feasible with the technology available and planned for spring 2011 [[Lanio](#)]. This is, however, only the first step in making the knowledge the CDs accessible; the second step is linking mathematics-related existing datasets to the OpenMath CDs, so that services for these existing datasets can be extended by mathematical functionality – as outlined for statistical datasets in [section 6.4.1.1](#). Further obvious candidates are datasets that already contain mathematical knowledge as OpenMath-compliant XML fragments opaque to linked data applications, such as the SysMO SEEK e-science dataset mentioned in [section 1.5.2](#), which contains mathematical models expressed in Content MathML. A particularly high-impact candidate is DBpedia [[Dbp](#)], as linking its mathematical concepts to OpenMath CDs would give its large audience a more formal perspective on mathematics.

In the direction of formal logic, the OpenMath CD dataset could be extended by a linked data version of the OMDoc-based Logic Atlas [[KMR](#)]. The latter is *almost* ready for deployment as linked data in that it already uses a well-designed URI format (cf. [section 6.4.1.3](#)) and creates an RDF-like outline of the logical/functional structures (cf. [section 6.5.2.2](#)). The only remaining tasks are making the URIs dereferenceable and translating the outline to a standard RDF serialization. It is particularly remarkable that the MMT URIs not only allow for identifying resources – and linking to them – that physically exist in a knowledge collection, but also, like a query language, virtual resources such as a symbol with a given name imported into a theory via a certain named import declaration (without naming the home theory of that symbol).

Mathematical knowledge collections that are already available on the Web, but not currently in a semantic representation, should also be “triplified”, at least with shallow mathematical metadata and links to relevant OpenMath CDs. The structure of the notation census [[Lib10a](#)], a collection of verified alternative notations for mathematical symbols, already follows the OpenMath CDs, but so far only in a human-comprehensible way. The DLMF [[Nat10](#)] could benefit from access to OpenMath-aware computational web services, whereas the benefit for PlanetMath (cf. [section 11.3.1](#) for its ongoing overhaul towards more semantics) would be similar as for DBpedia.

We should also take a serious view on the April fool’s joke “Linked Open Numbers”, a huge dataset describing billions of natural numbers [[VKR+10](#)]. It provides descriptions as trivial as the name of each number in natural language, its predecessor and its successor. But how about a dataset of non-trivial properties of numbers? Accessing, for example, prime factor decompositions of large numbers – an information relevant for cryptography – in a linked dataset, could be much faster than computing it once more, provided a supercomputer has already done the computation once and published the results. From an RDF reasoning and querying point of view, such a dataset could serve as an *oracle*, providing information whose original computation would by far exceed, e.g., description logic reasoning. Another such source of non-trivial knowledge about numbers is the Web 1.0 Online Encyclopedia of Integer Sequences [[Slo03](#)].

Related to information retrieval and computation is the development of suitable query languages and reasoners. Computation has so far not been considered a part of the semantic web architecture (cf. [section 2.3](#)) but as a concept complementary to logical reasoning; JÜRGEN ZIMMER has even



suggested a layer cake architecture with the two independent columns “semantic computation” and “semantic reasoning”, and no connection between the representation languages for mathematical knowledge and semantic web ontologies [Zimo8]. However, N<sub>3</sub> reasoners support a basic set of mathematical functions (cf. [section 3.2](#)), and the upcoming SPARQL 1.1 supports basic arithmetics. Additionally, many query processors allow for defining extension functions; a path for supplying arbitrary functions to query processors via OpenMath should be investigated. Taking formal semantics and computational complexity into account, such an extension could even be specified as an entailment regime (cf. [section 6.5.2.2](#)).<sup>3</sup>

The arXiv [[Arx](#)] offers a path towards publishing mathematical structures of scientific publications. A long-term effort to automatically recover their mathematical structure from the L<sup>A</sup>T<sub>E</sub>X sources is in progress [[GJA+09](#)], the translation of 400,000 publications to somewhat more semantically structured XHTML+MathML being a first success [[SKG+10](#)]. The publications have stable URIs, and their metadata are available as XML, which makes a linked data publication feasible right now. Next, much harder steps would be interlinking with existing publication datasets, such as DBLP [[Dbl](#)] or the RKB Explorer datasets mentioned in [section 1.5.2](#), and identifying mathematical symbols that could be linked to the OpenMath CDs. With an identification of statement- and theory-level logical/functional structures, this would ultimately enable machine support for the next collaborative web-based review of a  $P \neq NP$  proof, or the next collaborative effort to formalize a proof of a theorem like the Kepler Conjecture.

## 11.4 Conclusion

Overall, this thesis makes two key contributions:

1. It introduces a system architecture involving different representations of knowledge, translations between them, and services operating on them, that enables web collaboration on semiformal mathematical knowledge as it occurs in practice – that is, embedded into an environment of an application domain, a project, and users. This has been proved by developing a concrete implementation and evaluating its usability in a practical setting.
2. This architecture is modular and independent from a concrete representation language. Evident structural similarities between mathematical knowledge and knowledge of related domains in science, technology, and engineering suggest that the methods and tools that have primarily been evaluated for mathematical knowledge in this thesis, but also applied to ontology engineering, can similarly be applied in related domains.

This defines the first step towards a wider adoption of social semantic web technologies in science, technology, and engineering, and potentially in *any* domain whose knowledge can be represented by a markup language.

---

<sup>3</sup>This possibility has been pointed out by DENNY VRANDEČIĆ (personal communication, 2010-06-02).

# **Part V**

## **Appendix**



A

Appendix

# Namespace Prefixes

Table A.1: Common namespace prefix→URI bindings used in this thesis, both for XML elements and for URIs in RDF statements

Prefix	URI	Language/Vocabulary	Section
<i>cc</i>	<a href="http://creativecommons.org/ns#">http://creativecommons.org/ns#</a> <sup>1</sup>	ccREL	2.1.7.5, 3.4
<i>cd</i>	<a href="http://www.openmath.org/OpenMathCD">http://www.openmath.org/OpenMathCD</a>	OpenMath CDs	2.4.3
<i>cdg</i>	<a href="http://www.openmath.org/OpenMathCDG">http://www.openmath.org/OpenMathCDG</a>	OpenMath CD groups	2.4.3
<i>cds</i>	<a href="http://www.openmath.org/OpenMathCDS">http://www.openmath.org/OpenMathCDS</a>	OpenMath CD signatures	2.4.3
<i>dc</i>	<a href="http://purl.org/dc/elements/1.1/">http://purl.org/dc/elements/1.1/</a>	DC Elements	2.1.7.3, 3.4
<i>dct</i>	<a href="http://purl.org/dc/terms/">http://purl.org/dc/terms/</a>	DCMI Terms	2.1.7.3, 3.4
continued on next page			

<sup>1</sup>This is the namespace of the ccREL RDF vocabulary. OMDoc 1.2 used a version without a trailing hash for the XML elements in the CC module, but we will now deprecate this version.

Table A.1: Common namespace prefix→URI bindings (continued from previous page)

Prefix	URI	Language/Vocabulary	Section
<i>dra</i>	<a href="http://www.macs.hw.ac.uk/ultra/mathlang/document-rhetorical">http://www.macs.hw.ac.uk/ultra/mathlang/document-rhetorical</a> <sup>2</sup>	MathLang DRa	2.4.6, 2.4.10.2
<i>foaf</i>	<a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a>	FOAF	
<i>gams</i>	<a href="http://gams.nist.gov#">http://gams.nist.gov#</a>	GAMS	
<i>h</i>	<a href="http://www.cs.unibo.it/~schemata/schema-h.rdf#">http://www.cs.unibo.it/~schemata/schema-h.rdf#</a>	HELM objects	2.4.10.2
<i>hth</i>	<a href="http://www.cs.unibo.it/~schemata/schema-hth.rdf#">http://www.cs.unibo.it/~schemata/schema-hth.rdf#</a>	HELM theories	2.4.10.2
<i>ikewiki</i>	<a href="http://ikewiki.srfg.at/base/">http://ikewiki.srfg.at/base/</a>	IkeWiki system	9.3.1
<i>log</i>	<a href="http://www.w3.org/2000/10/swap/log#">http://www.w3.org/2000/10/swap/log#</a>	N <sub>3</sub> logics	2.4.10.1
<i>m</i>	<a href="http://www.w3.org/1998/Math/MathML">http://www.w3.org/1998/Math/MathML</a>	MathML	2.4.2
<i>marcel</i>	<a href="http://www.loc.gov/loc/terms/relators/">http://www.loc.gov/loc/terms/relators/</a>	MARC relators	5.1, 5.2
<i>math</i>	<a href="http://www.w3.org/2000/10/swap/math#">http://www.w3.org/2000/10/swap/math#</a>	N <sub>3</sub> math functions	2.4.10.1
<i>matharg</i>	<a href="http://omdoc.org/ontology/matharg#">http://omdoc.org/ontology/matharg#</a>	mathematical argumentation	3.6.2
<i>mda</i>	<a href="http://www.modeldriven.org/2008/Architecture0ontology/Authority.owl#">http://www.modeldriven.org/2008/Architecture0ontology/Authority.owl#</a>	ModelDriven.org authority	5.2
<i>mdv</i>	<a href="http://www.modeldriven.org/2008/Architecture0ontology/Versioning.owl#">http://www.modeldriven.org/2008/Architecture0ontology/Versioning.owl#</a>	ModelDriven.org versioning	5.2
<i>mnp</i>	<a href="http://www.iwi-iuk.org/material/RDF/1.1/Schema/Property/mnp#">http://www.iwi-iuk.org/material/RDF/1.1/Schema/Property/mnp#</a>	MathNet Properties	3.4
<i>mom</i>	<a href="http://www.openmath.org/cd#">http://www.openmath.org/cd#</a>	MONET: OpenMath	2.4.10.2
<i>mowgli</i>	<a href="http://mowgli.cs.unibo.it/metadata-schema#">http://mowgli.cs.unibo.it/metadata-schema#</a>	MoWGLI: OpenMath	2.4.10.2
<i>o</i>	<a href="http://omdoc.org/ns">http://omdoc.org/ns</a>	OMDoc	2.4.4
<i>om</i>	<a href="http://www.openmath.org/OpenMath">http://www.openmath.org/OpenMath</a>	OpenMath	2.4.3
<i>oms</i>	<a href="http://www.openmath.org/cd/">http://www.openmath.org/cd/</a>	OpenMath symbols	2.4.3
			<i>continued on next page</i>

<sup>2</sup>This is the namespace URI for the XML encoding of the MathLang DRa. As the sources are not freely available, it is not completely clear whether it is also the namespace URI of the DRa ontology.

Table A.1: Common namespace prefix→URI bindings used in this thesis (continued from previous page)

Prefix	URI	Language/Vocabulary	Section
<i>omv</i>	<a href="http://omv.ontoware.org/2005/05/ontology#">http://omv.ontoware.org/2005/05/ontology#</a>	OMV	5.2
<i>omvc</i>	<a href="http://omv.ontoware.org/2005/09/changes#">http://omv.ontoware.org/2005/09/changes#</a>	OMV changes	5.2
<i>oo</i>	<a href="http://omdoc.org/ontology#">http://omdoc.org/ontology#</a>	OMDoc ontology	3.2
<i>owl</i>	<a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a>	OWL	
<i>problem</i>	<a href="http://monet.nag.co.uk/problems/">http://monet.nag.co.uk/problems/</a>	MONET problems	2.4.10.2
<i>rdf</i>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>	RDF	2.3.3
<i>rdfs</i>	<a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a>	RDFS	
<i>reason</i>	<a href="http://www.w3.org/2000/10/swap/reason#">http://www.w3.org/2000/10/swap/reason#</a>	N <sub>3</sub> reasoning	2.4.10.1
<i>reify</i>	<a href="http://www.w3.org/2000/10/swap/reify#">http://www.w3.org/2000/10/swap/reify#</a>	N <sub>3</sub> reification	2.4.10.1
<i>sao</i>	<a href="http://salt.semanticsauthoring.org/ontologies/sao#">http://salt.semanticsauthoring.org/ontologies/sao#</a>	SALT annotations	3.3
<i>scv</i>	<a href="http://purl.org/NET/scovo#">http://purl.org/NET/scovo#</a>	SCOVO	6.4.1.1
<i>sdo</i>	<a href="http://salt.semanticsauthoring.org/ontologies/sdo#">http://salt.semanticsauthoring.org/ontologies/sdo#</a>	SALT documents	3.3
<i>sioc</i>	<a href="http://rdfs.org/sioc/ns#">http://rdfs.org/sioc/ns#</a>	SIOC	2.1.7.6, 3.5.1
<i>sioca</i>	<a href="http://rdfs.org/sioc/actions#">http://rdfs.org/sioc/actions#</a>	SIOC actions	3.4.4
<i>sioc_arg</i>	<a href="http://rdfs.org/sioc/argument#">http://rdfs.org/sioc/argument#</a>	SIOC argumentation	3.6.1
<i>sioc_t</i>	<a href="http://rdfs.org/sioc/types#">http://rdfs.org/sioc/types#</a>	SIOC Types	3.6.1
<i>sl</i>	<a href="http://vocab.deri.ie/scovollink#">http://vocab.deri.ie/scovollink#</a>	SCOVOLink	6.4.1.1
<i>sro</i>	<a href="http://salt.semanticsauthoring.org/ontologies/sro#">http://salt.semanticsauthoring.org/ontologies/sro#</a>	SALT rhetorics	3.3
<i>xhv</i>	<a href="http://www.w3.org/1999/xhtml/vocab#">http://www.w3.org/1999/xhtml/vocab#</a>	XHTML vocabulary	5.2.2
<i>xlink</i>	<a href="http://www.w3.org/1999/xlink">http://www.w3.org/1999/xlink</a>	XLink	C.1.2.1
<i>xsd</i>	<a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a>	XML Schema datatypes <sup>3</sup>	
<i>xsl</i>	<a href="http://www.w3.org/1999/XSL/Transform">http://www.w3.org/1999/XSL/Transform</a>	XSLT	

<sup>3</sup>The URI with a trailing hash is suitable for use in RDF. For XML namespaces, there is an alternative URI without that hash.





# Ontologies

## B.1 OMDoc

A high-level description of the OMDoc ontology, which mainly represents logical/functional structures of mathematical knowledge, can be found in [section 3.2.2](#). [Figure 3.1](#) shows the core of the class hierarchy and the main concrete properties at a glance; [figure 3.2](#) is more elaborate w.r.t. the property hierarchy. I have developed the ontology according to the methodology explained in [section 3.2.1](#), following the conceptual model of the OMDoc markup language, which has existed before and has been described in the OMDoc specification [[Koho6b](#)]. The Protégé [[Proc](#)] and Swoop [[KPS+06](#)] ontology development environments and the online OWL 2 validator [[Hor](#)] have been used for validating and debugging this ontology, as well as the others described in this chapter.

The OWL and OMDoc sources of the implementation – one file per ontology language and module of the OMDoc specification – are available at <https://svn.omdoc.org/repos/omdoc/trunk/owl/>. For compatibility with a wider range of semantic web tools, an all-in-one RDF/XML version generated from the OWL subset of the ontology has been published under the namespace URI <http://omdoc.org/ontology#> in compliance with best practices for publishing RDF vocabularies [[BPo8b](#)].

Tables [B.2](#) and [B.2](#) below list all classes and properties of the OMDoc ontology with their formalization details. In particular, their columns (from left to right) convey the following information:

**Hierarchy:** the hierarchy of classes or properties, respectively (i.e. *rdfs:subClassOf* or *rdfs:subPropertyOf*), denoted by indentation – plus domains and ranges of the properties, and inverse properties. Disjointness of classes – and, more rarely, properties – is not explicitly indicated; generally, however, all sibling classes (subclasses of the same superclass) have been declared disjoint. All classes and properties are in the OMDoc ontology namespace; the common prefix *oo:* is omitted.

**XML→RDF Translation:** the OMDoc XML structures that correspond to a class or property, i.e.:

**for Classes:** XPath expressions conforming to the XSLT *Pattern* production [[Kay07](#), section 5.5.2] (here: mostly referring to element nodes), from whose occurrences the

OMDoc→RDF translation (requirements specified in [section 3.7](#), implementation described in [section 8.1](#)) creates instances of the respective classes. When an XML node in an OMDoc document matches multiple patterns, the closest match according to the XSLT template rules [[Kay07](#), section 6] applies. For example, `<assertion type="theorem">` generates an instance of *Theorem* rather than *Assertion*, as `assertion[@type='theorem']` is a closer match.

**for Properties:** XPath expressions (mostly referring to attribute values), from which the OMDoc→RDF translation generates triples  $(s, p, o)$ , where  $p$  is the respective property. The subject  $s$  is the parent resource in terms of the translation algorithm specified in [section 8.1.1.1](#), and the object  $o$  is obtained by interpreting the given XPath expression as a URI (for object properties) or as an RDF literal value (for datatype properties).

Apart from the translation rules given here, which are specific to OMDoc, the OMDoc→RDF translation proceeds as described in [section 8.1.1.1](#).

**Inference:** Alternatively, for those classes or properties whose instances are not generated by the OMDoc→RDF translation but inferred from the ontology by a reasoner, the same column shows the respective *SR<sub>O</sub>IQ* axioms.

**Specification:** references to the OMDoc specification, for looking up those aspects of the semantics that the current formalization does not [yet] cover

Footnotes refer to exceptions from these rules.

Table B.1: Class hierarchy of the OMDoc ontology

Class	corresponding OMDoc XML element	specification section(s)
<i>MathKnowledgeItem</i>		
Theory level		
<i>Theory</i>	<sup>R</sup> <i>theory</i> <sup>1</sup>	15.6
Statement level		
<i>Statement</i> <sup>2</sup>		
<i>StatementInTheory</i> <sup>3</sup>		15
<i>ConstitutiveStatement</i>		15.2
<i>Axiom</i>	<sup>R</sup> <i>axiom</i> , <sup>I</sup> <i>omtext</i> [@type='axiom']	15.2.2
<i>Definition</i> <sup>4</sup>	<sup>R</sup> <i>definition</i> <sup>5</sup> , <sup>I</sup> <i>omtext</i> [@type='definition']	15.2.4
<i>Import</i>	<sup>R</sup> <i>imports</i> <sup>6</sup>	15.6.1
<i>Symbol</i>	<sup>R</sup> <i>symbol</i>	15.2.1
<i>NonConstitutiveStatement</i>		15.3, 15.4
<i>AlternativeDefinition</i>	<sup>R</sup> <i>alternative</i>	15.3.3
<i>Assertion</i>	<sup>R</sup> <i>assertion</i> , <sup>I</sup> <i>omtext</i> [@type='assertion']	15.3.1
<i>ProvenAssertion</i>	$\sqsubseteq \exists \text{provedBy.Proof}$	
<i>Corollary</i>	<sup>R</sup> <i>assertion</i> [@type='corollary'], <sup>I</sup> <i>omtext</i> [@type='corollary']	
<i>Lemma</i>	<sup>R</sup> <i>assertion</i> [@type='lemma'], <sup>I</sup> <i>omtext</i> [@type='lemma']	
<i>Proposition</i>	<sup>R</sup> <i>assertion</i> [@type='proposition'], <sup>I</sup> <i>omtext</i> [@type='proposition']	
<i>Theorem</i>	<sup>R</sup> <i>assertion</i> [@type='theorem'], <sup>I</sup> <i>omtext</i> [@type='theorem']	
<i>UnProvenAssertion</i>		

continued on next page

<sup>1</sup>An <sup>R</sup> superscript in front of an XML element name denotes that the corresponding RDF resource will be assigned a formality degree of *Rigorous*, <sup>C</sup> denotes *Computerized*, whereas <sup>I</sup> denotes *Informal*. Informal statement elements (*omtext*) are generally covered in section 14.3 of the OMDoc specification, but they have the same semantics as their formal counterparts.

<sup>2</sup>This is the most general statement type, not only subsuming statements that occur on the proper “statement level”, but also proof-local statements.

<sup>3</sup>This is what would usually be called a statement.

<sup>4</sup>The different types of definitions – simple, implicit, recursive – will be modeled as subclasses of a class *GeneralDefinition*, which is the common superclass both of this class and of *ProofLocalDefinition*, a definition that constitutes a step of a proof and is local to that proof. See below for both classes.

<sup>5</sup>The specific case *definition*[@type='informal'] also denotes an informal definition.

<sup>6</sup>will be renamed to *import* in OMDoc 1.6

Table B.1: Class hierarchy of the OMDoc ontology (*continued from previous page*)

Class	corresponding OMDoc XML element	specification section(s)
<i>AssumptionAssertion</i> <sup>7</sup>	$R_{\text{assertion}}[@\text{type}=\text{'assumption'}],$ $I_{\text{omtext}}[@\text{type}=\text{'assumption'}]$	
<i>Conjecture</i>	$R_{\text{assertion}}[@\text{type}=\text{'conjecture'}],$ $I_{\text{omtext}}[@\text{type}=\text{'conjecture'}]$	
<i>Formula</i>	$R_{\text{assertion}}[@\text{type}=\text{'formula'}],$ $I_{\text{omtext}}[@\text{type}=\text{'formula'}]$	
<i>Obligation</i>	$R_{\text{assertion}}[@\text{type}=\text{'obligation'}],$ $I_{\text{omtext}}[@\text{type}=\text{'obligation'}]$	
<i>Postulate</i>	$R_{\text{assertion}}[@\text{type}=\text{'postulate'}],$ $I_{\text{omtext}}[@\text{type}=\text{'postulate'}]$	
<i>Rule</i>	$R_{\text{assertion}}[@\text{type}=\text{'rule'}],$ $I_{\text{omtext}}[@\text{type}=\text{'rule'}]$	
<i>RefutedAssertion</i>	$\sqsubseteq \exists \text{refutedBy.Example}$	
<i>FalseConjecture</i>	$R_{\text{assertion}}[@\text{type}=\text{'false-conjecture'}],$ $I_{\text{omtext}}[@\text{type}=\text{'false-conjecture'}]$	
<i>Example</i>	$R_{\text{example}}, I_{\text{omtext}}[@\text{type}=\text{'example'}]$	15.4
<i>Proof</i>	$C_{\text{proofobject}}, R_{\text{proof}},$ $I_{\text{omtext}}[@\text{type}=\text{'proof'}]$	17
<i>NotationDefinition</i>	$R_{\text{notation}}, I_{\text{omtext}}[@\text{type}=\text{'notation'}]$	19.3 <sup>8</sup>
<i>Type</i>		15.2.3, 15.3.2
<i>DeclaredType</i>	$R_{\text{symbol/type}}$	15.2.3
<i>AssertedType</i> <sup>9</sup>	$R_{\text{type}}$	15.3.2
Substatement level: proof steps		
<i>ProofStep</i> <sup>10</sup>		17.1
<i>ProofLocalStatement</i> <sup>11</sup>		
<i>DerivationStep</i>	$I_{\text{omtext}}[@\text{type}=\text{'derive'}]$	
<i>DerivedConclusion</i> <sup>12</sup>	$R_{\text{derive}}[@\text{type}=\text{'conclusion'}]$	
<i>Gap</i>	$R_{\text{derive}}[@\text{type}=\text{'gap'}]$	
<i>continued on next page</i>		

<sup>7</sup> name chosen to disambiguate from *Assumption* (a part of a sequent)<sup>8</sup> We actually consider notation definitions as they will be given in OMDoc 1.3 and 1.6; see sections 2.4.5.2 and 2.4.5.3, but the ontological concept is sufficiently general to also comprise notations in the way of OMDoc 1.2.<sup>9</sup> also a subclass of *NonConstitutiveStatement*<sup>10</sup> The classes in this section generally correspond to XML elements that occur as descendants of a *proof* element, i.e. the actual XPath for, e.g., *ProofLocalDefinition*, is *proof//symbol*.<sup>11</sup> also a subclass of *Statement*<sup>12</sup> name chosen to disambiguate from *Conclusion* (a part of a sequent)

Table B.1: Class hierarchy of the OMDoc ontology (*continued from previous page*)

Class	corresponding OMDoc XML element	specification section(s)
<i>Hypothesis</i> <sup>13</sup>	<sup>R</sup> <i>hypothesis</i> , <sup>I</sup> <i>omtext</i> [@type='hypothesis']	
<i>ProofLocalDefinition</i>	<sup>R</sup> <i>definition</i> , <sup>I</sup> <i>omtext</i> [@type='definition']	
<i>ProofLocalSymbol</i>	<sup>R</sup> <i>symbol</i>	
<i>ProofText</i>	<sup>I</sup> <i>omtext</i>	
<i>NestedProof</i> <sup>14</sup>	<sup>R</sup> <i>derive/method/proof</i> , <sup>C</sup> <i>derive/method/proofobject</i> Substatement level: properties	17.2
<i>Property</i>	<sup>R</sup> <i>FMP</i> , <sup>I</sup> <i>CMP</i>	14.1, 14.2
<i>Statement</i>		
<i>SequentPart</i>		14.2
<i>Assumption</i>	<sup>R</sup> <i>assumption</i>	
<i>Conclusion</i>	<sup>R</sup> <i>conclusion</i>	
Definition types		
<i>Statement</i>		
<i>AnyLevelStatement</i>		
<i>GeneralDefinition</i>	<sup>R</sup> <i>definition</i>	15.2.4
<i>SimpleDefinition</i>	<sup>R</sup> <i>definition</i> [@type='simple'] <sup>15</sup>	
<i>ImplicitDefinition</i>	<sup>R</sup> <i>definition</i> [@type='implicit']	
<i>RecursiveDefinition</i>		
<i>InductiveDefinition</i>	<sup>R</sup> <i>definition</i> [@type='inductive']	
<i>PatternBasedDefinition</i>	<sup>R</sup> <i>definition</i> [@type='pattern']	

<sup>13</sup>This is the proof-local counterpart of *Axiom*. *Hypothesis* and the following proof-local statement classes have common superclasses with their statement-level counterparts:

$$\left. \begin{array}{l} \textit{Hypothesis} \\ \textit{Axiom} \end{array} \right\} \sqsubseteq \textit{GeneralAxiom} \quad \left. \begin{array}{l} \textit{ProofLocalDefinition} \\ \textit{Definition} \end{array} \right\} \sqsubseteq \textit{GeneralDefinition} \quad \left. \begin{array}{l} \textit{ProofLocalSymbol} \\ \textit{Symbol} \end{array} \right\} \sqsubseteq \textit{GeneralSymbol} \quad \left. \begin{array}{l} \textit{GeneralAxiom} \\ \textit{GeneralDefinition} \\ \textit{GeneralSymbol} \end{array} \right\} \sqsubseteq \textit{AnyLevelStatement} \sqsubseteq \textit{Statement}$$

<sup>14</sup>used for representing substeps of structured proofs, has a common superclass with *Proof*:

$$\left. \begin{array}{l} \textit{NestedProof} \\ \textit{Proof} \end{array} \right\} \sqsubseteq \textit{GeneralProof} \sqsubseteq \textit{AnyLevelStatement} \sqsubseteq \textit{Statement}$$

<sup>15</sup>This is the default value of the attribute.

Table B.2: Object and datatype properties of the OMDoc ontology

Property	domain	range	corresponding OMDoc XML element/attribute	Specification section(s)
General relations				
<sup>T</sup> <i>dependsOn</i> <sup>16</sup>	<i>MathKnowledge-Item</i>	<i>MathKnowledge-Item</i>		
<sup>T</sup> <i>wellFormedNessDependsOn</i>	<sup>17</sup>			
<i>validityDependsOn</i>	<sup>18</sup>			
<i>presentationDependsOn</i>				
<sup>T</sup> <i>hasPart</i> <sup>19</sup> <i>=isPartOf</i> <sup>-120</sup> <i>hasDirectPart</i> <i>=isDirectPartOf</i> <sup>-1</sup>	<i>MathKnowledge-Item</i>	<i>MathKnowledge-Item</i>	<sup>21</sup>	
<i>formalityDegree</i> <sup>22</sup>	<i>Statement</i> $\sqsubset$ <i>Property</i> $\sqsubset$ <i>ProofStep</i>	<i>FormalityDegree</i>		
<sup>T</sup> <i>verbalizes</i> <i>=formalizes</i> <sup>-1</sup>	<i>MathKnowledge-Item</i>	<i>MathKnowledge-Item</i>	<i>@verbalizes</i>	14.3, 14.4, 14.6
<i>justifiedBy</i>	<i>MathKnowledge-Item</i>	<i>AnyLevelStatement</i>		
Theory level and theory $\leftrightarrow$ statement relations				
<sup>T</sup> <i>wellFormedNessDependsOn</i> (see above)				
<i>continued on next page</i>				

<sup>16</sup>T denotes a transitive property.<sup>17</sup>As this property only has one subproperty chain so far but additional validity-related dependencies will be added in future, domain and range have been left unrestricted so far.<sup>18</sup>As this property only has one subproperty so far but additional presentation-related dependencies may be added in future, domain and range have been left unrestricted so far.<sup>19</sup>declared as subproperty of *dct:hasPart*<sup>20</sup>declared as subproperty of *dct:isPartOf*<sup>21</sup>*isDirectPartOf* is used whenever a more specific relation for relating an element to its parent is not available; for example, *Statements* are related to their home *Theory* via *homeTheoryOf* (see below), whereas *isDirectPartOf* is used for non-statement children of a *theory* element.<sup>22</sup>See the <sup>I</sup>, <sup>R</sup>, and <sup>C</sup> annotations in table B.1.

Table B.2: Object and datatype properties of the OMDoc ontology (*continued from previous page*)

Property	domain	range	corresponding OMDoc XML element/attribute	Specification section(s)
${}^T \text{imports}$ $= \text{importedBy}^{-1}$	<i>Theory</i>	<i>Theory</i>	<i>theory/imports/</i> $\leftrightarrow$ <i>@from</i> <sup>23</sup> <i>hasImport</i> $\circ$ <i>importsFrom</i> (s. bel.)	15.6.1, 18.1
${}^T \text{hasDirectPart}$ (see above) <i>homeTheory</i> $= \text{homeTheoryOf}^{-1}$	<i>Statement</i> $\sqcup$ <i>Theory</i>	<i>Theory</i>	<i>ancestor::theory</i>	15.6.1, 18.1
<i>hasImport</i> <sup>24</sup>	<i>Theory</i>	<i>Import</i>	<i>theory/imports</i>	15.6.1, 18.1
<i>importsFrom</i>	<i>Import</i>	<i>Theory</i>	<i>imports/@from</i>	15.6.1, 18.1
Statement interrelations				
<i>type</i> <i>declaredType</i>	<i>Symbol</i>	<i>Type</i> <i>DeclaredType</i>	<i>symbol/@for</i> , <i>type/@for</i> <sup>-1</sup>	15.2.3, 15.3.2 15.2.3
<i>assertedType</i> <sup>25</sup>		<i>AssertedType</i>	<i>type[@just-by]/@for</i> <sup>-1</sup>	15.3.2
<i>justifiedBy</i> (see above) <i>typeJustifiedBy</i>	<i>AssertedType</i>	<i>Assertion</i>	<i>type/@just-by</i>	15.3.2
<i>defines</i> $= \text{hasDefinition}^{-1}$	<i>Definition</i> $\sqcup$ <i>Property</i> $\sqcap$ <i>Informal-KnowledgeItem</i>	<i>Symbol</i>	<i>definition/@for</i> <sup>26</sup> , <i>omtext[@type=</i> $\leftrightarrow$ <i>'definition']/</i> $\leftrightarrow$ <i>@for</i> , <i>CMP//term</i> $\leftrightarrow$ <i>[@role=</i> $\leftrightarrow$ <i>'definiendum']</i>	15.2.1, 15.2.4
${}^T \text{wellFormedNessDependsOn}$ (see above) <i>hasOccurrenceOfInType</i> $= \text{occursInTypeOf}^{-1}$	<i>Symbol</i>	<i>Symbol</i>	<i>type</i> $\circ$ <i>usesSymbol</i> (s. below)	

*continued on next page*<sup>23</sup>will be renamed to *import* in OMDoc 1.6<sup>24</sup>subproperty of *homeTheoryOf*<sup>25</sup>This is not automatically a subproperty of *type*. An asserted type of a symbol is an actual type only if the corresponding assertion has a grounded proof.<sup>26</sup>OMDoc 1.6 will also have *definition* as a child element of *symbol*.



Table B.2: Object and datatype properties of the OMDoc ontology (*continued from previous page*)

Property	domain	range	corresponding OMDoc XML element/attribute	Specification section(s)
<i>hasOccurrence-OfInDefinition</i> <i>=occursIn-DefinitionOf</i> <sup>-1</sup>			<i>hasDefinition</i> ◦ <i>usesSymbol</i>	
<i>rendersSymbol</i> <i>=hasNotation-Definition</i> <sup>-1</sup>	<i>NotationDefinition</i>	<i>Symbol</i>	<i>prototype/</i> ↔ <i>descendant::</i> ↔ <i>om:OMS</i>	19.3
<sup>T</sup> <i>presentationDependsOn</i> (see above) <i>possiblyUses-Notation</i> <sup>27</sup>		<i>NotationDefinition</i>	<i>usesSymbol</i> ◦ <i>hasNotationDefinition</i>	
<i>exemplifies</i> <i>=exemplifiedBy</i> <sup>-1</sup>	<i>Example</i>	<i>Symbol</i> □ <i>Definition</i> □ <i>Alternative-Definition</i> □ <i>Axiom</i> □ <i>Assertion</i>	<i>example/@for</i> , <i>omtext[@type=</i> ↔ <i>'example']/@for</i>	15.4
<i>corroborates</i> <i>=corroboratedBy</i> <sup>-1</sup>		<i>Assertion</i>	<i>example[@type=</i> ↔ <i>'for']/@for</i>	15.4
<i>refutes</i> <sup>28</sup> <i>=refutedBy</i> <sup>-1</sup>			<i>example[@type=</i> ↔ <i>'against']/@for</i>	15.4
<i>proves</i> <i>=provedBy</i> <sup>-1</sup>	<i>Proof</i>	<i>ProvenAssertion</i>	<i>proof/@for</i> , <i>omtext[@type=</i> ↔ <i>'proof']/@for</i>	17.1
Proof steps				
<sup>T</sup> <i>hasDirectPart</i> (see above) <i>hasStep</i> <i>hasConclusion</i>	<i>Proof</i>	<i>ProofStep</i> <i>Derived-Conclusion</i>	<i>proof/*</i> <i>proof/derive</i> ↔ <i>[@type=</i> ↔ <i>'conclusion']</i>	17.1, 17.2 17.1
<i>justifiedBy</i> (see above) <i>stepJustifiedBy</i>	<i>DerivationStep</i>	<i>Statement</i>		17.2

*continued on next page*<sup>27</sup> same as *usesSymbol*<sup>28</sup> disjoint with *corroborates*

Table B.2: Object and datatype properties of the OMDoc ontology (*continued from previous page*)

Property	domain	range	corresponding OMDoc XML element/attribute	Specification section(s)
<i>stepLocally-JustifiedBy</i>		<i>NestedProof</i> $\sqcup$ <i>ProofLocal-Statement</i>		17.2
<i>stepJustified-ByPrecedingStep</i>		<i>ProofLocal-Statement</i>	<i>derive/method/</i> $\leftrightarrow$ <i>premise/@xref</i> <sup>29</sup>	17.2
<i>stepJustified-BySubProof</i> <sup>30</sup>		<i>NestedProof</i>	<i>derive/method/</i> $\leftrightarrow$ <i>proof</i> , <i>derive/method/</i> $\leftrightarrow$ <i>proofobject</i>	17.2
<i>stepExternally-JustifiedBy</i> <sup>31</sup>		<i>Definition</i> $\sqcup$ <i>Axiom</i> $\sqcup$ <i>Assertion</i>	<i>derive/method/</i> $\leftrightarrow$ <i>premise/@xref</i> <sup>32</sup>	17.2
<i>validityDependsOn</i>	<i>Proof</i>	<i>Definition</i> $\sqcup$ <i>Axiom</i> $\sqcup$ <i>Assertion</i>	<i>hasStep</i> $\circ$ <i>stepJustifiedBy</i>	
Properties				
<sup>T</sup> <i>hasDirectPart</i> (see above)				
<i>hasProperty</i>	<i>Statement</i>	<i>Property</i>	<i>*/CMP */FMP</i>	14.1, 14.2
<i>assumes</i>	<i>Property</i>	<i>Assumption</i>	<i>FMP/assumption</i>	14.2
<i>concludes-With</i> <sup>33</sup>		<i>Conclusion</i>	<i>FMP/conclusion</i>	14.2
<i>usesSymbol</i> <i>=occursIn</i> <sup>-1</sup>	<i>SimpleDefinition</i> $\sqcup$ <i>DeclaredType</i> $\sqcup$ <i>Property</i> $\sqcup$ <i>SequentPart</i>	<i>GeneralSymbol</i>	<i>om:OMS</i> <sup>34</sup> , <i>CMP//term</i> $\leftrightarrow$ <i>[@role='definiens']</i>	14.2

*continued on next page*<sup>29</sup>pointing to a step of the current proof<sup>30</sup>disjoint with *stepJustifiedByPrecedingStep*<sup>31</sup>disjoint with *stepLocallyJustifiedBy*<sup>32</sup>pointing to a statement outside of the current proof<sup>33</sup>disjoint with *assumes*<sup>34</sup>most commonly as a descendant of a formal property (*FMP*), sometimes also as a descendant of a statement that does not have an intermediate “property” level, such as a simple definition

Table B.2: Object and datatype properties of the OMDoc ontology (*continued from previous page*)

Property	domain	range	corresponding OMDoc XML element/attribute	Specification section(s)
<sup>D</sup> <i>hasText</i> <sup>35</sup>	<i>Property</i> □ <i>Informal- KnowledgeItem</i>	<i>rdfs:Literal</i>	<i>CMP//text()</i>	14.1

## B.2 OpenMath CDs

I have developed an OWL formalization and implementation of the abstract information model of OpenMath CDs; the latter has existed before [BCC+04, chapter 4]. Section 3.2.3 provides a high-level description of this ontology; figure 3.3 shows most of the class hierarchy and the concrete properties at a glance. Its source is available at <http://svn.openmath.org/OpenMath/OpenMath3/owl/><sup>36</sup>.

The tables below are structured analogously to those given for the OMDoc ontology (cf. appendix B.1), with the following differences:

**Hierarchy:** All classes and properties are in the OpenMath CD ontology namespace; the common prefix *omo:* is omitted here.

**Translation and Inference:** Some classes are enumerations of their instances. The string values permitted for the text content of the XML element corresponding to such a class correspond to instances. For example, the OpenMath→RDF translation generates a triple (*s*, *role*, *Binder*) when it encounters the element `<Role>binder</Role>` among the children of a definition of a symbol with the URI *s*.

Table B.3: Class hierarchy of the OpenMath CD ontology

Class	corresponding OpenMath XML element
<i>OpenMathConcept</i> <i>Composite</i>	
Dictionary level	
<i>ContentDictionaryGroup</i>	<i>cdg:CDGroup</i>

*continued on next page*

<sup>35</sup>D denotes a datatype property.

<sup>36</sup>In the transition from OpenMath 2 to the next version, the OpenMath Subversion repository is currently (spring 2011) undergoing a heavy restructuring. All of these URLs were valid at the time of submitting this thesis.

Table B.3: Class hierarchy of the OpenMath CD ontology (*continued from previous page*)

Class	corresponding OpenMath XML element
<i>Dictionary</i>	
<i>ContentDictionary</i>	<i>cd:CD</i>
<i>NotationDictionary</i>	<i>o:notations</i>
<i>SignatureDictionary</i>	<i>cds:CDSignatures</i>
Symbol definition level	
<i>Composite</i>	
<i>SymbolDefinition</i>	<i>cd:CDDefinition</i>
<i>NotationDefinition</i>	
<i>Signature</i>	<i>cds:Signature</i>
Sub-symbol definition level	
<i>OpenMathConcept</i>	
<i>Composite</i>	
<i>Property</i>	
<i>Example</i>	<i>cd:Example</i>
<i>CommentedPart</i> <sup>37</sup>	<i>cd:CMP</i>
<i>FormalPart</i>	<i>cd:FMP</i>
Others	
<i>OpenMathConcept</i>	
<i>CDBase</i>	<i>cd:CDBase</i>
<i>Status</i>	<i>cd:CDStatus</i>
Instances: <i>Official, Experimental, Private, Obsolete</i>	
<i>Role</i>	<i>cd:Role</i>
Instances: <i>Binder, Attribution, SemanticAttribution, Error, Application, Constant</i>	
<i>Version</i>	<i>cd:CDVersion, cd:CDRevision</i> <sup>38</sup>
<i>Status</i>	<i>cd:CDStatus</i>

<sup>37</sup> As it is unlikely that there will be more degrees of formality for OpenMath properties, they are modeled by fixed classes

<sup>38</sup> These OpenMath XML elements generate an instance of *Version*, whose *major* and *minor* properties are assigned as explained in [table B.4](#).

Table B.4: Object and datatype properties of the OpenMath CD ontology

Property	domain	range	corresponding OpenMath XML element/ attribute
General relations			
<sup>T</sup> <i>comprises</i> <sup>3940</sup>	<i>OpenMath- Composite</i>	<i>OpenMath- Composite</i>	
<sup>T</sup> <i>hasPart</i> <i>hasDirectPart</i>			
Content dictionary level			
<i>status</i>	<i>Content- Dictionary</i>	<i>Status</i>	<i>CDStatus</i>
<sup>D</sup> <i>reviewDate</i>	<i>Content- Dictionary</i> $\sqcup$ <i>Signature- Dictionary</i>	<i>xsd:date</i>	<i>CDReviewDate</i>
<sup>T</sup> <i>version</i>	<i>Content- Dictionary</i> $\sqcup$ <i>Content- Dictionary- Group</i>	<i>Version</i>	<i>CDVersion</i> , <i>CDRevision</i>
<i>containsNotationsFor</i> <i>=hasNotation- Dictionary</i> <sup>-1</sup>	<i>Notation- Dictionary</i>	<i>Content- Dictionary</i>	
<i>containsSignaturesFor</i> <i>=hasSignature- Dictionary</i> <sup>-1</sup>	<i>Signature- Dictionary</i>	<i>Content- Dictionary</i>	
<i>typeSystem</i>		<i>Content- Dictionary</i> $\sqcup$ <i>Content- Dictionary- Group</i>	
<sup>D</sup> <i>major</i>	<i>Version</i>	<i>xsd:non- NegativeInteger</i>	<i>CDVersion</i>
continued on next page			

<sup>39</sup> All properties are in the OpenMath ontology namespace, commonly prefixed *omo*.<sup>40</sup> <sup>T</sup> denotes a transitive property.

Table B.4: Object and datatype properties of the OpenMath CD ontology

Property	domain	range	corresponding OpenMath XML element/ attribute
$D_{minor}$	<i>Version</i>	<i>xsd:non-NegativeInteger</i>	<i>CDRevision</i>
<i>base</i>	<i>Content-Dictionary</i>	<i>CDBase</i>	<i>CDBase</i>
$T_{uses}$	<i>Content-Dictionary</i>	<i>Content-Dictionary</i>	
<i>usesDirectly</i>	$definesSymbol \circ \left\{ \begin{array}{l} exemplifiedBy \\ hasProperty \\ \circ hasFormalPart \end{array} \right\} \circ usesSymbol \circ definedIn^{41}$		
<i>comprises</i> (see above)			
<i>containsContent-Dictionary</i>	<i>Content-Dictionary-Group</i>	<i>Content-Dictionary</i>	
<i>hasDirectPart</i> (see above)			
<i>definesSymbol</i>	<i>Content-Dictionary</i>	<i>Symbol-Definition</i>	
$=definedIn^{-1}$			
<i>containsNotation-Definition</i>	<i>Notation-Dictionary</i>	<i>NotationDefinition</i>	
<i>containsSignature</i>	<i>Signature-Dictionary</i>	<i>Signature</i>	
Symbol level			
<i>hasPart</i> (see above)			
<i>hasDirectPart</i> (see above)			
<i>exemplifiedBy</i>	<i>Symbol-Definition</i>	<i>Example</i>	
$=exemplifies^{-1}$			
<i>hasProperty</i>		<i>Property</i>	
<i>role</i>	<i>Symbol-Definition</i>	<i>Role</i>	
<i>rendersSymbol</i>	<i>NotationDefinition</i>	<i>Symbol-Definition</i>	
$=hasNotationDefinition^{-1}$			

continued on next page

<sup>41</sup>Representing this in XML using *CD/CDUses* is deprecated.

Table B.4: Object and datatype properties of the OpenMath CD ontology

Property	domain	range	corresponding OpenMath XML element/ attribute
$\text{typesSymbol}$ $=\text{type}^{-1}$	<i>Signature</i>	<i>Symbol- Definition</i>	
Property level			
<i>hasPart</i> (see above)			
<i>hasDirectPart</i> (see above)			
<i>hasCommentedPart</i>	<i>Property</i>	<i>CommentedPart</i>	
<i>hasFormalPart</i>		<i>FormalPart</i>	
<sup>D</sup> <i>hasText</i>	<i>CommentedPart</i> $\sqsubseteq$ <i>Example</i>	<i>Symbol- Definition</i>	
<i>usesSymbol</i>	<i>FormalPart</i> $\sqsubseteq$ <i>Example</i>	<i>Symbol- Definition</i>	

### B.3 Mathematics-specific Issue and Solution Types

The conceptualization and formalization of my extension of a DILIGENT-style argumentation ontology (here, concretely, the SIOC argumentation module) by mathematics-specific issue and idea (solution) types have been explained in [section 3.6.2](#). This mathematical extension is currently maintained as a part of the SWiM wiki (cf. [chapter 9](#)) and available at <https://svn.salzburgresearch.at/svn/kiwi/IkeWiki/branches/SWiM/trunk/WEB-INF/ontologies/matharg/>.

The table below conveys the following information:

**Hierarchy:** the class hierarchy, denoted by indentation; first issue types, then idea types. All classes in the first column are in the mathematical argumentation namespace, those in the second column are in the OMDoc ontology namespace; the prefixes *matharg:* and *oo:* are omitted here.

**Applicability** of the respective issue/idea type: Issue types are only applicable to certain knowledge item types. Idea types are only applicable to certain issue types. In some cases, their usage additionally requires a particular knowledge item type, which is stated as footnotes. The formalization of the *matharg:appliesToKnowledgeItemType* and *matharg:appliesToIssueType*, which has been explained in [section 3.6.2.2](#), exceeds the expressivity of OWL. Therefore, it has not been implemented in the ontology, but will be implemented in a future implementation in the OMDoc language.

**Relevance** of the respective issue/idea type to domain experts, according to the survey explained in [section 3.6.2.1](#)



Table B.5: Mathematics-specific issue and idea (solution) types

Issue type	applies to knowledge item type	relevance
<i>Incomprehensible</i>	<i>MathKnowledgeItem</i>	9/20
<i>Wrong</i>	<i>Assertion, Example, Proof</i>	10/20
<i>UncertainWhetherTrue</i>	<i>Assertion</i>	9/20
<i>UnclearHowToUtilize</i>	<i>Symbol, Assertion</i>	4/20
<i>UnclearWhetherUseful</i>	<i>Symbol, Assertion, Definition, Axiom</i>	8/20
<i>InappropriateForDomain</i>	<i>Example, Definition, Proof, omo:NotationDefinition</i>	4/20
<i>UncommonStyle</i>	<i>Example, Definition, Proof, omo:NotationDefinition</i>	8/20
<i>RelationUnclear</i>	<i>Proof, Example, omo:NotationDefinition</i>	7/20
<i>Underspecified</i>	<i>Definition, Axiom, Assertion</i>	9/20
<i>Overspecified</i>	<i>Definition, Axiom, Assertion</i>	6/20
<i>TooManySubparts</i>	<i>Theory, Assertion, Example, Definition</i>	5/20
<i>Reinvention</i>	<i>Theory, Assertion, Definition, Axiom</i>	9/20
Idea type	applies to Issue type	relevance
<i>ImproveThis</i>	<i>Issue</i>	4 <sup>2</sup>
<i>FixSemantics</i>	<i>Issue</i>	12/21
<i>ImproveInformally</i>	<i>Issue</i>	11/21
<i>ImproveRelated</i> <sup>42</sup>	<i>Incomprehensible, UncertainWhetherTrue, UnclearHowToUtilize, UnclearWhetherUseful</i>	7/21
<i>CreateRelated</i> <sup>42</sup>	<i>Incomprehensible, UncertainWhetherTrue, UnclearHowToUtilize, UnclearWhetherUseful</i>	8/21
<i>ProvideExample</i>		4 <sup>3</sup>
<i>CreateAlternative</i> <sup>44</sup>	<i>Incomprehensible, Wrong, UncommonStyle, RelationUnclear, InappropriateForDomain</i>	2/21
<i>Split</i>	<i>TooManySubparts</i>	11/21
<i>RemoveParts</i>	<i>TooManySubparts, Reinvention</i>	9/21
<i>ReplacePartsByReferences</i>	<i>TooManySubparts, Reinvention</i>	5/21
<i>FactorOutParts</i> <sup>43</sup>	<i>TooManySubparts, Reinvention</i>	
<i>IntegrateOthers</i>	<i>Issue</i>	3/21
<i>KeepAsBadExample</i> <sup>45</sup>	<i>Issue</i>	1/21
<i>Delete</i>	<i>Issue</i>	10/21

<sup>42</sup>applies to the following knowledge item types: *Assertion, Definition, Symbol, Axiom*<sup>43</sup>not covered by the survey<sup>44</sup>applies to the following knowledge item types: *Proof, Example, omo:NotationDefinition*<sup>45</sup>applies to the following knowledge item types: *Assertion*



# Algorithm and Implementation Details

This appendix explains technical details of our algorithms as well as our implementations of the services, architectures, and systems introduced in [part III](#), in the order of the original chapters.

## C.1 Primitive Services

### C.1.1 TinyMCE+Sentido, a Visual Editor for Semantic Markup, Content Markup Formulæ, and Metadata

#### C.1.1.1 General Semantic Markup

We have realized our approach of using a WYSIWYG HTML editor for semantic markup, as introduced in [section 6.2.3](#), as a plugin for the TinyMCE editor [[Tin](#)]. The translation between OMDoc and OpenMath CD semantic markup and HTML for the editor has been implemented in two XSLT stylesheets. The editor currently leaves presentational HTML markup untranslated. In the setting of the SWiM semantic wiki (cf. [chapter 9](#)), where the editor has been used so far, I considered that rather helpful than harmful. The XSLT stylesheets that render OMDoc for presentation again leave this HTML untouched, thus providing an easy possibility to author presentation markup exactly as it will be displayed in the rendered view on a document. Our plugin provides additional toolbar buttons with submenus for inserting semantic markup; there is one menu item per supported XML element, plus a button for deleting semantic markup (cf. [figure 6.1](#)).

#### C.1.1.2 Integrating a Formula Editor

For editing formulæ, as described in [section 6.2.4](#), we have integrated the Sentido formula editor. It has originated in the Sentido mathematical environment [[GPo6b](#)], a prototype of a complete OMDoc development environment, but then also been made available as a standalone component. This component has so far been used for formulating queries for the MathWebSearch formula search engine [[KAJ+o8](#)]. The Sentido plugin for TinyMCE recognizes unparsed serialized OpenMath XML strings in *span* elements annotated with a special CSS class. Actual OpenMath XML would

have corrupted the content, as TinyMCE assumes HTML-only content, and, in the context of the SWiM semantic wiki, it would have interfered with the integrating HTML-based environment. Templates performing this serialization/parsing were included into the above-mentioned XSLT stylesheets. The Sentido plugin parses any serialized OpenMath that it recognizes and translates it into its own internal QMath linear syntax. Inside the formula editing popup window, the variant of linear syntax can be switched temporarily, in order to enable copy/paste of expressions from external systems using any of the supported syntaxes (cf. [figure 6.1](#) on page 190). On saving the content of the TinyMCE widget, the Sentido plugin parses the linear representation of each formula and serializes it as an OpenMath XML string once more, which the integrating environment finally has to parse into OpenMath XML.

The tool palettes in the visual formula editor window are currently hand-crafted in XHTML. They include all the symbols from the MathML CD group of the official OpenMath 2 CDs, plus a proof-of-concept extension palette of OWL-DL constructs, as shown in [figure 6.1](#) on page 190. Many symbols are represented by Unicode characters in QMath. They look nice but may be hard to enter with a conventional keyboard layout; therefore, Sentido's QMath context definitions provide ASCII alternatives for most of them, e.g. “inf” and “∞”.

Undo/redo in the TinyMCE widget is handled by TinyMCE. Inside the linear input field in the formula editor window, it is handled by the browser, which is sufficient, as changes in that field are parsed back immediately. Once a formula has been edited in the formula editor window, all changes made there become one undo step for TinyMCE.

One important OpenMath construct that Sentido does not currently support is the CDBase of a symbol. The translation XSLT stylesheet employed by the SWiM semantic wiki detects formulæ containing CDBase and other unsupported constructs and translates them to a custom linear syntax – wrapped into a different *span* –, which roughly resembles the abstract notation introduced in [section 2.4.3](#) and particularly supports arbitrary CDBases.<sup>1</sup> Formulæ in that syntax are directly editable in TinyMCE.

### C.1.1.3 Editing Notation Definitions

Our current editor implementation merely supports editing notation definitions as general markup, the side-by-side arrangement of the prototype and rendering component of pattern-based notation definitions (cf. [section 6.2.3](#)) being the only exception. Partial support for the linear notation definition syntax presented in [section 6.2.5](#), a superset of the above-mentioned linear OpenMath syntax, has been implemented in the JOMDoc library [[Jom](#)].

### C.1.2 JOMDoc, a Semantics-Preserving Renderer

Transformations from semantic XML markup to XHTML+MathML have traditionally often been implemented in XSLT (cf. [section 2.4.5.1](#)). Mathematical objects sometimes receive special treatment due to their complexity. This section explains how the JOMDoc renderer [[Jom](#)], also following such a hybrid approach, renders mathematical documents from OMDoc or OpenMath CD sources to XHTML+MathML+RDFa in a semantics-preserving way, according to the requirements specified in [section 6.4.2](#).

---

<sup>1</sup>Popcorn would not have been an option here, as it does not yet support CDBase either [[HRo9b](#)].

Listing C.1: Presentation MathML symbols with direct links

```

<semantics>
  ...
  <mo
    xlink:href="http://wiki.openmath.org/?uri=http://www.openmath.org/cd/arith1%2Bplus"
    xlink:type="simple"
    xref="#plus">+</mo>
  ...
  <annotation-xml encoding="application/openmath+xml">
    ...
    <OMS id="plus" cd="arith1" name="plus"/>
    ...
  </annotation-xml>
</semantics>

```

### C.1.2.1 Mathematical Objects

JOMDoc directly implements the pattern matching and rendering algorithm for the notation definitions introduced in [section 2.4.5.2](#) in Java. The renderer generates parallel content/presentation markup according to the requirements established in [section 6.4.2.5](#). It supports the pattern matching notation definitions introduced in [section 2.4.5.2](#) and handles brackets and operator precedences as described in [section 2.4.5.5](#). The original rendering process of OMDoc 1.2 (cf. [\[Koho6b, section 25.1\]](#) and [\[Lano7a, section 3.5\]](#)) – translating the (declarative) notation definitions to XSLT, collecting those generated XSLT stylesheets that would cover all symbols used in the document to be rendered, and finally applying the overall stylesheet (with a static part for statement-, theory-, and document-level elements) – had turned out to be too error-prone and hard to debug.

For rendering whole mathematical objects in OpenMath, our implementation reuses the OpenMath CD XSLT stylesheets by DAVID CARLISLE: Besides Presentation MathML, they generate a view of the OpenMath, Content MathML, and Popcorn source of an object, and a rendering with all operators in prefix notation. Buttons enable the user to show or hide these alternatives on demand, as shown in [figure 9.1](#).

JOMDoc offers a rich repertoire of strategies for collecting notation definitions and selecting among alternative ones [\[KMRo8; Mül10a\]](#). Additionally, a system that integrates JOMDoc can extend the notation collection API in a custom way.

For directly linking presentation markup symbols to their declarations, I have developed an XSLT stylesheet that post-processes the parallel markup generated by JOMDoc and attaches a link to each symbol. The environment that integrates this stylesheet can customize the function that computes the actual link from the *cdbase/cd#name* information of the symbol. The SWiM semantic wiki points any such links to the wiki page that declares the respective symbol (cf. [listing C.1](#)). The current linking implementation has been designed for the Firefox browser<sup>2</sup>. In compliance with

<sup>2</sup>We have tested our publication process with the most recent stable versions of the Firefox browser, 3.6.x at the time of this writing. Where this thesis speaks of Firefox, the statements possibly apply to all browsers that share the Gecko rendering engine.

the MathML 2 specification, Firefox supports simple XLinks on Presentation MathML elements [ABC+03, chapter 7.1.4.1]<sup>3</sup>, which can be activated with the middle mouse button.

### C.1.2.2 Rendering Non-Object Markup

For presenting non-object markup, our implementation reuses existing XSLT stylesheets for OMDoc and OpenMath CDs. When applied to a complete document, the JOMDoc renderer first renders the mathematical objects, and then, in an optional second pass, applies a given XSLT stylesheet. Such a stylesheet has to preserve all objects, as they have already been rendered for publication in the first pass. The SWiM semantic wiki follows the alternative approach of a single-pass XSL transformation with a special template for content markup objects, which, via a Java XPath extension function, invokes JOMDoc's renderer for each object. That way, the rendering process can be customized more flexibly. One such customization is the special way of rendering notation definitions presented below.

The semantics-preserving transformation of OMDoc to XHTML+RDFa<sup>4</sup> could have been implemented completely in XSLT, i.e. one could have hard-coded RDFa support into the existing OMDoc XSLT stylesheets. Particularly RDFa metadata in OMDoc (cf. [chapter 5](#)) could simply have been carried over into the XHTML output by such an implementation. In the interest of a higher modularity and reusability, and of avoiding duplicate implementations of an OMDoc→RDF translation, which other components of an integrated environment also need, my actual solution reuses the RDFa output module of the Krextor library (cf. [section 8.1.1.2](#)). Krextor's OMDoc→RDF translation covers both native OMDoc markup and RDFa annotations embedded into OMDoc. The OMDoc XSLT stylesheets have been extended to call, for every element of OMDoc's semantic markup that they process, the Krextor RDFa output utility module, which generates RDFa annotations for all RDF triples that have the resource represented by the given semantic markup element as their subject. The lower half of [figure 6.9](#) on page 221 shows how the TNTBase database (cf. [section 6.5.2.1](#) and [chapter 8](#)), into which this publishing process has been integrated, generates XHTML+MathML+RDFa documents – by transforming OMDoc to XHTML, including – as RDFa – RDF that has previously been extracted from OMDoc to a triple store. In absence of such an integrated environment that extracts RDF from OMDoc documents before publishing them, the RDFa-extended OMDoc XSLT stylesheets can alternatively execute Krextor's OMDoc→RDF extraction on the fly while rendering one OMDoc document.

The XSLT stylesheets for OMDoc resolve all document inclusions by default. As modularity had not been supported for OpenMath CDs before but was needed for storing statement-level fragments of CDs in the SWiM semantic wiki (cf. [section 9.3.2.1](#)), I have implemented analogous support using XInclude [[MOV06](#)].

### C.1.2.3 Rendering Notation Definitions

My implementation of rendering a preview of a pattern-based notation definition, another extension of the OMDoc and OpenMath CD XSLT stylesheets, replaces all placeholders in the content markup pattern by strings, whose values are the names of the placeholders, and then renders

---

<sup>3</sup>In MathML 3, an *@href* attribute is built into the language [[ABC+10](#), chapter 2.1.6].

<sup>4</sup>RDFa annotations are an optional feature.

Listing C.2: SPARQL query for knowledge items with unresolved issues in the OpenMath wiki

```
# Return all wiki pages P and dates date ...
SELECT ?P ?date WHERE {
  # ... where P has a discussion forum ...
  ?P ikewiki:hasDiscussion ?D .
  # ... that contains an issue ...
  ?C a sioc_arg:Issue;
    sioc:has_container ?D;
  # ... posted on some date ...
  dc:date ?date .
  # ... on which no decision has been made yet ...
  OPTIONAL { ?Dec sioc_arg:decides ?C . }
  FILTER (!bound(?Dec)) .
# ... and order the results by date
# (note: pages with multiple issues are returned once per issue)
} ORDER BY ?date
```

that expression using exactly the current notation definition. For the notation definition from [listing 2.6](#) on page 73, we would obtain  $\text{@(arith1\#divide, } arg1, arg2)$  rendered as  $\frac{arg1}{arg2}$ . [Figure 6.10](#) on page 221 shows the same for an  $n$ -ary operator, where  $arg1, \dots, argn$  are used as arguments.

### C.1.3 Querying Multiple Structural Dimensions with SPARQL

This section explains the implementations of the multi-dimensional queries introduced in [section 6.5.2.3](#).

Listings [C.2](#) and [C.3](#) demonstrate two queries over OpenMath CDs and discussions about them that have been implemented in the SWiM semantic wiki. Each mathematical knowledge item, represented in terms of the OpenMath CD ontology, has a discussion forum associated via an application-specific link. The raw structure of a discussion thread is represented in terms of the SIOC Core ontology; the argumentative structure, represented in terms of the SIOC argumentation module, forms an overlay graph. The first query returns knowledge items of any type that have unresolved issues, considering each knowledge item once per unresolved issue. The second query takes the logical/functional structural dimension into account and only returns certain types of mathematical concepts, each mathematical concept at most once. The two queries have in common that they look up the discussion posts and their dates; these properties are underlined in the listings. The query in [listing C.2](#) restricts the result set along the argumentative dimension, whereas the query in [listing C.3](#) restricts the result set along the logical/functional dimension and additionally consults the raw discussion thread structure to make sure that only the date of the latest discussion post per knowledge item is reported. As these additional constraints are orthogonal, one could combine them in a third query that would return all mathematical concepts of a certain type that have an unresolved issue, ordered by the date of the issue reported most recently.



Listing C.3: SPARQL query for OpenMath concepts (except CDs and symbols) with discussions in the OpenMath wiki

```

# Return all wiki pages P and dates maxDate ...
SELECT ?P ?maxDate WHERE {
  # ... where P has a discussion forum D and is an OpenMath concept ...
  ?P a omo:OpenMathConcept; ikewiki:hasDiscussion ?D .
  # ... but neither a symbol nor a CD
  OPTIONAL {
    ?P a ?T .
    FILTER (?T = omo:SymbolDefinition || ?T = omo:ContentDictionary ) }
  FILTER (!bound(?T)) .
  # In the Discussion forum, find all posts C and their dates ...
  {
    ?C sioc:has_container ?D; dc:date ?maxDate .
    OPTIONAL {
      ?otherC sioc:has_container ?D; dc:date ?otherDate .
      FILTER(?otherDate > ?maxDate) . } .
    # ... but only keep the most recent date (query pattern according to [Fei])
    FILTER (!bound(?otherDate)) }
  # Order the results by date (descending)
} ORDER BY DESC(?maxDate)

```

Listing C.4 demonstrates a that combines several application-specific dimensions of knowledge. A software project manager would execute it<sup>5</sup> when searching a substitute for the employee Alice. First, all documents are retrieved that Alice is responsible for. For any *object* (i.e. knowledge item relevant for the software process) in each of these documents<sup>6</sup> (e.g. the detailed formalization of a feature of the software), the query selects those *relatedObjects* that have been refined by the first *object* in the course of the software engineering process (e.g. a high-level specification of the same feature). Additionally, objects related by mathematical definition, i.e. along the logical/functional dimension, are considered. For any *relatedObject*, we find out what document it belongs to, but we are only interested in recent documents. Finally, the query determines the persons responsible for all such documents and returns their real names from their FOAF profiles. The assumption behind this query is that, if, for example, Bob is responsible for the high-level specification of a feature, which Alice has refined, Bob will be capable of substituting Alice.

### C.1.4 Automated Problem-Solving Assistance

Support for mathematics-specific argumentation and a proof-of-concept assistance with implementing a few common types of solutions (cf. section 6.6.3) has so far been implemented in the SWiM semantic wiki. This has been realized as an extension of the non-argumentative discussion

<sup>5</sup> Access to such queries should, of course, be given via a friendly user interface, for example a context menu displayed wherever information about an employee occurs in the XHTML+RDFa version of a document (cf. section 7.5.3).

<sup>6</sup> *oo:hasPart* is slightly simplified. Locating knowledge items in documents actually requires querying for all parts of a document in terms of the *document* ontology and then keeping those parts that constitute relevant knowledge items w.r.t. the dimension of interest.

Listing C.4: Finding a substitute for an employee in a software project

```

# application-specific dimensions:
PREFIX ver: <http://www.sams-projekt.de/ontologies/VersionManagement#>  # versioning
PREFIX sp:  <http://www.sams-projekt.de/ontologies/V-model#>          # software process
# prefixes for logical/functional structures (OMDoc), administrative metadata (DCMES),
# and user profiles (FOAF) omitted

SELECT ?potentialSubstituteName WHERE {
  # for each document Alice is responsible for, get all of its parts,
  # i.e., transitively, any kind of semantic (sub)object in the document
  ?document ver:responsible <.../employees#Alice> ;
      oo:hasPart      ?object .

  # find other objects that are related to each ?object
  # 1. in that ?object refines them w.r.t. the software process
  { ?object sp:refines ?relatedObject }
UNION
  # 2. or in that they are other mathematical symbols defined in terms
  #   of ?object (only applies if ?object itself is a symbol)
  { ?object oo:occursInDefinitionOf ?relatedObject }

  # find the document that contains the related object and the person
  # responsible for that document ...
  ?otherDocument oo:hasPart      ?relatedObject ;
      dc:date      ?date ;
      sp:responsible ?potentialSubstitute .

  # (only considering documents that are sufficiently up to date)
FILTER (?otherDocument > "2009-01-01"^^xsd:date)

  # ... and the real name of that person
  ?potentialSubstitute foaf:name ?potentialSubstituteName .
}

```

forums that have previously existed in the underlying IkeWiki system [Scho6]. Section 9.3.3.2 provides a complete walk through the implementation, from an issue to assistance with implementing a solution, as further background knowledge about SWiM is required to understand it.

IkeWiki has one discussion forum of type *sio:Forum* per wiki page, which is connected to the knowledge item represented by the page by an RDF link of type *ikewiki:hasDiscussion*. In such a forum, a user can post a new top-level comment, which becomes the root of a thread, or reply to an existing comment. I have extended the user interface by the possibility to post not just untyped comments but arguments of specific types, including domain-specific ones, as shown in figure 6.12 on page 232. An argumentative thread starts with an issue of one of those types that is applicable to the type  $t_k$  of the knowledge item to be discussed. For every possible type of reply to a discussion post, there is a dedicated reply button. Some buttons open menus, from which the type of relationship of the reply can be chosen. For example, for a position, one can choose whether it should agree or disagree with the current post, or be neutral. Similarly, for an argument, one can choose whether it should support or challenge the current post, and what specific type it should have. As with issues, one can select the specific type of an idea from a list of all types that apply to the combination  $(t_k, t_{is})$  of knowledge item type and issue type. Finally, there is still the possibility to create untyped posts, in case the intended contribution does not fit into the schema of the argumentation ontology.

The formulæ for identifying unsolved legitimate issues and identifying the best proposed solutions cannot be implemented as single closed queries in standard SPARQL, as they involve filtering issues and ideas by the counts of agreements and disagreements; therefore, it has been implemented as an algorithmic sequence of SPARQL queries against the RDF graph of the discussion about the knowledge item currently viewed.<sup>7</sup>

## C.2 JOBAD, a Library of Assistive Services for Interactive Documents

Our initial implementation of the JOBAD architecture introduced in chapter 7 focuses on XHTML+MathML documents displayed in browsers that support interactivity via JavaScript; analogous ideas should, however, also be realizable in other environments for interactive applications, such as Adobe Flash with ActionScript [Ado]. Parts of this implementation have been realized on top of the TNTBase (cf. section 6.5.2.1) and MMT (cf. section 2.4.4.1 and [Raba]) backends.

Listing C.5 demonstrates how JOBAD services are initialized in a document generated by the server backend, as explained in section 7.3.1.

### C.2.1 A Generic Proxy for Accessing Remote Web Services

In the TNTBase and MMT systems, which currently implement parts of the JOBAD architecture, the primary backend provides most of the functionality used by the JOBAD services. Our JavaScript realization of JOBAD is effectively limited to connecting to its primary backend due to the “same origin policy” [Zal10, part 2], which is effective in all contemporary browsers. It states that a

<sup>7</sup>With the sub-query and aggregate extensions (*GROUP BY*, *count*, and *HAVING*) of the ARQ query processor [Arq], which is used here, this could be realized completely in SPARQL.

Listing C.5: Service initialization code in a document

```

<!-- utility functions (module loading, document manipulation,
      client/server communication) -->
<script src="../../scripts/jobad.js"/>
<!-- our own initialization follows -->
<script type="text/javascript">
// GUI elements to be enabled
jobadInit("ui/contextmenu"); // loads the context menu
// In-document services
jobadInit("service/elision");
// Web services
jobadInit("service/definition-lookup", "Look up definition",
  "http://jobad.mathweb.org/backend?action=definition-lookup
    &cdbase=$cdbase&cd=$cd&name=$name");
</script>

```

browser script may only request resources from the same domain, protocol, and port from which it has originally been served. The HTTP proxy functionality in the primary backend, which works around this limitation, has been realized by offering a RESTful interface that accepts, as a parameter, an escaped representation of the remote URL. Suppose the proxy is running at <http://jobad.mathweb.org/proxy>, and a JOBAD service wants to send an HTTP GET request to a remote URL <http://example.org/service?param=value>. In that case, the JOBAD service would send a GET request to <http://jobad.mathweb.org/proxy?url=http%3A%2F%2Fexample.org%2Fservice%3Fparam%3Dvalue>, which the proxy would forward accordingly.

## C.2.2 In-Document Services

### C.2.2.1 Folding Subterms and Undoing Interactions

The implementation of the folding service described in [section 7.4.1](#) relies on the *maction* element (cf. [section 6.4.2.3](#)), whose *@selection* attribute indicates the child element currently displayed. Caches are realized by creating *maction* elements on the fly, which hold cached content in a child element currently invisible. Caches for different JOBAD services are distinguished by different values of the *maction/@actiontype* attribute.

Any subterm that is grouped according to [requirement 2](#) in [section 6.4.2.5](#) is eligible for folding on demand. When the user requests folding of a subexpression for the first time, we put both the original subterm and its folded version into a dynamically generated *maction* element with *actiontype folding* for making the folding action undoable.

### C.2.2.2 Flexible Elision and Display of Reading Aids

Listings [C.6](#) and [C.7](#) show examples of how elidable brackets and type annotations (cf. [section 7.4.2](#)) are represented in MathML. Invisibility is realized by selecting an (empty) *mspace* child element of an *maction*. The JOMDoc renderer described in [appendix C.1.2](#) generates such markup for

Listing C.6: An elidable bracket in Presentation MathML

```

<maction actiontype="elision" selection="1" o:ellevel="100">
  <mspace/>
  <mo>(</mo>
</maction>

```

Listing C.7: An elidable type annotation in Presentation MathML

```

<msub>
  <mi>F</mi>
  <maction actiontype="elision" selection="1" o:egroup="type" o:ellevel="300">
    <mspace/>
    <mrow> $\iota \rightarrow o$ </mrow>
  </maction>
</msub>

```

brackets. Type annotation markup has so far only been realized in the context of the Logic Atlas [KMR], where OMDoc documents are generated from Twelf sources [Pfe01]. The Twelf system is capable of reconstructing types and annotates them with special attributions in the exported OMDoc, which renderer of the MMT backend [Raba] preserves, and which the variant of the JOBAD elision service employed in the Logic Atlas uses to show or hide the reconstructed types.

## C.2.3 Symbol-based Services

### C.2.3.1 Definition and Type Declaration Lookup

So far, definition and type lookup (cf. [section 7.5.1](#)) have been implemented for the TNTBase (OMDoc 1.2/1.3) and MMT (OMDoc 1.6) backends via the RESTful URL format shown in [listing C.5](#) [ZK09; DKL+10a]. Here, the backend offers a whole range of services, which are distinguished by action verbs. In the following, I explain how definition lookup has been implemented for OMDoc 1.2/1.3 documents in the TNTBase backend. Here, [a rendering of] the whole *definition* element  $\text{def}(\sigma)$ , i.e. with informal and formal properties, is returned. [Listing C.8](#) shows the XQuery implementation employed by TNTBase; it assumes that the URI of the symbol has been split into the *cdbase*, *cd*, and *name* components, where *cdbase* is ignored in the current implementation – assuming that there are no two different theories of the same name in the knowledge base. The complete collection of OMDoc documents is searched for the theory in question, from which the definition of the symbol in question is looked up. The query takes into account the possibility that an OMDoc theory does not contain symbol declarations and definitions as direct children but rather as children of intermediate grouping constructs (*omgroup*). It also takes into account that one definition can define more than one symbol; consider a mutually recursive definition of two symbols, as discussed for the example of odd/even in [section 3.2.3.5](#). The performance of this query can be speeded up by creating appropriate indices, as explained in [section 6.5.2.1](#). Once more, this will be easier to realize for simple definitions in strict OMDoc 1.6. Actually, the

## Listing C.8: XQuery code for definition lookup

```
(: look up the theory :)  
for $theory in tnt:collection()//theory[@xml:id = $cd]  
(: look up all definitions in that theory (but not in nested sub-theories) :)  
let $definition = $theory//definition[ancestor::theory[1] = $theory]  
(: return the definition of the given symbol :)  
return $definition[tokenize(@for, '\s+') = $name]
```

functionality of this query is largely redundant with the OMDoc→RDF translation that a versatile knowledge base will have to perform in any case in order to enable a larger number of services, as discussed in [section 6.7](#).<sup>8</sup> If an RDF graph has already been extracted from the OMDoc documents, the query for looking up a definition simplifies to the SPARQL code **SELECT** ?definition **WHERE** { ?definition oo:defines <URI-of-symbol> . }. Note that neither query takes into account that, in an invalid document collection, the symbol itself might not have been declared, and that multiple *definition* elements might claim to define the same symbol. Thus, we assume that the document collection has been validated before running definition lookup.

### C.2.3.2 Linked Data Navigation

The linked data navigation service for non-formulae (cf. [section 7.5.3](#)) leverages RDFa annotations and has been implemented using the rdfQuery JavaScript library [Ten+].

### C.2.3.3 Visualizing Rhetorical Structures

The implementation of the rhetorical structure visualization described in [section 7.5.4](#) predates the proper JOBAD implementation and the addition of RDFa output to the JOMDoc renderer. In the XHTML output generated from a rhetorically annotated OMDoc source, the rhetorical structures are preserved using a non-standard mixture of RDFa and microformat-style annotations. Such an XHTML document is then post-processed by an XSLT stylesheet, which adds the interaction widgets to the nuclei. A port to the current JOBAD implementation would, however, be straightforward.

## C.2.4 Expression-based Services

### C.2.4.1 Rendering as a Service

The implementation of the rendering service (cf. [section 7.6.1](#)) in the TNTBase backend relies on the JOMDoc library (cf. [appendix C.1.2](#)), whereas its implementation in the MMT system relies on the MMT library (cf. [section 2.4.4.1](#) and [Raba]).

---

<sup>8</sup>Given that we have implemented the OMDoc→RDF translation in XSLT, as explained in [section 8.1](#), even the source code of the query is largely redundant, as XSLT and XQuery share a common XPath foundation.

#### C.2.4.2 Looking up General Computational Information with Wolfram Alpha

As Wolfram Alpha neither understands OpenMath nor Content MathML, the JOBAD Wolfram Alpha service introduced in [appendix C.2.4.2](#) has to translate content markup selected in a document to the Mathematica syntax that Wolfram Alpha natively understands. This translation is performed by calling the OpenMath to Mathematica translation web service offered by the MathDox project [[Matc](#)]. In its current state, the JOBAD Wolfram Alpha service displays the presentation markup returned by XHTML and Presentation MathML response received from Wolfram Alpha in a popup window. In-place rewriting the selected mathematical expression, e.g. replacing it by its derivative if desired, is not currently possible in a JOBAD-compliant way. In order to satisfy [requirement 3](#), one would have to obtain content markup from Wolfram Alpha and let the JOBAD rendering service render it into parallel markup. Wolfram Alpha can be asked for a Mathematica response, but a Mathematica→OpenMath translation has not yet been integrated into the JOBAD Wolfram Alpha service.<sup>9</sup>

#### C.2.5 Unit Conversion

The web service on which our unit conversion service introduced in [section 7.6.3](#) relies does not currently talk OpenMath, despite using it internally, but uses string input/output. Therefore, our client-side implementation translates quantities between their OpenMath and string representation (e.g. “1.5 metre”). So far, we have neither realized this for compound units (e.g.  $\frac{m}{s}$ ) nor for prefixed units (e.g.  $km$ )<sup>10</sup>, but instead rely on an OpenMath interface to become available eventually.

### C.3 Transparent Translations in Knowledge Bases

#### C.3.1 The Krextor XML→RDF Translation Framework

Traditionally, XML→RDF translations have often been implemented in XSLT (cf. [section 8.1.4.1](#)) – so is Krextor (cf. [section 8.1](#) and [[Lan+](#)]). However, it aims at enabling developers to realize more functionality – i.e. supporting multiple XML input languages and multiple RDF output serializations – with less coding effort than using pure XSLT. I chose XSLT for the following reasons<sup>11</sup>: (i) It automatically traverses the input document in a depth-first recursion, which is convenient for the (almost) complete translation of XML documents to RDF required here; exceptions from that rule are also supported. (ii) Parameters can be looped through multiple tree recursion levels without explicit passing (“tunnel parameters”); this technique is employed for the  $b$  and  $p$  parameters shown in algorithms [8.1](#) to [8.5](#). (iii) It supports full XPath access to any node of the input document and documents at other URIs at any time in the transformation process. (iv) An XSLT implementation is extensible by overriding templates; additionally, the

---

<sup>9</sup>The translation that the MathDox project offers as a web service only works with Mathematica expressions in full form (without any “pragmatic” syntax), which Wolfram Alpha does not currently return. We have investigated other possibilities, none of would be ready to reuse without major adaptations [[DLRio](#)].

<sup>10</sup>This explains the unrealistic example in [figure 7.6](#).

<sup>11</sup>The “development notes” page of the Krextor homepage [[Lan+](#)] provides a detailed comparison of XSLT with potential alternative languages for implementing Krextor; [section 8.1.4](#) discusses further alternatives.



FXSL extension library enables higher-order functional programming<sup>12</sup> (albeit in a clumsy syntax) [Nov]. Krextor is available as a collection of XSLT stylesheets, with an optional Java wrapper for direct integration into applications. For scripting and debugging, there is a command-line frontend, which reads XML from the standard input or a given file and writes RDF in the desired serialization to the standard output; for example, the command `krextor omdoc-owl..turtle doc.omdoc` would translate the file *doc.omdoc*, treating it as an OWL ontology implemented in OMDoc, to RDF in Turtle serialization.

### C.3.1.1 Built-in Functions for Generating Fragment Identifiers

Krextor provides the following functions for generating fragment identifiers for the URIs that it mints (cf. section 8.1.1.1): *xml-id* uses the `@xml:id` attribute of an XML element (if present), *generate-id* calls the *generate-id* function built into XSLT, which generates a unique identifier for the given XML node<sup>13</sup>, and *pseudo-xpath*, which generates, e.g., a string like *doc-section2-para1* from a node whose XPath is `/doc/section[2]/para[1]`.

### C.3.1.2 Compliance of the Built-in RDFa Extraction Utility Module

For the purpose of testing Krextor's RDFa extraction utility module (cf. section 8.1.2) routines against the RDFa test suite [HY07], which was limited to the XHTML host language at the time of testing (January 2009), I have also developed an extraction module for XHTML+RDFa, which adds the XHTML-specific RDFa processing rules. It passes 90 out of the 100 test cases; however, those where it fails, do not affect the recommended syntax for embedding RDFa into OMDoc specified in section 5.2.2.

### C.3.1.3 Built-in Output Modules

This section explains technical details for some of Krextor's output modules (cf. section 8.1.1.2).

The Java callback interface is due to the Saxon XSLT processor [Kay08]. It allows for registering a function that is called once per triple extracted and thus eliminates the need for a Java application to parse Krextor's output once more. The RDF/XML and Turtle outputs group triples by common subjects, and predicates in the case of Turtle, by first obtaining RXR and then transforming it to the serialization desired using XSLT grouping – a compromise between efficiency and a clean separation of concerns. RXR has been chosen as Krextor's central output serialization, as its uniform XML structure (every triple represented by an *rxr:triple* element with children *rxr:subject*, *rxr:predicate*, and *rxr:object*) makes it easy to parse.<sup>14</sup> One shortcoming of RXR is its lack of support for anonymous blank nodes, which are therefore not currently supported by Krextor.

<sup>12</sup>XSLT 3 will natively support that [Kay10].

<sup>13</sup>*generate-id* is guaranteed to “always [return] the same identifier for the same node and [...] different identifiers [...] from different nodes” [Kay07]; however, “there is no guarantee that a generated unique identifier will be distinct from any unique IDs specified in the source document” [Kay07]. The latter may lead to clashes when a Krextor extraction module uses both *xml-id* and *generate-id* to generate URIs.

<sup>14</sup>Besides that, the author of RXR points out 12 more advantages over RDF/XML, including (i) the ability to validate RXR against an XML schema, as names of RDF resources – whose vocabulary is unrestricted – only occur in attribute values, not in element names, (ii) and the absence of excessive alternative ways of serializing the same RDF graph [Beco4a]. (IAN DAVIS counted 16 ways of writing down three RDF triples [Dav05].)

Implementing the publication process for a language in XSLT makes RDFa-annotated output easiest to obtain. Such an XSLT stylesheet has to apply Krextor's templates in RDFa output mode to every semantically relevant element of the input document while rendering it. The RDFa output utility module currently expects the triple store that it queries to serve RXR. That works in *our* implementation on top of TNTBase, but for future integration with different systems I plan to support the SPARQL Query Results XML format, which SPARQL endpoints return [BB08]. [Appendix C.1.2.2](#) explains how Krextor's RDFa output facility has been integrated into XSLT stylesheets that generate XHTML+RDFa from OMDoc.

The syntactic sugar that some RDF serializations offer for improving human readability and for saving space has only partly been implemented. Neither author-defined namespace prefixes, nor “anonymous” blank nodes, nor RDF containers or collections are supported at the moment. In RDFa output mode, Krextor groups annotations by common subject; the output element representing the subject gets an *@about* attribute holding the subject URI and as many child elements (in XHTML: *span*), each with a predicate (*@rel* or *@property*) and object (*@resource* or *@content*) as there are triples. Further possible RDFa space optimizations, as discussed in [section 6.4.2.7](#), are not currently performed. Fully inlining RDFa annotations into the markup of the published document is not feasible with this generic annotation approach that is largely decoupled from the publication process. As argued in [section 8.1.1.2](#), the lack of syntactic sugar does not make a difference from a semantic point of view. Nevertheless, supporting some of it remains on the agenda, as it facilitates testing when developing extraction modules.

#### C.3.1.4 Implementing Extraction and Output Modules

**Extraction Modules:** In the most general case, the XSLT templates of a Krextor extraction module map patterns of an input XML document to partial RDF subgraphs. Krextor's generic module offers convenience templates and functions for many common XML→RDF translation tasks, which reduces the amount of XSLT code required, compared to a hand-crafted extraction of, e.g., RDF/XML from semantic markup. The defaulting behavior of the predefined templates for creating a resource that is instance of a given class (partly rendered in [algorithm 8.3](#)) and for adding literal- or URI-valued properties to the current resource (algorithms [8.4](#) and [8.5](#)) requires the developer of an extraction module to provide little information explicitly. In addition to the functionality described generally in [section 8.1.1.1](#), my XSLT implementation supports deviating from a depth-first recursion (which is merely the default processing order), overriding the base URI of an input XML document, using properties in inverse direction, sequences of property or object parameters that lead to the creation of multiple triples at once, chaining resources into RDF collections, and normalizing whitespace in literal values. As an alternative for simple languages, where XML elements or attributes directly map to ontology classes or properties, a declarative mapping can be given as annotated literal XML. For complex input formats, such as OMDoc, the Turing-complete computational power of XSLT (cf. [[Kep04](#)]) is available, at the expense of readability.

[Listing C.9](#) demonstrates part of the OpenMath CD extraction module. The declarative syntax, which maps element names to classes or properties, works for large parts of the syntax. The *krextor:resources* list maps, e.g., *CDDefinition* to *omo:SymbolDefinition* and creates a link of type *omo:definesSymbol* from the resource represented by the parent element (i.e. the CD). The template be-

Listing C.9: Excerpt from Krextor's OpenMath CD extraction module

```

<!DOCTYPE xsl:stylesheet [
  <!-- ENTITY omo "http://www.openmath.org/ontology#" -->
  <!-- ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" -->
  <!-- ENTITY dct "http://purl.org/dc/terms/" -->
]>
<xsl:stylesheet version="2.0"
  xpath-default-namespace="http://www.openmath.org/OpenMathCD"
  xmlns:krextor="http://kwarc.info/projects/krextor"
  xmlns="http://www.openmath.org/OpenMathCD">
  <xsl:variable name="krextor:resources">
    <CD type="&omo;ContentDictionary"/>
    <CDDefinition type="&omo;SymbolDefinition"
      related-via-properties="&omo;definesSymbol"/>
    <!-- ... -->
  </xsl:variable>
  <xsl:template match="CD|CDDefinition|..." mode="krextor:main">
    <xsl:apply-templates select="." mode="krextor:create-resource"/>
  </xsl:template>

  <xsl:variable name="krextor:literal-properties">
    <Name property="&dct;identifier" normalize-space="true"/>
    <Description property="&dct;description" normalize-space="true"/>
    <!-- ... -->
  </xsl:variable>
  <xsl:template match="Name|Description|..." mode="krextor:main">
    <xsl:apply-templates select="." mode="krextor:add-literal-property"/>
  </xsl:template>

  <xsl:template match="om:OMOBJ//om:OMS" mode="krextor:main">
    <xsl:call-template name="krextor:add-uri-property">
      <xsl:with-param name="property" select="'&omo;usesSymbol'"/>
      <xsl:with-param name="object"
        select="om:symbol-uri((ancestor-or-self::om:*/@cibase)[last()], @cd, @name)"/>
    </xsl:call-template>
  </xsl:template>

  <!-- ... -->
</xsl:stylesheet>

```

low the variable declaration instructs Krextor to create resources according to these declarations.<sup>15</sup> Analogously, the *krextor:literal-properties* list maps, e.g., the text content of *Name* elements to *dct:identifier* properties, ignoring whitespace surrounding the values. The flat representation of OpenMath objects and the symbols that occur in them has to be created by a regular Krextor template. It matches any *OMS* element in an object, creates an OpenMath-compliant URI for the symbol, and links it to the parent resource (a *CommentedPart* or *FormalPart* of a *Property*, or an *Example*) by the *omo:usesSymbol* property.

**Output Modules:** A new output module merely has to implement one template for low-level RDF generation, accepting the parameters subject (URI or blank node ID), subject type (URI or blank node), predicate (URI), object, object type (URI, blank node ID, or literal), language, and datatype. More ambitious output modules can be realized by post-processing output from existing output modules.

**Composing a Complete Translation:** The complete translation from an input format to an output format is performed by an XSLT stylesheet that includes the respective extraction and output modules, as shown in [listing C.10](#). The Java wrapper and the command-line frontend can generate such stylesheets on the fly.

Listing C.10: A complete translation, composed of an extraction and an output module

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
  <xsl:import href="output/java.xsl"/>    <!-- Java output module -->
  <xsl:include href="extract/omdoc.xsl"/> <!-- OMDoc extraction module -->
</xsl:stylesheet>
```

### C.3.2 A Translator of OWL Ontologies from RDF to OMDoc

An initial translation of OWL 1 ontologies from an RDF implementation to an OMDoc implementation (cf. [section 8.2.2](#)) has been realized as a Java command-line application, using the Jena RDF API for Java [[Jen](#)] for parsing the RDF input<sup>16</sup> and the Velocity templating engine [[Apab](#)] for creating the OMDoc output [[Kuro9](#)]. Its key features are:

**Customizable Theory Names:** The name of the resulting OMDoc theory can be auto-generated from the ontology's namespace URI or defined by the user. This holds analogously for reused external ontologies.

**Auto-generation of Theory Imports:** For each external ontology reused, the translation adds a theory import to the primary OMDoc theory.

**OMDoc Syntactic Sugar:** The OMDoc syntactic sugar for avoiding subject-predicate-object axioms for resource and property type declarations (with range and domain) and class definitions, as specified in [section 4.3.2](#), is supported.

---

<sup>15</sup>In *krextor:create-resource* mode, Krextor locates the mapping for the element currently processed in the *krextorresource* list. Implementing one Krextor template per element involves more lines of XSLT code, which an XSLT processor, however, can optimize more easily.

<sup>16</sup>currently limited to RDF/XML, but an extension to Turtle or N-Triples would be trivial to realize with the Jena API.

**Selectable Default Namespace:** An RDF input file may describe entities from multiple namespaces, but the output is always one OMDoc theory corresponding to one of these namespaces. Entities from that primary namespace are represented as OMDoc symbols with the above-mentioned syntactic sugar, whereas subject-predicate-object axioms – in the same theory, for now – are generated from the descriptions of entities from other namespaces. The user can indicate the primary namespace; otherwise, the translator picks the namespace with the largest number of resources described.

### C.3.3 Translations Between Different Representation Granularities

#### C.3.3.1 An Importer/Exporter for Filesystem Documents into a Knowledge Base

I have implemented a generic version of both directions of the file import/export algorithm specified in [section 8.3.2.1](#). In the context of the SWiM semantic wiki (cf. [chapter 9](#)) and its application to the OpenMath CD maintenance workflows introduced in [section 6.1.2](#), this implementation has been specialized to OpenMath CDs, signature dictionaries, and notation dictionaries. The JOMDoc library [[Jom](#)] independently provides a partial implementation of the same algorithm for OMDoc.

The generic implementation is realized by two XSLT 2 stylesheets. The one responsible for splitting imported documents uses the *xsl:result-document* facility for creating multiple result documents, one per fragment split off. A language-specific XSLT stylesheet implements the *s* predicate mentioned in [section 8.3.2.1](#), which identifies whether a fragment is eligible for splitting, by copying its input but matching all elements for which *s* is defined true by a template that switches processing to *split* mode. In OpenMath CDs, merely CDs symbol definitions, and symbol type signatures have fragment identifiers, represented by the *CDName* and *Name* child elements and *@name* attributes, respectively. For fragments that should be split but do not have an identifier, the importer generates a pseudo-XPath identifier – for example *FMP2*, if *v* is the second *FMP* child element of its parent – and appends it to the identifier of the (sub)document currently processed to obtain a complete URI, as, for example *http://www.openmath.org/cd/arith1.ocd+plus+FMP2*. The OpenMath instance of the *e* predicate for determining whether a given document *D* from the knowledge base is admissible for export, and, if not, the lookup of the nearest exportable parent document, has been implemented in Java and SPARQL, drawing on the RDF extracted from the CDs by Krenator (cf. [section 8.1.2](#)). If *D* is an instance of *omo:Dictionary*, it can be exported; if not, the *omo:hasPart* links are traversed in reverse direction until a qualifying parent has been found.

The JOMDoc library [[Jom](#)] provides a partial OMDoc-specific implementation of the same algorithm as a part of its general reference introduction and contraction facility. It is OMDoc-specific in that it assumes that the OMDoc *ref* element is used for inclusion, but, in addition to the XSLT implementation mentioned above, it allows the end user of the JOMDoc command-line frontend to specify the *s* predicate by passing a list of XPath node test expressions. In the current implementation, the fragments, for which references have been introduced, are not split off into individual documents; instead, all of them are either written into one new document, or into a hidden section of the original document.

### C.3.3.2 A Process that Makes Metadata Accessible for Different Services and Editors

This section describes the implementation of how the SWiM semantic wiki makes metadata in OpenMath CDs available both to RDF- and document-oriented services (cf. [section 8.3.3](#)).

On any OpenMath CD XML document that has been saved in the editor or imported into the knowledge base, Krextor is run and extracts an RDF outline, including metadata (cf. [section 8.1.2](#)). After that, an XSLT is applied to the document that removes from any element carrying metadata the metadata value and adds the CURIE of the corresponding RDF property as an application-specific attribute. For example, `<CDDate>2010-30-09</CDDate>`, which yields the RDF triple `<URI-of-CD> dc:date "2010-30-09"^^xsd:date`, is replaced by `<CDDate swim:meta="dc:date"/>` in the XML document in the knowledge base. On publishing or exporting such a document, these metadata are retrieved from the RDF triple store. When preparing a document for editing in the HTML document editor presented in [section 6.2.6](#), all elements with `@swim:meta` attributes are transformed into editable text spans that contain the current value of the metadatum, as retrieved from the RDF triple store. On saving that document, the metadata fields from the document editor, which have possibly been changed, are first translated back from the HTML format specific to the editor to the original semantic markup (e.g. `<CDDate>2010-10-01</CDDate>`), to which RDF extraction<sup>17</sup> and the above-mentioned rewriting with `@swim:meta` are applied, thus updating the value of the metadata field in the RDF triple store.

This process is capable of making new metadata fields added in the document editor available in the form interface, whereas metadata added in the RDF-based form interface do not appear in the document editor on the next edit, as no placeholder element pointing to them has been added to the XML document. In the SWiM implementation, such metadata fields would only become available after one export/import run, which generates self-contained semantic markup. When SWiM runs on top of a Subversion repository, as explained in [section 9.3.2.2](#), it exports documents to the repository after every metadata edit.

## C.4 The Semantic Wiki SWiM

This section provides further technical details about the architecture and implementation of SWiM (cf. [section 9.3](#)).

### C.4.1 IkeWiki's Ontology and Reasoning Support

The administrator who installs IkeWiki can choose the ontologies to be preloaded into the database on a screen of the installation assistant. Preloading ontologies right on setup is the easiest way, but one can also import additional ontologies later on. The triple store is accessible via a Jena API [[Jen](#)] and SPARQL queries processed by ARQ [[Arq](#)]. Beyond RDFS reasoning, there is the experimental possibility to enable OWL reasoning using Pellet [[SPCG+06](#)]. In a setup with the expressive OMDoc and OpenMath CD ontologies and dozens of documents or CDs containing thousands of instances of these ontologies, however, the latter has been found to extremely slow down the performance of all actions depending on the RDF triple store.

---

<sup>17</sup>after deleting all triples extracted in the previous run

## C.4.2 Storage Backend

### C.4.2.1 Document Translation and Storage

Whenever a wiki page is saved inside SWiM, or whenever an external document is imported into SWiM (from a file, or from a connected Subversion repository), the following translations are performed, for reasons outlined in [section 9.3.2.1](#):

1. The document is split into fragments of the desired granularity according to [section 8.3.2](#), each of which is stored as one wiki page.
2. Krexitor (cf. [section 8.1](#)), using the OMDoc or OpenMath CD extraction module, extracts RDF from the XML representation of any mathematical knowledge item that results from the previous step. Krexitor directly feeds its output into the RDF triple store, using the Java output module.
3. After markup elements containing metadata have been extracted to the RDF triple store, a placeholder is put into their original position, so that the document editor and the publication process still have access to them, and that they remain in the right place in exported files (cf. [section 8.3.3](#)).

On exporting a wiki page from SWiM to the file system or a Subversion repository, steps 1 and 3 are applied to the nearest exportable parent page (in terms of [section 8.3.2](#)), in order to obtain a self-contained file. Note that the new values of the *dc:date* (in OpenMath CDs: *CDDate*) and *dc:creator* (not present in OpenMath CDs) metadata fields reflect the last change made to the respective page in SWiM.

### C.4.2.2 A Client for Subversion Repositories

SWiM's Subversion client, introduced in [section 9.3.2.2](#), has been realized as an extension of the file import/export facility, as that entailed the least impact on the IkeWiki code base.

Currently, SWiM implements the bare minimum of Subversion commands that are required for connecting to a repository: *update* (performed automatically on every access to a wiki page)<sup>18</sup>, *commit*, *lock*, and *unlock*. While a wiki page of a working copy is opened for editing, SWiM locks it in the repository. This is contrary to Subversion's approach of "optimistic locking", also called "copy-modify-merge" [[PCSFo8](#), chapter 1], where users may simultaneously edit files but the user trying to commit a file that has already been changed by somebody else has to resolve any resulting conflicts first.<sup>19</sup> File locking does not require a specific user interface, whereas conflict resolution does.<sup>20</sup> All Subversion commands resulting from an access to a page  $P$  in SWiM's working copy are applied to the nearest exportable parent page  $D_p(P)$  according to [section 8.3.2](#), but a reference to  $P$  is recorded in the commit log message.

<sup>18</sup>Unless a special post-commit hook [[PCSFo8](#), chapter 5] is set up on the repository, SWiM does not notice revisions committed to the repository from other clients. Therefore, the revision history of a wiki page in SWiM's Subversion working copy is only a subset of the full revision history.

<sup>19</sup>In well-structured documents, such conflicts occur rarely. When two users edit different sections of a file, Subversion can merge the two changes. Conflicts only occur when two users commit changes to the same line.

<sup>20</sup>Many wikis, such as MediaWiki, feature conflict resolution user interfaces, but IkeWiki does not.



Subversion repository access is configured per namespace.<sup>21</sup> In the easiest case, a qualified name *nsprefix:localname* of a wiki page is mapped to the Subversion resource *concat(nsuri, localname)*, *nsprefix* being the prefix and *nsuri* being the URI of the namespace. More complex mappings are possible (cf. [table 9.1](#) on page 293). SWiM and Subversion do not currently use unified user accounts. For each combination from Users  $\times$  Namespaces inside SWiM, a Subversion username and password can be configured. One can set up an 1:1 mapping of SWiM to Subversion users, but it is less effort to only maintain one Subversion account for each group of SWiM users who should have the same permissions in the repository. SWiM enables identification of the wiki user who initiated a commit by including its name in every log message (cf. [listing 9.1](#) on page 293).

### C.4.2.3 Efficiently Publishing Formulæ when Notation Definitions are Changeable

This section adds specific technical remarks to the description given in [section 9.3.2.4](#).

Rendering mathematical knowledge items is expensive in SWiM not only due to the inherent complexity of rendering semantic mathematical markup, but also for two technical reasons: (i) The above-mentioned splitting large units of knowledge into small fragments explained in [appendix C.4.2.1](#) requires looking up a large number of XIncluded knowledge items from the database when completely displaying a large unit, such as a whole CD, to the user.<sup>22</sup> (ii) The incompatibility of the Dojo user interface toolkit [[The](#)] employed by IkeWiki with XML enforces post-processing of all Presentation MathML objects.<sup>23</sup> This is worth mentioning, for it is not merely a bug that only occurs in the specific setting of IkeWiki. Ignorance of XML by major browsers and web development libraries is still, as in the early ages of MKM on the Web (cf. [section 1.4.3.3](#)), a major obstacle to the adoption of MathML.

Caching of rendered documents, which the description in [section 9.3.2.4](#) assumes, is not currently implemented in IkeWiki/SWiM. Thus, every page is re-rendered when a user visits it. However, caching would be easy to realize.

The query for knowledge items affected by a change to a notation definition is currently completely implemented in SPARQL, as the RDF triple store employed by IkeWiki neither supports the OWL 2 property chain axioms needed for computing the *presentationDependsOn* property introduced in [section 3.2.2.6](#), nor the transitivity of *hasPart*.

## C.4.3 User Interface

### C.4.3.1 Giving Local Access to the Editor

The links for opening subparts of larger knowledge items as wiki pages of their own, in order to enable local editing (cf. [section 9.3.3.1](#)), are created by an extension of the XSLT stylesheet

<sup>21</sup>This has to be configured manually in the database [[Lano8b](#)].

<sup>22</sup>Employing a database with fine-grained fragment access, as discussed in [section 8.3.4.1](#), would not solve that problem but merely shift it: Then, rendering a large unit would only require one database lookup, whereas rendering a fragment of a large unit would first require extracting it.

<sup>23</sup>Dojo's widgets are configured using non-standard HTML attributes. These make the Firefox browser – the only browser that supports MathML to an extent sufficient for conducting the research presented here – fall back from the *application/xhtml+xml* content type to the *text/html* “tag soup” content type, in which XML namespaces are not recognized. Therefore, SWiM post-processes any Presentation MathML into JavaScript code that inserts the same Presentation MathML into the document via the DOM when the document is viewed in the browser.

used for publishing. It has been extended not to not only expand all included fragments when rendering a large knowledge item (cf. [section 6.4.2.8](#)), but to attach to each such fragment a link to its representation as a wiki page of its own.<sup>24</sup> As the document editor presented in [section 6.2.3](#) does not currently resolve links to included fragments, authors are actually forced to open the most local knowledge item for editing. This restriction aside, local access to the editor is a true feature in many web collaboration environments, as discussed in [section 9.5.4](#).

---

<sup>24</sup>Due to a technical restriction of IkeWiki's user interface, this link does not actually point to the editing view but to the rendered view of that page.















## Survey Results

### **D.1 Reporting and Solving Issues with Mathematical Knowledge Items**

52 people participated in this survey, whose results are summarized in [section 3.6.2.1](#).

#### **What is your previous experience with mathematical knowledge management?**

52 participants answered this question. Multiple answers were possible.

Operated/hosted/maintained a knowledge base	15	
Developed a knowledge base or a system for managing it	17	
Participated in a website (e.g. Wikipedia, Planet-Math)	25	
Contributed to a library of a software tool (e.g. of an automated theorem prover or a computer algebra system)	30	
Participated in a knowledge base open for any contribution (e.g. Wikipedia)	22	
Participated in a knowledge base writable by a group of editors/developers, where users could comment	11	
Participated in a knowledge base only writable by editors/developers, read-only for others	17	
Participated in a completely closed, non-public knowledge base	5	
Collected general-purpose knowledge (as e.g. in Wikipedia)	15	
Edited knowledge for e-learning (as e.g. in ActiveMath)	15	
Edited knowledge for scientific publishing	22	
Other	6	



















“Other” responses:

- “teaching”
- “used mathematical assistance systems for formalizing mathematical statements and proofs”
- “developed an ontology of instructional objects, used, e.g., in ActiveMath ”
- “WWW usage”
- “build and coordinate research for a Mathematica Grid network”
- “developed an ontology”

### What knowledge management and issue reporting/resolving features did the mathematical knowledge base(s) support that you participated in?

33 participants answered this question. Multiple answers were possible. The following explanation of the term “knowledge item” was provided:

Note that by “knowledge items” we mean the smallest unit of resources that knowledge management tasks can be done on. This definition includes knowledge items that consist of subitems, e.g. a theory consisting of definitions and axioms. With plain, unstructured text, a knowledge item would be a file. With structured documents, this unit can be a smaller fragment of a file or a web resource.

Knowledge items of the size of a course unit or lecture	21	
Knowledge items of the size of one presentation slides	13	
Knowledge items of the size of one mathematical theory (i.e. a few related definitions and axioms)	19	
Knowledge items of the size of one mathematical statement (e.g. one definition, one axiom, one assertion, one proof, one example)	18	
built-in facility to submit structured issue reports where you could select from pre-defined types of issues	7	
built-in facility to submit unstructured issue reports or general comments	5	
Mailing list, forum, newsgroup or any other place to report issues outside of the knowledge base	18	
Contacting authors/editors personally (e.g. by e-mail)	19	
Guidance given on how to report issues	8	
Ability for issue reports to refer to exactly one knowledge item	8	
Ability for issue reports to refer to multiple knowledge items	7	
Issues tracked on a general level (e.g. in one mailing list for the whole knowledge base)	13	
Support for manual editing and restructuring of knowledge items in a rather formal language (looking like a programming language)	17	
Manual editing and restructuring of knowledge items in a representation that looked like mathematics written on paper (e.g. WYSIWYG editing of formulæ)	10	
Assistance with restructuring (like refactoring support in software development environments)	5	
Semi-automated solutions offered for common issues, e.g. by software assistance	6	
Automated solutions offered for common issues	8	
Other	2	

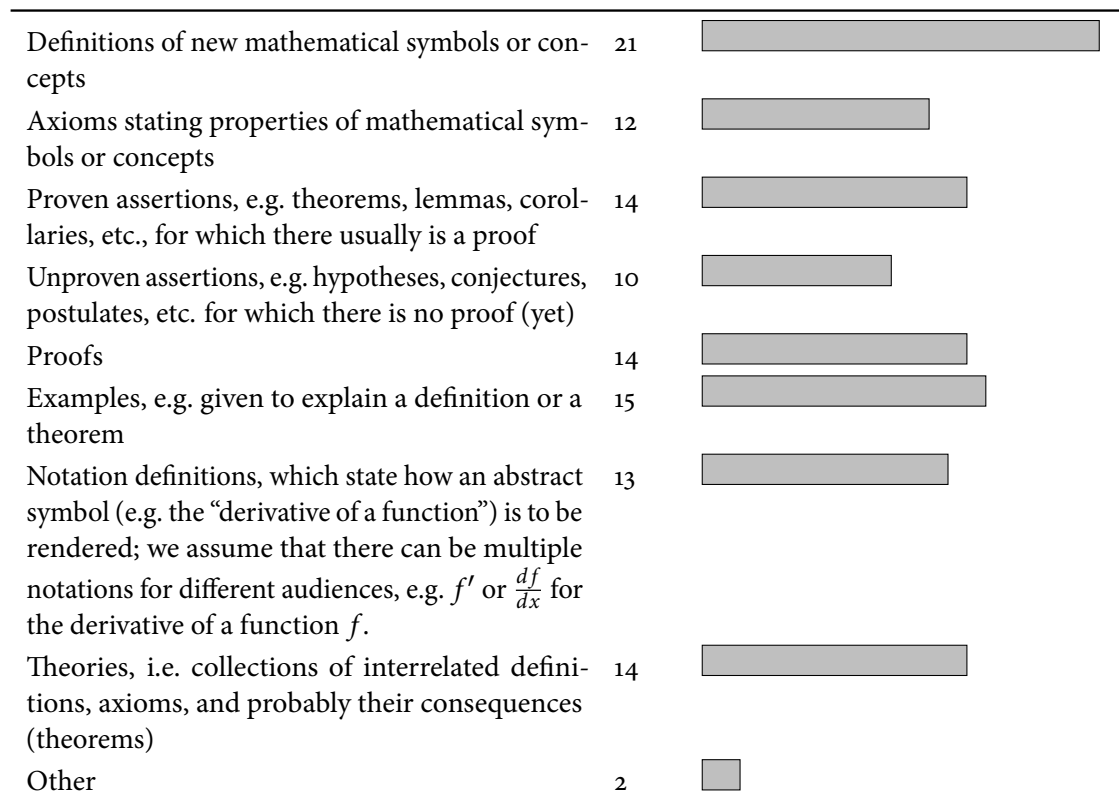
“Other” responses:

- “automatic generation of courses”
- “maintain and continually expand the learning and distribution capabilities of the network”

## Issues and Solutions

### What types of knowledge items have you ever experienced issues with?

26 participants answered this question.



“Other” responses:

- “proof steps, formulas”
- “implicit knowledge items, e.g. hidden home theories in a discussion”

### What kind of issues with knowledge items have you ever experienced?

21 participants answered this question.















It was incomprehensible	10	
It was wrong (applies, e.g., to a theorem or a proof)	10	
It was uncertain whether it was true or false	9	
It was not clear how to use it or how to apply it to solve a problem	5	
It was not clear whether it was useful	8	
It was not appropriate for the domain of interest (applies, e.g., to an example applying a theorem in a different domain, or to a notation that is not common in a certain domain)	4	
It had an uncommon style (e.g. a proof that used some very strange and unrelated method)	8	
It was not clear in what way a knowledge item related to another one (e.g. an example for something) actually was related to the other one	7	
It was underspecified (e.g. a definition leaving out an essential property of something)	9	
It was overspecified (e.g. a definition or an axiom imposing too many constraints on something)	6	
It contained too many independent subparts (e.g. a theorem stating two independent properties of something, or a theory containing axioms independent of each other)	5	
It was a reinvention of the wheel (e.g. a statement that had already been made before, in a different place of the knowledge base, probably in slightly different words or in an equivalent but different formalisation)	9	
Other	1	

“Other” responses:

- “how to represent adequately?”

### How was an issue with a knowledge item solved?

22 participants answered this question.








The affected knowledge item was fixed, changing its formal semantics (e.g. changing a definition)	12	
The affected knowledge item was improved, not changing its formal semantics, but improving its wording/structure/presentation	11	
The affected knowledge item was not changed, but a directly related knowledge item (e.g. the proof of a theorem) was improved	7	
A new, directly related knowledge item was created, e.g. an example to explain the original knowledge item, a proof, or a counter-example	8	
A sibling knowledge item, pointing to the same other knowledge item as the affected one, was created to provide an alternative, e.g. a second example for something, when the first example was criticised	2	
The affected knowledge item was split into more than one knowledge item	11	
Parts of the affected knowledge item were removed	9	
Parts of the affected knowledge item were replaced by references to other knowledge items.	5	
Other knowledge items were merged or integrated into the affected one	3	
The affected knowledge item was kept as an instructive example of how not to do it	1	
The affected knowledge item was deleted from the knowledge base	10	
The issue was not solved at all	9	
Other	0	

**If your experience does not fit into the schema of the previous questions, please explain it below, e.g. by describing the affected knowledge item, the issue, and the solution.**

The two responses given to this question were rather descriptions of collaboration environments than of problems and solutions; therefore I have omitted them.

**If an issue remained unresolved, why?**

14 participants answered this question.

Insufficient tool support for editing knowledge items	5	
Insufficient tool support for reorganising/refactoring/restructuring knowledge items	7	
Insufficient awareness of the users that there actually is an issue	6	
Insufficient social interaction (e.g. unresponsiveness of the maintainer)	6	
Users considered the issue irrelevant or illegitimate	4	
Users were no longer interested in a solution	2	
Other	1	

One “other” response has been omitted due to irrelevance.

## D.2 OpenMath Wiki Evaluation






15 people participated in this survey, whose results are summarized in [section 10.4](#). Four of them did not make it past the second question; their data were removed from the sample.

The spelling and formatting of free-text answers has been moderately adapted for readability.

### D.2.1 General OpenMath Questions

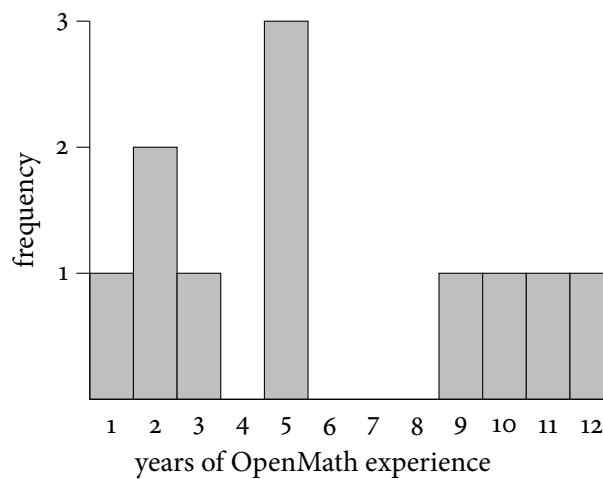
#### D.2.1.1 What is your role in OpenMath?

Multiple answers were possible.

Author of documents that use OpenMath	9	
Developer of Software using OpenMath	10	
Editor/maintainer/contributor of OpenMath CDs	8	
Organizational work in the OpenMath Society or community	6	
Other	4	

relevant “other” responses:

- “Interested in the area and I use it as an example project for MSc students to develop online CD maintenance and management systems”
- “writer of translations from/to OpenMath ”



### D.2.1.2 Since when have you been using OpenMath?

### D.2.1.3 How often do you work on the following CD-related tasks?

Throughout this questionnaire, “CD” refers to content dictionaries, signature dictionaries, and notation dictionaries.

	more than once a week	once a week	more than once a month	once a month	more than once a year	less often	never
Looking up information about CDs, symbols, CMPs/FMPs, ...	3	3	0	2	2	1	0
Reviewing CDs	1	1	1	0	5	2	1
Discussing about CDs	1	0	2	1	4	2	1
Presenting/publishing CDs	1	0	1	0	3	3	3
Other	1	0	2	0	0	1	7

“Other” responses:

- “Looking at the general structure of CDs and the reasons for their creation.”
- “I use the CDs in bursts: short periods with lots of accesses, long periods when I don’t access them at all”
- “work on transformations from or to OpenMath ”

**D.2.1.4 On what types of CDs do you work?**

	mostly	second	third	least
Official CDs approved by the OpenMath Society	7	2	2	0
non-standard CDs developed by yourself	3	2	5	1
non-standard CDs developed by others	1	4	4	2
Other	0	3	0	8

“Other” responses:

- “CDs developed by RIACA/our group (which is almost the same as ‘yourself’, but not quite)”
- “CDs developed by the RIACA group”
- “Most of my CD work is in the shape of OMDoc CDs”

**D.2.1.5 How do you locate information about a CD or symbol?**

	always	most of the time	some-times	rarely	never
I know what CD file to open	1	6	3	1	0
I do a text-based search on the CD files (e.g. grep)	0	2	1	5	3
I do an XML-based search on the CD files (e.g. XPath)	0	0	1	2	8
I browse human-friendly presentations of the CDs	2	7	1	1	0
I query a database that contains CD information	0	1	2	2	6
Other	0	0	1	0	10

“Other” responses:

- “... and that renders the formulæ!”
- “I browse the openmath website symbol list: <http://www.openmath.org/cdindex.html>”

**D.2.1.6 How do you edit a CD?**

	always	most of the time	some-times	rarely	never
Text editor	3	1	5	1	1
XML-aware editor	3	3	3	0	2
OpenMath-aware editor (please state below which one)	1	0	3	2	5
Other	0	0	3	1	7

relevant “other” responses:

- “My own OpenMath editor”<sup>1</sup>

<sup>1</sup>This comment refers to a tool that has been discussed in [section 6.2.1.4](#).

- “Wiris input editor sometimes, rather always QMath ”
- “Emacs with nxml package. Otherwise I occasionally use the tools developed by my MSc students.”<sup>1</sup>
- “I rarely edit CDs, but am interested in efficient ways of doing so”
- “but my text editor (Emacs) is XML OpenMath aware via nxml-mode”

### D.2.1.7 How do you create a new CD?

This may differ from editing a CD that already exists.

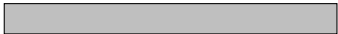




	always	most of the time	some- times	rarely	never
Text editor	2	3	1	1	4
XML-aware editor	3	3	0	0	5
OpenMath-aware editor	0	0	0	3	8
Interactive assistant (e.g. web form)	0	0	0	1	10
Automatically (e.g. generated from program code)	1	0	1	2	7
Other (state below)	1	0	1	1	8

“Other” responses:

- “My own OpenMath editor”
- “Not done this.”
- “Only using the tools developed by my MSc students”
- “I don’t create new CDs often, but interested in efficient ways of doing so”
- “but my text editor (Emacs with nxml) is XML OpenMath aware via nxml-mode”

### D.2.1.8 What communication media do you use to communicate about CDs?

Please check all that apply:

Personal communication	11	
Mailing lists	10	
Trac	1	
Wiki	3	
Other	2	

“Other” responses:

- “ActiveMath collections and preview. Scribbles for the earlier stage and also for debating steps, e.g. on Drupal.”
- “svn RSS feed (sometimes to keep up to date on what has changed)”

**D.2.1.9 How often do you use the OpenMath wiki for the following tasks?**

	more than once a week	once a week	more than once a month	once a month	more than once a year	less often	never
Looking up information about CDs, symbols, CMPs/FMPs, ...	0	0	0	1	2	4	3
Reviewing CDs	0	0	0	0	2	3	5
Discussing about CDs	0	0	1	0	2	1	6
Presenting/publishing CDs	0	0	0	1	2	2	5
Other	0	0	0	0	1	2	7

relevant “other” responses:

- “I have used the static CD pages so far.”
- “this only applies in the periods where I am actively working on CDs, which is not always.”

**D.2.2 Minor Edits**

This feature is described in [section 9.4.1](#) under the revised title “Quickly Fixing Minor Errors”.

**How do you agree with the following statements about such a feature?**

	strongly agree	agree	neutral	disagree	strongly disagree
This feature is relevant for OpenMath	6	2	1	0	0
This feature is relevant for me	3	2	2	1	1
It would save me time	2	2	3	1	1
It would give me new possibilities of working	3	3	1	1	1
I would enjoy using it	4	2	2	1	0

**Have you ever used this feature?**

Frequently	0	
A few times	1	
Never or only once	8	

Reasons for the latter:

- “I didn’t know this feature up to now.”



- “I have found errors in CDs, but I never knew how to fix them.”
- “I have a full local svn checkout of the OpenMath site and Emacs key bindings built into my brain, so I do XML editing in Emacs nxml mode almost exclusively.”
- “I did not need to do this.”
- “I hadn’t observed it”
- “Never needed to”
- “I am currently not so involved in working with CDs”
- “just for testing ”

### How well did it work for you?

	didn't do this	not at all	very badly	quite badly	moder- ately	quite well	very well
Navigating to the piece of the CD that had a mistake	5	0	0	1	2	1	0
Opening the affected piece for editing	5	0	0	1	1	2	0
Editing it	5	1	0	1	1	1	0
Saving it	6	0	0	0	1	2	0
Making use of the Subver- sion log message	6	0	0	0	2	1	0

### Any further comments on this feature?

- “I think this is important.”
- “It feels a bit slow, which is a bit of a problem with navigation (which people expect to be quickly). For the other tasks (opening, editing, saving) people might expect and accept a bit of waiting. I couldn’t find how to enter a log message. It was not clear that I should use the summary. I wasn’t sure whether this was a summary of the thing I was editing. When I saw no comment in the history I added a summary for the next edit and then noticed that this was put in subversion log. An easy solution is mentioning it in the documentation of the feature or to rename the field.”
- “I think this would be an excellent feature *if* there was some proper moderator support for it: namely that rather than generating lots of new, single entry CDs, if there was an interface for moderators to view all the suggested changes and accept or reject them, possibly after corrections. Otherwise there is a serious danger of large numbers of questionable suggestions clogging up management of the CDs.”



### D.2.3 Discussing Major Revisions

This feature is described in [section 9.4.3](#) under the revised title “Peer Review and Preparing Major Revisions by Discussion”.

### How do you agree with the following statements about such a feature?

	strongly agree	agree	neutral	disagree	strongly disagree
This feature is relevant for Open-Math	3	4	2	0	0
This feature is relevant for me	4	2	3	0	0
It would save me time	3	1	3	2	0
It would give me new possibilities of working	3	3	2	1	0
I would enjoy using it	3	2	4	0	0

### Have you ever used this feature?

Frequently	0	
A few times	2	
Never or only once	7	

Reasons for the latter:

- “I’m using the wiki for a few time.”
- “In practice, discussion happens on email lists perhaps about other subjects (in particular about the MathML 3 spec) and then only tangentially touches on CD issues, so discussion often carries on at the same place rather than switching to a wiki. It may be different if (as we were in the beginning) actively generating new CDs”
- “Discussion has taken place mostly on mailing lists.”
- “I feel the mailing-list is already sleepy enough!”
- “Never needed to”
- “I am not active”
- “just for testing once”

**How well did it work for you?**

	didn't do this	not at all	very badly	quite badly	moder- ately	quite well	very well
Posting an initial discussion post about some topic	4	0	0	0	1	3	1
Posting a reply to a discussion post	5	0	0	0	1	2	1
Choosing a type for a discussion post	5	0	0	0	3	1	0
Finding open discussions from the main page	5	0	0	2	2	0	0
Reading existing discussions	5	0	0	0	2	2	0
Configuring e-mail subscription for discussion post	6	0	0	1	1	1	0
Receiving and reading notification e-mails	7	0	0	1	0	1	0

**Any further comments on this feature?**

- “I really like the feature, but it needs a lot of polish.”
- “It might be nice to be able see the (topics of) discussion items for the whole page and/or to navigate to them. This might work better for people better used to working with wikis. With some experience choosing a discussion type and finding open discussions, as well as configuring e-mail would work quite well.”
- “I am sure this works well for those who actively are creating CDs. I wish there was when I was editing and submitting CDs”
- “In general, I think it is a useful feature, but I would prefer it to be accessible directly from the CDs web page, rather than on a separate wiki”
- “Trouble is... why can't we use existing infrastructures to do this communication, mailman being the tool of choice, instead of adding yet another spot to monitor.”

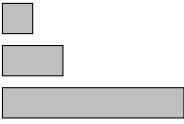
**D.2.4 Editing and Verifying Notations**

This feature is described in [section 9.4.2](#) under the revised title “Fixing and Verifying Notations”.

**How do you agree with the following statements about such a feature?**

	strongly agree	agree	neutral	disagree	strongly disagree
This feature is relevant for OpenMath	6	3	0	0	0
This feature is relevant for me	4	2	2	1	0
It would save me time	4	0	2	3	0
It would give me new possibilities of working	3	1	3	2	0
I would enjoy using it	4	0	5	0	0

**Have you ever used this feature?**

Frequently	1	
A few times	2	
Never or only once	6	

Reasons for the latter:

- “Never had the necessity”
- “I maintain the main ‘alternative’ to these notation declarations, the XSLT stylesheets to MathML, so these come more naturally to me.”
- “no need.”
- “I don’t really edit CDs: I am interested in the CD management problem, rather than the CDs themselves.”
- “I am not active”
- “didn’t found out how to test this in the SWiM test.”

**How well did it work for you?**

	didn't do this	not at all	at	very badly	quite badly	moderately	quite well	very well
Navigating from a mis-rendered symbol to its notation definition	6	0	0	0	1	1	1	0
Editing the notation definition	7	0	0	0	1	1	0	0
Verifying whether you got the notation right	6	0	0	0	0	2	1	0

**Any further comments on this feature?**

- “Probably wouldn’t use it often, but it seems useful.”

## D.2.5 Other Wiki Features

### Other Supported Features

Short judgments on the following other features of the OpenMath wiki.

1. Do you consider a feature relevant?
2. Have you used it?
3. If so, how well did it work?

	relevant	I have used it	worked very badly	quite badly	moder- ately	quite well	very well
RSS feed of recently changed pages	9	2	0	0	0	1	1
User permission management	9	0	0	0	0	0	1
Sample queries (linked from main page)	8	1	0	0	1	3	0
Possibility to do your own queries	6	1	0	0	2	0	0
Interaction with the Sub-version repository	9	1	0	0	0	2	0
Semantic navigation ("References" box)	9	1	0	0	0	1	0
Symbols being linked to their CDDefinition (middle mouse button)	8	2	1	0	0	1	0
Document-oriented editor for CD structures	6	1	0	1	0	0	0
Form-based editor for metadata	5	1	0	1	0	1	0
Visual formula editor	8	1	0	0	1	0	0
Full-text search	7	0	0	0	0	0	0
Online help (FAQ, troubleshooting, other explanations)	9	1	0	0	1	1	0
Usability in general	7	2	0	0	3	1	0

### Wishlist

How much would you wish to have the following features?

	not all	at	not nec- essarily	some- what	much	very much
A text search that works better	0		4	2	0	2
Formula search	0		1	2	3	2
Global search/replace for text	0		2	4	2	0
Global search/replace for CD structures	0		2	4	2	0
Global search/replace for formulae	0		3	3	2	0
Assistance with implementing solu- tions for common problems	0		2	3	1	2
Better Subversion integration (cur- rently only update/commit/lock sup- ported)	0		4	1	1	2
Usable support for adding new content (CDs, symbols)	0		0	3	1	4
Ability to link to discussion posts	0		0	4	2	2
More appropriate types for discussion posts	0		3	3	1	1
A personalized view on your topics and discussions of interest	0		3	2	1	2
More interactive features in documents (e.g. customizable notations)	0		2	4	1	1
Editing of types (e.g. STS)	0		1	1	4	2
Type-checking of expressions	0		1	2	1	4
CAS integration	0		3	2	0	3
Dedicated CD review workflow	0		2	0	2	4
Other	5		1	1	0	2

“Other” responses:

- “Help with checking the ‘formal mathematical properties.’”
- “I really wanted to enter ‘No Opinion’ for all of this question, but that was not an option so I entered ‘Somewhat’ for everything here. By far my biggest problem with the wiki is its poor speed and reactivity. I am really put off using it, or even exploring it properly, before this issue is fixed.”<sup>2</sup>
- “speed up (if possible) especially for (simple/CD) navigation.”  
“easy link to parent if there is one (symbol→CD→CD group)”

#### D.2.5.1 General Comments

- “The wiki seems a good idea. It makes it easier to make small changes or start discussions on them (for those who do not like to edit the CDs themselves).”
- “I am afraid that my comment is really of the unhelpful form ‘If I were you I wouldn’t start from here...’ (Sorry): My preference would be to not have a separate wiki from the CD

<sup>2</sup>In response to this statement, all answered of that participant have been removed from the table given above, except for the “other” option.

management site but to have the wiki features fully integrated: users could attach discussion threads to individual CDs or definitions in CDs, when viewing a CD, the discussions would be available there, just like the FMPs, CMPs etc. Suggestions could be made in place but moderators could log in to vet them, reject or accept, even a voting system on changes could be supported if required. One of the critical barriers to this is getting a much better quality OpenMath to MathML translator working. All of the currently available ones have serious problems.”

- “One problem with (any) wiki which isn’t really addressed in the survey is editorial control. For both OpenMath and MathML we need, or seemed to need, close editorial control and consistent editorial style over the whole collection. This is somewhat at odds with the more fluid editorial style that tends to be adopted in a wiki. In particular the versions of the CDs in the wiki were based on a somewhat speculative set of CDs from the svn repository which in the end we couldn’t use for MathML 3, which needed to be based on the CDs at [openmath.org/cd](http://openmath.org/cd) and with text that was copied into chapter 4 of the MathML spec and then edited consistently rather than (as was tried at the start) dynamically pulling each symbol description from the relevant CD, which produced a far too disjointed chapter of the MathML spec”
- “The system seems not completely stable. I have received several warning windows and unknown errors”

### D.3 OpenMath Wiki Usability Experiments

#### How to move the cursor one character forward in vi

*The correct answer is: <ESC>l a which works in all modes. Except at the beginning of a line, where the above command will move the cursor two characters forward. If it did anything else, it would not be vi. So at the beginning of the line, this answer is the correct one: <ESC>li. And of course neither will work at the end of the line. At the end of the line, the correct command is: <ESC>j^i. The topic of the next two lectures will be “how to move the cursor one character backward in vi”.*

—Per Abrahamsen [Abr]

14 test persons participated in the usability experiments. [Section 10.5](#) explains the setting and summarizes the results; detailed logs are given here.

Some test persons gave more detailed feedback on these features, or additional feedback on other features. Literal quotes from the test persons are quoted and highlighted in italics. “*Just like this sentence.*” My own interaction with the test persons, such as instructions or explanations given, is quoted and highlighted in a sans-serif font. “Just like this sentence.” Further explanatory comments for the reader are given as footnotes.

I classified distinct feedback statements in the following way:

- ☺ Positive feedback, subclassified into:
  - U** The user gave explicit positive feedback about a feature
  - T** Actions or verbalizations show evidence that the user understood how to accomplish a certain task, and it worked
  - D** Verbalizations show evidence that the user understood a design concept behind the user interface.



- ⊙ Negative feedback, subclassified into:
  - U** The user gave explicit negative feedback about a feature.
  - E** The user explicitly stated that the system did not meet his expectations
  - N** Verbalizations show evidence of dissatisfaction with an aspect of the interface.
  - C** Verbalizations show evidence of confusion/uncertainty about an aspect of the interface.
  - S** Verbalizations show evidence of confusion/surprise at the outcome of an action.
  - T** The user tried a wrong approach, or verbalizations show evidence that the user did not know how to accomplish a certain task.
  - D** Verbalizations show evidence that the user did not understand a design concept behind the user interface
  - P** The user wanted to do the right thing but then experienced an unexpected problem of which I had not been aware of before, or a bug of which I had been aware of before starting the experiments.
- ⊛ The user made a suggestion. There is no clear distinction between suggestions and “negative feedback” in the sense of the system not meeting a user’s expectation; I tended to classify requests as ideas that were to-the-point constructive suggestions, or that the users themselves classified as “nice to have” or as enhancements, or whose realization in SWiM would be a major effort, as compared to just fixing a minor bug.
- ▷ This groups a list of subtopics, ...
  - such as this one.

### D.3.1 User 1 (2009-12-03)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

- ⊙E expected a text box in place
  - ▷ “open this” links:
    - ⊙T didn’t figure out that it opens the section *starting* there (but rather clicked on the following “open this” link)
    - ⊛ should be directly attached, e.g., to the section headline, instead of showing up far on the right, in order to make its functionality more obvious
  - ⊛ on pages with little content, there should be a horizontal rule designating the end of the content
  - ⊛ expected the “edit” button at the bottom of the window<sup>3</sup>. – “The user interface is similar to MediaWiki.”
    - ⊛ Generally, the page actions should be grouped more logically. A user may not be used to MediaWiki.

In the document editor:

- ⊙C confused when moving the text cursor around, e.g. into the bold labels of the metadata fields
- ⊙N particularly irritated by the “non-text” “=” sign, found it easy to get something wrong when editing metadata

In the metadata editing form:

<sup>3</sup>The user had previous experience with Trac, where the button for editing a wiki page is at the bottom.

- ☹U did not find it obvious that a double click was needed to edit a field; the field rather seemed to be read-only
    - ✧ Suggestion: don't put the "delete" button into a column titled "action" if that is the only action that can be performed
  - ☹U too many metadata fields related to IkeWiki maintenance
    - ✧ maybe rename metadata fields
  - ☹T figured out himself/herself that "Return" saves a metadata field.
- Subversion log message:
- ☹U fine

### Fixing and Verifying Notations

Navigating to the symbol and notation definition:

- ✧ There should be some online help saying that you have to middle click on symbols.
- ☹T/D It was not clear that the notation definition of a symbol is linked from the symbol via the navigation tree.
  - ✧ Better labels should be provided for the RDF links in the navigation tree.

Preview of the notation definition:

- ☹N The meaning of "arg1" and "arg $n$ " in the preview of the rendering of an  $n$ -ary operator was not obvious.
  - ✧ use subscripts, e.g. "arg<sub>1</sub>"
  - ✧ There should be a larger padding of the cells in the table
  - ✧ "I have been thinking about coloring." – Yes, or highlight matching parts on hover
  - ✧ distinguish symbols from variables

Editing:

- ☹T no idea how to edit – "click 'edit'"
- ☹U not familiar with cluttered tables
  - ✧ there should be a palette for inserting Unicode symbols<sup>4</sup>, or support for XML-like entities, e.g. `&#1234`;
- ☹C/D not sure whether I'm editing the right table – in the rendered view, the preview is on the right side, but now in the editor one is supposed to *edit* on the right side?

Complete workflow:

- ☹U The idea that you can edit and instantly see the result is good
- ☹U did not like it that symbol links were only accessible by the middle mouse button
  - ✧ It should be particularly easy to edit the *rendering* of a symbol; usually you don't need to edit the *prototype*.
  - ✧ It should be easier to change a rendered symbol, e.g. a button "change notation" next to rendered symbols in formulæ, then a popup dialog, where you can select the desired rendering from a symbol palette, or enter a Unicode character.

### Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface:

---

<sup>4</sup>Such a palette exists in the TinyMCE toolbar (a button with an  $\Omega$  icon), but it may not be obvious to identify

- ✧ In the popup menu for posting a position, the order should be more logical, namely “agree/neutral/disagree” instead of “agree/disagree/neutral”
- ☹N What if I agree but have one concern? (I.e. how to combine two argumentative types?)
  - ✧ The untyped “reply” button should be more clearly separated from the others
  - ✧ Domain-specific issues: “A definition – in the strict sense – cannot be wrong.” – But there are wrong definitions; consider RUSSELL’s paradox
- ☹E “The drop-down list for selecting a specific Issue/Idea type must not be editable; otherwise I wonder what I should enter there.”
  - ▷ Icons:
    - ☹U The “issue” icon is not intuitive.
    - ☹N The difference between the icons for “evaluation” vs. “position” is unclear.
  - ✧ There should be a lighter editor, maybe only show the heavy editor on request
  - ✧ Assistance (cf. [section 6.6.3](#)): can’t judge whether it would work, because there are no *real* discussions here.<sup>5</sup>
  - ☹U “I don’t like user interface, nor the colors used.”
  - ☹U “The workflow is nice for mathematical discussions.”

E-mail notification:

- ▷ Subscription:
  - ☹E The “auto-watch” checkbox belongs into a user profile dialog, not here.
  - ✧ Suggestion: “[watch this](#) / [unwatch this](#)” as a link

Queries:

- ✧ Hide queries from the editor; they might confuse newbies.
- ☹U Queries combining mathematical and argumentative structure would be useful
  - ▷ suggested further queries:
    - ✧ “symbols with  $\geq 2$  notations, of which one has an issue”
    - ✧ What CDs are used most?
    - ✧ Who reused symbols from CDs that I wrote?
  - ✧ When a discussion is about a symbol, the rendered symbol should be shown in the query results.
- ☹U The results of the query for ongoing discussions are not presented nicely; it is not obvious what the discussions are about.
  - ✧ A discussion with two issues should show up as two query results
  - ✧ Have each discussion on a separate line, the most recent one on top.
  - ✧ Display a limited subset of query results if there are many; offer an “expand” button (e.g. “+”) that will show more

## General Feedback

- ✧ Pay more attention to small usability features

<sup>5</sup>There were some real discussions in the wiki (cf. [section 10.3](#)), but we did not study them for this experiment.

### D.3.2 User 2 (2009-12-04)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

- ▷ “open this” links:
  - ☹U It is a bit confusing to understand where it belongs
  - ☹S After clicking “open this”: “*Now I’m seeing the same again?*”
- ☹U Where is the “edit” button? – Too far away

In the document editor:

- ☹U Display of XInclude links is confusing
- ☹U Editing of metadata is unintuitive
- ☹E I cannot add anything.<sup>6</sup>
- ☹E Some of the “forbidden” markup (e.g. metadata labels) can be edited

In the metadata editing form:

- ☹U Liked this better [than the document editor]
- ☹U It takes too long to load
  - ✧ Enable cursor key navigation between the cells, like in Excel

Other comments:

- ☹C How do I navigate to the parent page?
  - ✧ Breadcrumbs for navigating back
- ☹U Some of the labels in the navigation tree are a bit unintuitive
  - ✧ Formulæ should directly be editable in the formula editor, without going through the document editor; that formula editor window should be accessible via its own URL for more flexible integration
  - ✧ Similarly, informal sections should be directly editable in a text box.
  - ✧ For text mixed with formulæ, the TinyMCE view could simply contain links to the formula editor instead of also making the formulæ editable in linear syntax
- ▷ Formula editor dialog:
  - ✧ The linear syntax of a long formula should spread over multiple lines
  - ✧ The functional tree structure should be exposed.

Subversion log message:

- ☹U helpful

#### Fixing and Verifying Notations

Navigating to the symbol and notation definition:

- ✧ Symbol  $\xleftarrow{\text{rendersSymbol}}$  NotationDefinition: inverse relation would be more intuitive

Preview of the notation definition:

- ✧ There should be a larger padding of the cells in the table
- ☹U Looks good!
- ✧ Add a “prototype preview”, i.e. `<OMA><OMS cd="the" name="symbol"/><OMSTR>arg1</OMSTR>...</OMA>`
- ✧ In the prototype, use colors for *expr/@name*

---

<sup>6</sup>Metadata can be added here, but not in an obvious way (cf. [section 8.3.3](#)).

Editing:

☺U The two-column arrangement makes sense.

☺U The tables use too much space.

Complete workflow:

☺U Actually good, ...

☺U but the document editor is too clumsy

## Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface:

- ✧ Offer local links to the discussion forum, e.g. in the place where “open this” is at the moment
- ✧ Think about offering a review workflow

Queries:

- ✧ Display the subject of a discussion thread as a part of the query result

### D.3.3 User 3 (2009-12-04)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

☺T picked the right “open this” link.

☺C slightly confused: “*Can I edit it here?*”

In the document editor:

☺P Tried to remove a metadata field; exception occurred on save

☺U Metadata vocabulary is different from the terminology used in the rendered CD  $\Rightarrow$  confusing

In the metadata editing form:

☺P Encountered the cursor highlighting bug<sup>7</sup>

☺N In the drop-down list for adding a new property, only the formal identifiers are shown

- ✧ Offer metadata language selection from a drop-down list instead of forcing the user to enter a two-letter string<sup>8</sup>

Subversion log message:

☺U helpful

☺E when changing a metadatum in the document editor, its name does not show up in the log message<sup>9</sup>

Other comments:

☺P no “open this” for *discussion* children of a symbol definition;<sup>10</sup> they are only linked via the general *hasDirectPart* relation, but not via a more specific one, such as *hasDiscussion*

☺U not always intuitive what links are incoming vs. outgoing

<sup>7</sup>Most users were irritated by this bug; in subsequent experiments I gave instructions for a workaround and did not record this bug any more.

<sup>8</sup>While this is generally a valid concern, the OpenMath CD language does not have any means of expressing multilingual metadata, which makes this feature of IkeWiki useless here.

<sup>9</sup>We would need an XML diff in order to identify the change made in the document editor (cf. [section 9.6.4.2](#)).

<sup>10</sup>*discussion* elements were an experimental feature of the OpenMath 3/MathML 3 draft CDs.

- ☹N no dependencies/imports shown<sup>11</sup>
- ☹P the “prefix form” view of a formula does not have valid links to symbol definitions

### Fixing and Verifying Notations

Navigating to the symbol and notation definition:

- ☹S didn’t notice that the symbol definition opened in a new tab after the middle click<sup>12</sup>
- ☹S clicking on a rendered symbol does not lead to the notation definition but to the symbol’s (semantic) definition

Preview of the notation definition:

- ☹U The terms “prototype” and “rendering” are not intuitive for non-experts.

Editing:

- ☹E Cannot enter a Unicode or  $\LaTeX$  symbol
- ✧ Fold those large HTML tables
- ☹P While editing the cell for the content of an *mo* element (i.e. 

mo
<b>this one</b>

), one must not hit “Return”, as that would insert an HTML *p* element, which would then confuse the parser on saving.

Complete workflow:

- ☹U getting to the notation definition is too cumbersome
  - ✧ suggested improvement: right click on symbol shows context menu with entries “show definition” and “show notation”
- ☹D didn’t know *when* to use the navigation tree
  - ✧ more self-explanatory labels in the navigation tree, e.g. “go to notations”

### Peer Review and Preparing Major Revisions by Discussion

Argumentation ontology:

- ☹D didn’t understand difference between “idea” and “supporting argument”
- ☹D otherwise understood the argumentation ontology quite well intuitively

Discussion forum user interface:

- ☹U mostly clear
  - ✧ for an *Example* supporting an *Issue*, use a “!”-like icon, as it is an *Idea*-like discussion post
- ☹N/U posts not ordered by time. “*That destroys the order of replies.*”
  - ✧ suggestion: identify, display and print out conflicts, e.g. “User A and B conflict, so let’s ...”<sup>13</sup>
  - ✧ count votes, display the count in the discussion user interface

E-mail notification:

- ☹D understood how it works
  - ✧ suggestion: select argumentative types of posts about which I want to be notified
- ☹E reveal a bit more of information about the post in the notification e-mail, e.g. “new idea”, “decision made”

---

<sup>11</sup>This is a valid concern w.r.t. dependencies, but OpenMath CDs do not have import declarations.

<sup>12</sup>The user was not used to Unix-like systems, where this is the default browser behavior.

<sup>13</sup>This has indeed been one of the design goals of the DILIGENT model adopted here (cf. [section 6.6.5](#)).

Queries:

- ☺U queries for discussions are helpful
- ☺P clicking on URIs containing a “+” doesn’t work

### D.3.4 User 4 (2009-12-11)

#### Quickly Fixing Minor Errors

Navigating to the editor:

- ☺T intuitively understood “open this” link
- ☺T first went to the metadata editor because he or she had instantly spotted that tab (but also recognized the “edit” tab)
  - ▷ After editing a symbol: *“How do I get back to the page of the complete CD?”*
    - ☺E “Back” button does not work – “due to excessive usage of AJAX ”
    - ☺T Navigation via “references”→“incoming” was intuitively clear<sup>14</sup>
    - ☺D grasped that the symbol definition is a semantic object, ...
    - ☺T ... therefore noticed the incoming link “contains symbol definition”
    - ☺T *“Can I also open a mathematical property via ‘open this?’”*
      - ✧ In the navigation tree, the way back to where you came from should be unfolded by default.
  - ☺S When using the linked data style navigation navigation links while in the “edit” tab, the user found it unintuitive that the edit tab remains active, i.e. that the control is not handed back to the article tab.

In the metadata editing form:

- ☺N *“I can only delete here?!”*
- ☺C *“How do you save?”* – “Return.”
  - ✧ Saving with “Return” is easier than opening a field for editing (= double click), but actually *opening* should be easier
- ☺S What does the status indicator in the lower right corner do? “It indicates that your edit is being saved.” – found that irritating
- ☺C *“Can I cancel an edit?”* – Yes, with “Escape”.
- ☺T Wondered whether the “annotate” tab is somehow related to metadata
- ☺E Expected the metadata edit to show up in the history

Subversion log message:

- ☺N *“Why does it show dc:description but not the label shown in the metadata editor, or the name of the OpenMath CD XML element?”*
  - ✧ Instead of *cd:swimtest+swim*, show human-friendlier names of resources, such as “symbol *swim* from the CD *swimtest*”.
- ☺C *“When I switch the language to German, does the Subversion log also become German?”* – “Unfortunately not, and IkeWiki’s language switching is not supported by the SWiM extensions anyway ☺”

Other comments:

<sup>14</sup>The user had previous experience with semantic annotation and links.



- ✧ If language switching were supported (see above), the labels in the rendered document (e.g. “Description”) should be multilingual, but still the connection to the OpenMath CD XML element should be clear.<sup>15</sup>
- ▷ on “minor edits”:
  - ☺U “What does ‘minor edit’ mean; why is this scenario called ‘*minor* edits?’” – One can also make a *major* edit with the same interface.
  - ✧ “One could introduce a way of marking an actual minor edit as ‘minor’, as, e.g., in MediaWiki” – “Yes, *this feature is missing.*”

## Fixing and Verifying Notations

Rendered objects:

- ▷ (re)presentations of a mathematical object:
  - ☺T/D What do the green buttons mean? “They reveal different (re)presentations of an object.” – “*I didn’t know that these are four views on the same.*” – not intuitive
  - ✧ use more informative labels for these buttons

Navigating to the symbol and notation definition:

- ☺D understood that the notation of a mathematical object is in Presentation MathML
- ☺T tried “open this” for the mathematical property
- ☺T “*I want to change the presentation*” – clicked on “presentation” (the button that toggles the Presentation MathML display of a mathematical object), then on the “edit” tab
- ☺E expected “*presentation editor*” there
- ☺S “Middle click on the symbol.” – “*Did I open it?*” (didn’t notice the new tab)<sup>16</sup>
- ☺C looking at the symbol definition that was opened, confused: “*Where is the notation?*” – “Navigate there via the navigation tree!”

Preview of the notation definition:

- ☺C Why “prototype”?
- ☺U Rendering is green – “*Good, that way I see that it is output.*”
- ☺C “*I see  $\arg 1$ , but where is that in the prototype?*”
- ☺N Important distinction: The rendering preview is the rendering of an *instance* of the prototype (i.e. `<OMA><OMS cd="the" name="symbol"/><OMSTR>arg1</OMSTR>...</OMA>`), but not the prototype itself.
- ✧ Extend preview by examples: What mathematical objects are concrete instances of this rendering?

Editing:

- ☺C Why is the toplevel element `mcd:notation`<sup>17</sup>? “*I didn’t see that in the ‘article’ view.*”
- ☺T “*I recognize the operator symbol ...*”
- ☺N “*... but do not understand much otherwise*”
- ☺C/☺U “*What are the empty white fields for?*” – “elements without text content” – “OK, that is like XML, one can get used to that.”
- ☺E Hitting “Return” in the edit summary field does not work

<sup>15</sup>This is obviously easy to realize for English, but harder for other languages.

<sup>16</sup>another Windows user

<sup>17</sup>The notation definition markup is now in the OMDoc namespace.

Complete workflow:

- ✧ Context-sensitive notation selection (e.g. according to settings in the user profile) would be important
- ✧ Looking at a complete notation dictionary: “A table of contents would be nice.”

### Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface:

- ▷ Posting a new *Idea* for an existing *Issue*:
  - ☹N “In the ‘new post’ form I don’t see that I have clicked on ‘Idea’ before.”
  - ☹U nonsense that the subject of my *Idea* is predefined by the subject of the *Issue* I’m replying to.
- ☹S Confused by the status bar enabled in TinyMCE, which shows the path to the currently selected HTML element.
- ☹C Challenging *Argument* – “How would I recognize such a post?” – “At the moment only from the ‘thumbs down’ icon” – was not clear to the user
  - ✧ Provide a legend of the icons, particularly for clarifying the appearance of ratings (positive/negative/neutral *Positions*)
- ☹N chronological order of the posts unclear
- ☹U good to have untyped posts as an alternative

Argumentation ontology:

- ☹D Difference of *Argument* and *Position* was clear
- ☹E When there is an *Example*, why can’t one reply with an *Argument* or an *Elaboration*, e.g. “I agree with regards to content” (as opposed to a merely subjective *Position*), or “it is a bad example”

E-mail notification:

- ▷ Subscription:
  - ☹U “automatically watch” checkbox is inappropriate, does not belong here
  - ☹U “automatically watch” and “watch/unwatch” do not fit together
    - ✧ Use “save” instead of “OK”. “Actually I expected that my settings would directly be saved after clicking the checkboxes.”

Queries:

- ✧ Would be interesting to have a list of resolved issues (for experience management and e-learning)

### General Feedback

- ☹U a lot of usability issues
  - ☹ an evaluation of the wiki in a collaboration scenario is missing
    - ▷ this experiment was titled “usability evaluation”, but actually it was also about understanding the semantic structure behind the user interface:
      - document vs. graph structure
      - “Where you have chosen semantic objects different from the ones typically occurring in wikis, has a user used them?”

- “What new possibilities do people have?” – analyze the potential
- “Did the people become aware of the nature of the semantic objects they have worked with?”

### D.3.5 User 5 (2009-12-16)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

- ☹E expected to edit description directly by clicking
  - ▷ “open this” links:
    - ☹C not *exactly* clear what it does
    - ☹T nevertheless identified the right one (“*analogous to MediaWiki*”)
    - ☹E expected direct access to the edit mode
    - ☹U “*The user does not understand the granularity*”
- ☹T instantly found and used the “edit” tab

In the document editor:

- ☹T instantly found the description field
- ☹T wrote summary, saved

In the metadata editing form:

- ☹C/☹T “*Can I click into the fields?*”
- ☹C “*How do I save?*”
- ☹E no possibility to give a summary here

Subversion log message:

- ☹T user recognized his/her changes
- ☹U didn’t find *swimtest+swim* completely obvious
- ☹U on the other hand, for well-known CDs and symbols e.g. *arith1+plus*, it is quite OK; then, it is short and concise
- ☹E you only see *that* a metadatum was changed, but not what *exactly* has changed
  - ✱ use “/” as a separator

#### Fixing and Verifying Notations

Navigating to the symbol and notation definition:

- ☹E/T tried to click on the symbol, nothing happened, frustrated – “middle button”
- ☹U It is easy to miss the symbol when clicking
- ☹D It was clear to the user that clicking leads to the symbol’s definition
- ☹T/D “*Outgoing – those seem to be irrelevant*”
- ☹D “*Watches? – that is related to watching/viewing*”<sup>18</sup>
- ☹D “*rendersSymbol’ – that is related to appearance*”
- ☹T “*How do you expand a tree?*” With “+”?

Preview of the notation definition:

- ☹N couldn’t recognize the prototype in the rendering

---

<sup>18</sup>There was a link from the article to a person watching it (actually: its discussion forum) for changes, labeled “watches”.

☺D “args” corresponds to “arg1...argn”, separated by the operator symbol?

✧ would like to see the source code of the rendering in an additional column

Editing:

☺U/☺T “I didn’t read all that [the tables]” – just spotted the operator symbol

☺T entered summary

☺T saving worked

Complete workflow:

☺U rendered objects are a bit small, ...

✧ ... suggested a larger font

## Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface/Argumentation ontology:

☺C/☺T read the tooltip for “new issue”, then clicked “new comment”

☺S Confused by the TinyMCE status bar

☺D “Why didn’t you choose to post an *Issue*?” – “Because I found ‘wrong’ too strong”<sup>19</sup>

✧ would have preferred something weaker, like Bugzilla’s “new feature”

✧ would like to enter Presentation MathML

☺T would study existing discussions in order to learn the argumentative structure

✧ position for voting? – A poll user interface would be better.

☺C/☺T “Agree with an Idea? What happens here?” – writes a reply to an idea

☺T next replies to the same idea with a challenging *Example*

☺T always changes the subject manually

☺D “The reply buttons are restricted context-sensitively”

✧ “I would like to directly post a counter-argument to an argument”

Queries:

☺U good that this feature exists

▷ query result lists on the main page:

☺N not clear whether they are disjoint; disjoint lists would probably be more clearly arranged

☺U separation of query results by dashes is confusing

▷ custom SPARQL queries:

☺U “If I worked with the system more often, I could imagine to write some queries that are relevant for me.”

☺D “But for that I would have to know the ontology. How do I get access to it and to its documentation?”

### D.3.6 User 6 (2009-12-18)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

<sup>19</sup>The tooltip for the “New Issue” button reads: “something that is wrong with this knowledge item and should be solved”.

▷ “open this” links:

☹T tried “open this” for the mathematical property, as that was the closest link

☹E expected something like “open this” on the same line as the description, which directly leads to the editor, or directly click to edit in place

☹N “the ‘edit’ toolbox [on the left of the screen] does not contain relevant stuff”

☹T did not initially notice the tab bar (possibly due to its light color)

☹T opened “metadata” tab ...

☹S/N ... but then left it again, as he or she didn’t find any obvious possibility to *edit*

In the document editor:

☹P inserted a line break

▷ XInclude links to sub-fragments:

☹P can’t follow any link to included documents

✧ would be more intuitive to have them out of the main view, e.g. below, as we can’t edit them anyway

In the metadata editing form:

☹C “I can’t edit” – didn’t see any possibility to edit

☹E expected a “save” button – saving by pressing “Return” was not obvious

Subversion log message:

☹C wondered why the log message said (no comment)

☹N “You can enter a summary in the editor.” – Probably hadn’t noticed the edit summary text box due to distraction by the box-like XIncludes.<sup>20</sup>

### Fixing and Verifying Notations

Navigating to the symbol and notation definition:

☹P the “prefix form” view of a mathematical object does not have valid links to symbol definitions

☹T it was not obvious that one had to use the middle mouse button, ...

☹S ... and that the symbol opened in a new tab

☹T found “rendersSymbol” in the navigation tree

Preview of the notation definition:

✧ on the page of the symbol, there should be a directly editable “notation” field (as the user had requested for the description and other symbol metadata before)

Editing:

☹T editing worked

☹U found tables representing prototype and rendering and their children disturbing

✧ would like a symbol palette

Complete workflow:

✧ would like to hover over a mathematical object to see alternative notations: “This symbol can also be rendered as ...”

---

<sup>20</sup>This is possibly a “*later rationalization*” [Nie93] that should not be overrated. As additional recording equipment, such as an eye tracker, was not available, the actual reason why the user overlooked the summary field is unknown.

## Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface:

- ☺T looked at an existing thread, posted an *Elaboration* on an *Idea*
- ☺N random post ordering
- ☺C confused by structure
- ☺E expected an *Elaboration* to be shown in the same box as the *Issue/Idea* it elaborates on
- ☺U found icons helpful
- ☺U found supply of reply types OK

Argumentation ontology:

- ☺D wanted to say “this is the case because ...” and posted an *Elaboration* – it was not clear that here an *Argument* would have suited better

E-mail notification:

- ☺U found subscription configuration GUI self-explanatory
- ☺E include the title of the thread in the e-mail if there are multiple threads in the forum

Queries:

- ✧ The warning boxes suggesting assisted problem solutions should link to the discussion thread containing the suggested idea, so that the user can check that once before running the assistant.
- ▷ Query result lists:
  - ☺U a good feature
    - ✧ but would prefer a link to the thread instead of a link to the article
- ▷ Writing queries:
  - ☺N You have to know SPARQL – “Eventually, there will be a GUI for composing queries”
    - ✧ In such an interface, allow for selecting the type of resource to search, as well as filter criteria from drop-down lists with reasonable preselections, then generate SPARQL from that

### D.3.7 User 7 (2009-12-21)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

- ▷ “open this” links:
  - ☺U found them to work as in MediaWiki, ...
  - ☺E ... but expected direct access to editor
  - ☺D comment about the target of the link: “*This is only the symbol.*”
- ☺T looked at “edit” toolbox on the left
  - ✧ local access to the editor – just next to the rendered text – would be most intuitive
- ☺T on the symbol page, tried “open this” of the property, then opened the “Edit” tab

In the document editor:

- ☺P inserted line break
- ☺T saving was clear
  - ✧ suggested improvement for summary field (see “Subversion log message” below): have a help text “enter summary here” as an initial content

In the metadata editing form:

- ▷ saving:
  - ☹U found it OK that one metadata field is saved if one selects another field with the mouse
  - ✧ but the same should also work when clicking into the whitespace outside of the whole metadata table

Subversion log message:

☹U/D “*I can see what has actually changed.*”

☹C/☹D why (no comment)? – “*Ah, because of the ‘summary’ field in the editor*”

☹U “Why didn’t you enter a summary?” – “*If it [the summary field] is empty, it doesn’t mean much.*” (see suggested document editor improvement above)

Other comments:

- ✧ could use a load progress indicator (as when opening the metadata form) also when loading a rendered page takes long
- ☹T How do you navigate to the main page?
  - ▷ navigation tree:
    - ☹N text is quite small
    - ✧ put references on top of the page
    - ▷ open most relevant trees by default, such as:
      - ✧ incoming *usesSymbol* link ( $\hat{=}$  “appears in”) for a symbol definition
      - ✧ outgoing *hasDiscussion* link to the discussion page<sup>21</sup>
  - ▷ (re)presentations of a mathematical object:
    - ☹U green color for an “opened” representation is useful
    - ✧ the gray buttons should have the same color as the buttons used elsewhere on the user interface

### Fixing and Verifying Notations

Navigating to the symbol and notation definition:

- ☹T first entered the rendering of the symbol into the page search box<sup>22</sup>
- ☹U middle mouse button is OK, once you know it
  - ✧ maybe one could catch the middle click event and handle it like a left click on a normal link (so that the link would not open in a new tab)
- ☹D clear that the link from the rendered symbol leads to the symbol’s definition

Preview of the notation definition:

- ✧ provide labels for the two prototype formats: “OpenMath ” and “MathML ”
- ☹E missed the *declaration* of the rendering (i.e. its XML source code) in the preview

Editing:

- ☹T after saving: “*Now let’s see if it got applied everywhere*” – user wanted to see if the change worked
- ☹U tables are probably confusing if you don’t know XML
- ☹T understood the two-column arrangement of prototype and rendering: “*It follows the design of the ‘Article’ view.*”

---

<sup>21</sup>That would make sense but does not work in the current IkeWiki design (cf. [section 9.4.3](#)).

<sup>22</sup>During this experiment, the rendering was an ASCII character.



☺T looked at the second notation defined for the symbol (for the “constant” role, i.e. the case without arguments), understood it

☺S tried language switching

Complete workflow:

☺U found two-step editing workflow (first navigate to the symbol, then to the notation definition) acceptable

✧ of course, an in-place popup on any rendered symbol would be nice

## Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface:

▷ made a comment about a symbol on the discussion page of the CD – because ...

☺U locally discussing goes via “open this” and is inconvenient

☺T/D one symbol might interact with others in the CD – “*I’d rather report on CD level*”

✧ would be good to see on the CD-level discussion page all discussions about the subitems of the CD

☺T posted an *Issue* and replied to that with a *Position*

✧ suggested ▲/▼ as a fast way for stating one’s position

Argumentation ontology:

☺D “Did you understand the difference between *Argument* and *Position*?” – “*In an Argument, you can give more details; it has more semantics.*”

▷ decisions:

☺C/D confused by the fact that one can only post an untyped reply to a *Decision*

☺D “explained the argumentation ontology” – then understood it

E-mail notification:

☺E “Why is ‘auto-watch’ here and not in the user profile?”

☺E why no direct link to the discussion in the e-mail?

Queries:

▷ “Explained problem-solving assistance idea”

☺U “*In a general wiki, it might work.*”

☺U “*In a mathematical wiki, the right assistance is hard to provide.*”

☺U “*It might be too much inhibiting the user’s thinking; the suggested action, e.g. ‘delete this’, might be too tempting, even with a better user interface.*”

✧ only offer it in certain extreme cases, e.g. an *Issue* with a lot of negative feedback

✧ instead of suggesting an action, merely provide a link to the discussion of the *Issue*

▷ Query result list:

✧ would like a list of “most discussed”, “most active”, “best rated” topics, as in forums

☺E no order of the results

✧ symbols in the result set should be formatted as sym [found in cd], with links to the symbol and CD

✧ there should be more vertical space between results, and more horizontal space between different queries

▷ Writing queries:

☺U “*SPARQL is a bit involved*”

- ☼ would like an interactive display of the RDF graph

### D.3.8 User 8 (2009-12-21)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

- ▷ “open this” links:
  - ☹T rightly found out which of the links does not belong to the symbol but to its mathematical property
  - ☹T had trouble finding the “edit” button
  - ☹T tried to search the content of the page for “edit”

In the document editor:

- ☼ for the value of the description, there should be more space than for name or title (where there is just one line)
- ☹P inserted a line break

In the metadata editing form:

- ☹E “How do you enter a newline?”

Subversion log message:

- ☹N slight mismatch *dc:description* vs. *Description*
- ☹U otherwise OK

#### Fixing and Verifying Notations

Rendered objects:

- ☹D understood what the different (re)presentations of a mathematical object are about
- ☹E “Is it editable?” – “Can I edit a formula’s source code?”
- ☹P the “prefix form” view of an object does not have valid links to symbol definitions
- ☹D “How does it know which notation to choose?”

Navigating to the symbol and notation definition:

- ☹U found operators too small for hitting them by clicking
  - ☼ a “hand” cursor would obviously indicate that the operators are linked
- ☹D “Why is the notation separate?”
  - ☼ display it together with the symbol definition
  - ▷ navigation tree:
    - ☹T/D tried to find the notation via *hasPart*, didn’t find anything
    - ☹C/D looked at “incoming” *hasPart*, confused
    - ☹D expected an outgoing “has symbol” link; expected the notation definition to be a *part* of the symbol definition
    - ☹D “How does the ontology work?” – imagined a class for each operator, then a subclass or instance for each of its renderings or instances – “explained the difference between the system ontology and the CDs being its instances”

Preview of the notation definition:

- ☹N this view is quite different from the view of a rendered object, for no obvious reason
  - ☼ when prototypes are large, maybe show the rendering on top

## Editing:

- ☺C “Am I editing the page of the operator [in general], or of its occurrence in this formula?” – ...
- ☺D ... “the general one” – “That makes sense, because I came from the general symbol definition.”
- ☺N the names of the *mcd*:\* elements are not self-explanatory
- ☺C/☺T “Can I put more than one character for the operator symbol?”<sup>23</sup>

## Editing formulæ:

- ☺D “How can I enter non-ASCII symbols?” – “The formula editor is content-oriented, not presentation oriented. It allows for entering some symbols, such as *arith1#sum*, as Unicode symbol ( $\Sigma$ ) or in ASCII (*sum*).” – “The ASCII variant should be made more obvious”
- ☺S entered  $f(x)$  – recognized as “variable *f* times variable *x*” – “You would probably first have to tell the editor that *f* is a function symbol in some CD.”
  - ✧ there should be a way of entering a new function symbol directly, without first defining it in a CD
- ☺C/☺D user wondered how to use the symbol, whose notation he or she had edited before, in the formula editor.<sup>24</sup>

## Complete workflow:

- ✧ should be possible to fix a rendering locally by clicking

**Peer Review and Preparing Major Revisions by Discussion**

## Discussion forum user interface:

- ✧ “How can I initiate a vote?”
- ✧ first write a post, *then* use buttons [in the post editing dialog] to select its argumentative type
- ✧ “Can you insert formulæ into posts?” – also found it important to intermix text and mathematical objects, e.g.  $f_{\text{name}}$ <sup>25</sup>
- ▷ tree structure:
  - ☺N random post ordering
  - ☺U found tree difficult to browse
    - ✧ should be collapsed when there are many issues
    - ✧ differentiate between open issues vs. closed threads
    - ✧ put *Decisions* on top of the thread they decide on
- ✧ provide ▲/▼ buttons for voting (“as in reddit”); “You do not always elaborate on your vote”

## Argumentation ontology:

- ☺D noticed the difference between *Argument* and *Position*
- ☺T/D allow for multiple decisions<sup>26</sup>

## E-mail notification:

- ☺U OK not to provide the content of the discussion in the e-mail: “You can’t do that in an e-mail-friendly way; people need to go to the site anyway”

## Queries:

---

<sup>23</sup>That is no problem.

<sup>24</sup>The symbol was not part of the official CDs covered by Sentido’s symbol palette.

<sup>25</sup>This would require a presentation markup editor.

<sup>26</sup>This is possible.

- ☺U warning messages about issues are good to tell people that the content is not complete/not trustable
- ☺N “How can users who are not familiar with the argumentation ontology query or search discussions?”
- ✧ offer a query composer as in Protégé
  - ▷ improve the structure of the queries on the main page:
    - ✧ show all unresolved issues, then arrange more specific lists in subtrees: unresolved issues with CDs/symbols
- ☺U the current state is confusing, as some lists are disjoint (discussions about CDs/symbols/other), whereas others are not  $((CDs \cup symbols \cup other) \cap unresolved\ issues \neq \emptyset)$

### D.3.9 User 9 (2010-01-19)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

- ☺C wondered about the buttons that expose alternative (re)presentations of a mathematical object, ...
- ☺T/D ... tried them, understood them
  - ✧ would like to edit in place, e.g. using a context menu
- ☺S went to the “edit” tab while still being on the toplevel page of the CD – “This is not the right place to edit.” – went back to the “Article” tab.
- ☺T used “open this” for the symbol, then again “open this” for its mathematical property – “You overshot the mark a bit.” – “I didn’t realize that I was wrong.”
  - ▷ Navigating back from the mathematical property to the symbol definition:
    - ☺C/T/E “How do I go back [to the page I came from]?” – no breadcrumbs or similar – “There is a facility for that.”
    - ☺T/D used incoming *hasProperty* reference
    - ☺U/D “I didn’t know that I should use the navigation tree, ...”
    - ☺U “... but once I saw it, it was OK”
  - ▷ “edit” toolbox on the left:
    - ☺C first confused by it
    - ☺T/D “Oh, I see it’s something global”
      - ✧ maybe rename it

In the document editor:

- ☺C wondered about the three columns (element name, *@xml:id*, other attributes) in the first row of the table
- ☺S “I don’t see the text I want to edit” (while editing the toplevel of the CD in the first try)
- ☺D “The user has to understand the semantic structure in order to edit!”
  - ✧ system should explain itself and guide the user (tutorial or inline/live guidance)
  - ✧ “A real XML editor would be good, as I’m familiar with XML”
- ☺P “How do I see that I saved?” – experienced the bug that, after pressing “Save” in the editor, SWiM would not jump back into the “Article” view.

In the metadata editing form:

☺E expected an “edit” button

☺T figured out double click himself/herself

Subversion log message:

☺N “*Why does it say ‘actually changed’?*”

✧ it would be helpful to prepend “summary: ...”

☺U replaced metadata field `dc:description` is clear

## Fixing and Verifying Notations

Navigating to the symbol and notation definition:

✧ would like to have a context menu

☺C On the page of the symbol definition of *swimtest#testop*, the user was confused by a mathematical property of *that* symbol, which looked strikingly similar to the one of *swimtest#swim* on which the experiment started and which was at the same screen position<sup>27</sup>

☺T/D went to the “edit” tab – “*I can’t change it!*” – “The notation definition is a separate object.”

☺E “*How would I open ab [invisible times]?*”

✧ The mathematical object should be larger

✧ Show a special cursor (e.g. hand)

✧ highlight the symbol on hover

Preview of the notation definition:

☺U found the preview OK<sup>28</sup>

✧ would like *n*-ary operators to be rendered as  $\text{arg1} \circ \text{arg2} \circ \dots \circ \text{argn}$ , i.e. two actual operators, so that one can see the spacing of the operator symbol

Editing:

✧ Tables should be smaller

☺T “*I can only enter ASCII*” – wants a character map

## Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface:

☺T/U read the tooltip of the “New Issue” button, liked it

▷ “new post” dialog:

☺U “*very nice post popup*”

✧ would be nice to see what type of post I’m writing (i.e. what button I clicked)

▷ icons:

✧ should have tooltips

✧ “thumbs up” icon for a supporting *Argument* should be accompanied by a text label

☺N random post ordering

☺U same user interface for posting *Positions* – “*I don’t mind. Keep the UI simple.*”

Argumentation ontology:

✧ when a decision has already been made, let users still say “I like/don’t like that”

<sup>27</sup>This was an unfortunate condition that I should have avoided when setting up the test pages; in realistic CDs, such similarities are much less likely to occur.

<sup>28</sup>The user was familiar with the syntaxes of OpenMath and Content MathML.

Queries:

- ✧ use a different separator sign in the list of results – “Does the ‘-’ separator have any semantics?”  
– “No, it doesn’t.”
- ☹D “You need to know SPARQL and the ontologies [for writing your own queries].”

### D.3.10 User 10 (2010-01-21)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

- ▷ “open this” links:
  - ☹T knew section editing from Wikipedia, ...
  - ☹N ... so “why is it called ‘open this’?”
  - ☹T “open this” link for the mathematical property “is not the right one”; then opens the right fragment
- ☹T “Aha, up there, there is an ‘Edit’ button, as in Wikipedia ”

In the document editor:

- ☹E “Would I be able to change [the metadata field label] ‘Description’?”
- ☹P didn’t see the text of the symbol’s *discussion* child in the editor.<sup>29</sup>
- ☹N “Much of the content of a page I don’t see in the editor.”

In the metadata editing form:

- ☹T “How do I open a field [for editing]?”
- ☹U “It’s nice that I don’t have to hit ‘save’.”
- ☹P/U overly sensitive to clicks: clicked the “language” drop down list (without editing!), but still a “change” was saved

Subversion log message:

- ☹C In the “actually changed fragments”, “what do those IDs mean?”
  - ✧ choose better names
  - ▷ granularity of commits:
    - ✧ “I do not always want to commit my change instantly.” – “KiWi will support transactions.” – “That would be nice.”
  - ☹U fine-granular commits are good, ...
  - ☹N ... but there will be quite a bunch of them – how to identify which ones are relevant?
    - ✧ “It would be nice to be able to manually add coarse-granular messages [to a set of multiple commits], such as ‘incorporated the results of the last OpenMath meeting’.”

Other comments:

- ☹U multiple interaction patterns, all following a different style ⇒ should be uniform

#### Fixing and Verifying Notations

Navigating to the symbol and notation definition:

- ☹T clicks “open this” for the mathematical property, ...
- ☹T/D goes to “edit” tab.

---

<sup>29</sup>This element should actually have been a fragment of its own but was not accessible in the wiki due to a bug.

- ☹T in the editor, did not notice that the Sentido button was pressed, i.e. that the cursor was in an editable formula
- ☹D “But now I can only edit the semantics, the content markup, so it has to be in a different place.”
- ☹T tried the buttons that expose alternative (re)presentations of a mathematical object, understood them
  - ✧ some action for the right mouse button would be nice
- ☹T/D then noticed the “rendersSymbol” link, and noticed that “the ID of the target is something about ‘notation’”

Preview of the notation definition:

- ☹U didn’t find preview intuitive
  - ✧ would like to see the XML source of the rendering, and possibly *additionally* the rendered preview<sup>30</sup>

Editing:

- ☹T instantly spots the operator symbol
- ☹C otherwise confused by too many tables
  - ✧ “I would best like to directly edit the XML; I’m used to that. – Or a formula-editor like interface that can generate Presentation MathML ”
  - ✧ such an editor could log which palette symbols are used

Complete workflow:

- ☹U workflow with the “symbol definition→notation definition” deviation is too cumbersome
  - ✧ would like to change the notation directly using “Edit”, e.g. in the formula editor
  - ✧ then there should be a warning “Attention, this will have a global effect, not just on *this* mathematical object!”

## Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface:

- ☹T posted an untyped new comment – “How do I get a ‘thumbs up’ now?”
- ☹E “You would have had to post a supporting argument instead.” – “I’d prefer having the buttons for choosing the argumentative type in the ‘new post’ form, because in the editor I know what kind of argument I have made.”
  - ✧ an untyped post should also allow for a typed reply
  - ✧ “Can I delete a comment?”
- ☹U found *Elaboration* icon unclear
- ☹U “The idea is good, but its realization could be improved.”
  - ✧ would like (un)foldable threads for a better overview
- ☹E copy/paste from other posts is impossible, because the “new post” dialog blocks the rest of the GUI

Argumentation ontology:

- ✧ do not initially restrict the possible reply types, but rather see what users want
- ☹D why can’t one reply to an *Idea* with another *Idea*?

<sup>30</sup>The user was familiar with the XML syntax of notation definitions.



☺U “Explained the rationale for restricting reply types” – “OK, on the other hand forcing people to make up their mind is helpful.”

☺D did not understand supporting/challenging initially

E-mail notification:

☺C “What does ‘auto-watch’ mean?”

☺U “It is good that the ‘auto-watch’ option exists, ...”

☺E “but it belongs into the user profile”

☺D need for notification e-mails is unclear

Queries:

☺U liked the automated assistance idea

▷ improve the structure of the queries on the main page:

☺U links to result pages are confusing

✧ instead of page IDs, author and subject should be listed

✧ restrict display to the three latest posts, ...

✧ ... only on request display all posts on a searchable, filterable page

▷ Writing queries:

☺N there should be a more user-friendly access than SPARQL

✧ aggregated pages should be possible (like the virtual documents in TNTBase), e.g. a page of all notation definitions.<sup>31</sup>

### D.3.11 User 11 (2010-01-21)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

☺T when searching for the editor, clicked “open this” link of the mathematical property

☺C/S/☺T “Where am I?” – opened “metadata” tab: no “description” field there; opened “edit” tab: no “description” either. – “This is not what I wanted to edit”

☺T next tried the buttons that expose alternative (re)presentations of a mathematical object

☺U/D those buttons look good, “green means opened”

☺T looked at navigation tree and “edit” toolbar

☺T goes back to the CD, tried “edit” tab there – “Use ‘open this’ for the symbol definition!”

☺U “When you know it, e.g. from the manual, it is OK.”

☺T/D opens the “metadata” tab without further instructions, “because ‘description’ is a metadata.”

In the metadata editing form:

☺S “There is only ‘delete’ there”

☺T ... but then figures out double click himself/herself

☺U nice to have the saving status displayed as a green popup

☺P after editing the metadata, the “Article” perspective was not re-rendered

In the document editor:

☺T everything worked fine

Subversion log message:

---

<sup>31</sup>Porting SWiM to a TNTBase backend will bring such functionality (cf. sections 8.3.4.1 and 9.6.4.4).

- ☺T understood the *swimtest+swim* (i.e. *cd+name*) page name format
- ☺U replaced metadata field – “OK, short and concise”
- ☺U (no comment) – “If I want to know more, I can make a diff anyway.”
- ☺T “The comment (if given) comes from the ‘summary’ field in the editor” – “Ah, there is a ‘summary’ field? I thought that was part of the document!”
- ✧ rename the “summary” field to “change summary”

## Fixing and Verifying Notations

Navigating to the symbol and notation definition:

- ☺E clicked on the symbol in the mathematical object – doesn’t work
- ☺T tried “open this” for the mathematical property
- ☺T tried the buttons for the OpenMath/MathML views
- ☺T/D tried the “edit” tab: does not work, ...
- ☺D as there is only a semantic formula editor
- ☺S “Do a middle click” – went to the symbol definition; confused by the mathematical property similar to the one on the page before (cf. [appendix D.3.9](#) for an explanation)
- ☺T explores the navigation tree, looks at *usesSymbol* – “Keep looking there!”

Preview of the notation definition:

- ☺U refers to the symbol as *swimtest#testop*; inconsistent with the *swimtest+testop* notation used elsewhere
- ☺T/D “I see a preview of the rendering.”

Editing:

- ☺T “I would like to edit the symbol” – opens “edit” tab
- ✧ “In-place editing would be nice, ...”
- ☺U “but, OK, going via the ‘edit’ tab is definitely consistent.”
- ☺T found the operator symbol
- ☺U “The table is hard to read, ...”
- ☺U “... but the [knowledge] structure is complex after all.”
- ☺T back in the “Article” perspective: “Aha, I see a preview of my change!”
- ☺T how to enter a Unicode symbol? – discovered the  $\Omega$  icon in the toolbar<sup>32</sup>

Complete workflow:

- ✧ Would like to click on the symbol. Left click should be reserved for navigating to the symbol’s definition, ...
- ✧ ... but on right click, there should be a popup menu with an “edit notation” entry, even if that was the only entry.
- ✧ “The editor could be arranged more clearly, ...”
- ☺U “... but I understand the complexity.”
- ☺U found the perspective tabs (e.g. “Article”, “Edit”) nice, compared them w.r.t. usability to the menu bar of Mac OS: “I know that the ‘Edit’ button is always in the same place.”

<sup>32</sup>This was the first user who discovered that icon.

## Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface:

- ✧ would like to collapse threads
- ☺T posts an agreement with an existing *Issue* – does not object against having to write a text for each *Position*
- ▷ icons:
  - ☺U “*I can recognize agreement/disagreement from their icons ...*”
  - ☺N “*... but only from the icons.*”
  - ✧ would like to configure that (as you configure a toolbar in an application)
- ✧ “*Remove the parentheses in the tooltips*”
- ☺T/D understood the difference between “New Issue” and “New Comment”

Argumentation ontology:

- ☺D “*What is the difference between a supporting argument and an agreeing position?*”
- ☺D understood what an *Elaboration* is from the tooltip of the reply button

E-mail notification:

- ▷ Subscription:
  - ✧ use more coherent labels: when a user is currently watching a discussion, provide an option “unwatch” instead of a checked checkbox labeled “watch”
  - ☺U good to have the auto-watch option
- ▷ Mail text:
  - ☺U “*Short, concise, then [the user can go] into the system for details.*”
  - ☺N what exact post does the mail refer to? – “*In the forum, I can only identify it by its date.*”<sup>33</sup>
  - ☺E include the subject in the e-mail

Queries:

- ▷ “Explained problem-solving assistance idea”
  - ☺U in principle this is good, ...
  - ☺U ... but it will be hard to find post types on which consensus might be obtained
  - ☺U “The community will be able to edit and thus adapt the argumentation ontology.” – “*That is good.*”
- ▷ improve the structure of the queries on the main page:
  - ☺U “*The idea is good – this is information I want to get.*”
  - ☺U “*The presentation is bad*” – i.e. that the URIs of the pages appear as results
    - ✧ do it as in some feed aggregators: subject, author and date of recent posts
    - ✧ for each such preview item, provide a “read more” link.
- ▷ Writing queries:
  - ☺U “*Cool!*”
  - ☺C/☺T does SPARQL support *LIMIT*?<sup>34</sup>
  - ☺U/D “*Everybody can change/adapt the presentation of the results.*”

---

<sup>33</sup>Additionally, one can identify a post by its author, whose name is also included in the notification e-mail.

<sup>34</sup>Yes, it does.

### D.3.12 User 12 (2010-01-26)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

▷ “open this” links:

☹T first tried the “Edit” tab, but then discovered “open this” link

☹T “*This is like in Wikipedia.*”

☹T “*How do you get back [from a symbol to the CD]?*”

In the document editor:

☹T used the summary, then saved

In the metadata editing form:

☹C/☹T “*Where can I edit?*” – then tried double clicking

☹T also found out himself/herself that one can save by pressing “Return”

Subversion log message:

☹S “*Why does it say ‘fragment’?*”

✧ Rather say “symbol” or “example”, depending on the type of fragment

☹U found metadata change log OK

Other comments:

☹N “*There is a lot of information in the surrounding toolbars, ...*”

☹U “*... but otherwise one gets along.*”

☹U “*With a little technical background, it is all at a normal level of difficulty*”

#### Fixing and Verifying Notations

Navigating to the symbol and notation definition:

✧ would like a tooltip with the name of the symbol

☹S confused by the middle click opening a new tab

☹T/D “*Where am I? – Aha, in the definition of the symbol.*”

☹T tried the “Edit” tab but doesn’t find anything helpful there

☹T “*Where is the link to the notation definition?*”

☹D “*‘outgoing’... – oh, well, it could as well be ‘incoming’*”

☹U “*hasProperty is too technical a name*”

☹U “*rendersSymbol – sounds logical, ...*”

☹C/D “*... but why in that direction?*”

Preview of the notation definition:

☹U looks OK

☹E “*Where is the rendering described? – That [i.e. some view on the rendering element itself] is missing.*”

Editing:

☹T “*You don’t have to understand the [table] structure.*”

☹T changes the symbol, gives a summary

☹U “*How did you find the symbol?*” – “*There was such a chaos; I just changed it on spec.*”

▷ Special characters:

☹T “*Where do you enter special characters? Aha, there is the  $\Omega$  symbol.*”

✧ “Can you also enter &bullet;?”

Complete workflow:

- ☺U “When you know that the symbol links are always listed under ‘references’<sup>35</sup>, and when you know the schema – what is incoming, what is outgoing –, then it is OK”
- ☺T/D “The ‘references’ box is your main tool”
  - ✧ it could be larger, or colored, as it is important.
  - ▷ Difficulties pointed out:
    - ☺U going on from the occurrence of the symbol in a mathematical object
    - ☺U/T going back to the original object<sup>36</sup>
      - ✧ idea: link to the “referrer”, e.g. by breadcrumbs
- ☺U changing the rendering is at a normal level of difficulty (as in an XML editor)

### Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface/Argumentation ontology:

- ☺N each entry uses a lot of space
- ☺U “looks provisional”
- ☺C what do the icons mean?
  - ✧ provide a legend for the icons
- ☺D “‘Agree’ – is that my opinion?”
- ☺U “complex – you can do a lot”
- ☺U “After a bit more of feeling around, it makes sense.”
- ☺D “‘Decision’ – and then the answers disappear?”
  - ✧ provide +/- buttons for (un)folding threads: nobody will probably use them manually, ...
  - ✧ ... but there should be a reasonable pre-folding
  - ▷ Positions:
    - ☺D “‘Position’ is a rating – ...”
    - ☺E “do you see them anywhere?”
      - ✧ directly display ratings
      - ✧ or, e.g., as on YouTube: content rated negatively is initially invisible
- ☺U about the reply possibilities: “convention over configuration”
- ☺U “First impression: ‘quite a lot’”
  - ✧ give posts of different types more distinction – not just by icon, but also by color

E-mail notification:

- ☺C did not understand what “auto-watch” means: “the whole discussion, or just replies to me?”
  - ▷ Mail text:
    - ☺U “Something has happened [i.e. the e-mail does not tell its receiver much more than that] – OK”
    - ☺N “What if it is a large discussion?”
      - ✧ include subject
      - ✧ could also include content of the post

Queries:

---

<sup>35</sup>I.e. the navigation tree

<sup>36</sup>The user did not notice that the original object is still open in the original tab.

- ☹N “*The presentation looks like text – no optical guides*”
  - ✧ show “discussions that are expiring now” – those with unsolved problems are probably relevant
- ☹N “*Am I interested in that?*”
- ☹N “*What if the list overflows?*”
  - ✧ when logged in, show or prefer those discussions in which *I* am participating
  - ▷ Writing queries:
    - ☹U “*You can draw attention to topics.*”
      - ✧ “*A bird’s eye view – how else can you get that?*”

## General Feedback

- ☹D “*I consider it [the design decisions behind the user interface] reproducible*”
- ☹U with appropriate explanations it is OK
- ☹U “*But if everything will be accessible in a ‘blind flight’?*”
- ☹U found “socialise” links unnecessary
  - ✧ move them to the bottom of the left column

### D.3.13 User 13 (2010-02-04)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

- ☹E expected “*edit this*” instead of “*open this*” (as known from other wikis)
- ☹T clicked the right “*open this*” link
- ☹T from the symbol page: “*How do you get back to the whole CD?*” – expected a navigation tree
- ☹T “*There is a navigation tree, i.e. the navigation tree.*” – “*I didn’t notice that.*”
- ☹T/D looked at some outgoing links, then at some incoming links; had not noticed the “*outgoing*”/“*incoming*” headings
- ☹E expected outgoing *isPartOf*

In the document editor:

- ☹T edited and saved successfully

In the metadata editing form:

- ☹E expected an “*edit*” button
- ☹T clicked into another field in order to save

Subversion log message:

- ☹U replaced metadata field – OK
- ☹U actually changed fragment – OK
- ☹E “*It could also tell me that something was changed from ‘old value’ to ‘new value’ – but you can also see that from the diff.*”
- ☹C “*Why no comment?*”
- ☹T/D “*You can give a summary in the ‘edit’ tab*” – “*I didn’t notice it. And I wasn’t aware that the summary goes to a database.*”

## Fixing and Verifying Notations

Navigating to the symbol and notation definition:

- ☺T/D “How do I identify the CD the symbol comes from?” – tried the OpenMath view of the mathematical object
  - ✧ would like to click on a symbol and see its CD or name in a popup
  - ✧ ... or to have a menu item “go to notation”
    - ▷ Clicking on symbols:
      - ☺E “How would I navigate to  $\frac{a}{b}$  or  $ab$  [invisible times]?”
      - ☺E “Can I click on variables?”
- ☺T/D after navigating to the symbol definition with the middle mouse button: “So this is the definition of the operator.”
- ☺T “How do you go to the notation now?”
- ☺T again tried to view the alternative (re)presentations of the FMP that the current symbol had,  
...
- ☺T ... then noticed the *rendersSymbol* link in the navigation tree
- ☺T/D after editing (while on the page with the notation definition): “How do I navigate back?” – used the link referring to the symbol (here: *swimtest#testop*, then went from the symbol to the CD via the navigation tree

Preview of the notation definition:

- ☺C/D “What are the OpenMath and Content MathML [prototype] parts about; what is the difference?”
- ☺C/D user did not realize what the XML does, did not realize that it is about pattern matching
  - ☺C “How do I know that the symbol is rendered as  $\otimes^{37}$ ? – Where does it come from?”
    - ✧ additionally show the XML code of the rendering

Editing:

- ☺D “The left part reminds me of what I have seen before.”
- ☺S/☺D “The right part I have not seen before – I guess it is the rendering.”
  - ☺T discovered the  $\Omega$  button; chose a Unicode symbol
  - ☺T also tried to open the formula editor<sup>38</sup>
  - ☺P rendering preview was empty after saving

Complete workflow:

- ☺U identifying the symbol and finding the place where the notation of the symbol is defined are much easier now [compared to the traditional file system/text editor alternative that was mentioned in the description of the experiment]
- ☺U “I couldn’t understand the pattern.”
- ☺U editing the notation got much easier
- ☺U checking whether the symbol is rendered correctly in its original occurrence after editing now got much easier (thanks to the preview)

---

<sup>37</sup>That was its rendering at the time of the experiment.

<sup>38</sup>Using the [content markup] formula editor should actually be forbidden when editing renderings.

## Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface:

- ☺D understood the basic argumentative structure
- ☺C/☺D “What does the label ‘Justification’ [on a post] with ‘thumbs down’ mean? – Aha!”
- ☺U “The icons are helpful; they give a quick overview without reading a lot”
- ☺C/☺D/U first wondered about missing icons on some posts, but then found the idea of untyped posts good
- ☺E “I’m more used to linear forums where I can just write.”
- ☺D/U “Here, I have to think first (which is good), e.g. ‘am I for or against it?’”
- ☺C/E “What is a ‘supporting evaluation?’” – no tooltips in the popup menus of the *Argument* button

E-mail notification:

- ☺U good that the feature is there
- ☺U “The mails have exactly the right content.”

Queries:

- ▷ “Explained problem-solving assistance idea”
  - ☺U found the general idea good
  - ☺U “Refining Idea types seems too restrictive, or in any case quite challenging to get right.”
  - ☺U “Assistance is good, but maybe hard to implement.”
- ▷ Query result lists:
  - ☺U generally helpful, e.g. for the administrator
  - ☺U/D “If I’m responsible e.g. for the arith1 CD, I see what’s going on”
  - ☺D “So it’s like an advertisement or motivation.”
- ▷ Writing queries:
  - ☺U “You can use *foaf:knows* with user profiles” – “That’s really cool!”

### D.3.14 User 14 (2010-02-04)

#### Quickly Fixing Minor Errors

Navigating to the document editor:

- ☺E expected table of contents in “Article” perspective
  - ▷ “open this” links:
    - ☺T “As it says ‘open this’ here, I rather tend to clicking on ‘Edit’.”
    - ☺E expected “edit this”, as in other wikis
    - ☺T still, did “open this”, then “Edit”
- ☺D looking at the navigation tree: “I see all kinds of dependencies here.”
- ☺C confused by the toolbars for ontology editing

In the metadata editing form:

- ☺E misses “edit” button
- ☺T discovered double click for editing and “Return” for saving
- ☺T “At the bottom, it says ‘saved’.”
- ☺P after editing the metadata, the “Article” perspective was not re-rendered

In the document editor:



- ☺T clicked the Sentido button
- ☺N “Did you see that it is already pressed?” – “No, I didn’t notice that.”
- ☺T confused by the pressed button; thought that one can no longer click it.

Subversion log message:

- ☺C/D “What is the difference between ‘field’ and ‘fragment’ – *aha, the fragment provides context.*”<sup>39</sup>
- ☺C Why no comment?
- ☺C confused by the relation between the two lines replaced metadata field and *actually* changed – “*The first line says something, whereas the second line says ‘no, actually something else was changed’; this is misleading*”
- ☼ use changed field F in CD+name

### Fixing and Verifying Notations

Navigating to the symbol and notation definition:

- ☺T first tries “open this” for the mathematical property
- ☺T uses the outgoing *usesSymbol* links from the navigation tree
- ☺T identified the symbol in question from the list of linked symbols – “Respect!” – “*But that works when I understand the formula, i.e. then I can associate rendered symbols to ‘usesSymbol CD+name’.*”
- ☺T on the page of the symbol, found the incoming *rendersSymbol* link intuitive

Preview of the notation definition:

- ☺U OK

Editing:

- ☺N “Uuuuaah...”
- ☺U “Aha, in the part on the right I have to change something, that is the relevant part.”
- ☺T gave a summary (after a hint)
- ☺T/D navigates back using the navigation tree with superior ease<sup>40</sup>

### Peer Review and Preparing Major Revisions by Discussion

Discussion forum user interface:

- ☺T studies a long existing thread
- ☺T reads the tooltips of the existing reply buttons
- ☺T “As I am an admin, may I post a Decision?”
- ☺D “Why do certain posts have fewer reply buttons?”
- ☺D did not understand the difference between a challenging and a supporting argument from the user interface
  - ▷ icons:
    - ☺T intuitively understood what a “Justification” label with “thumbs down” icon means on a post
    - ☺U “You quickly get used to the icons.”

---

<sup>39</sup>This is a misunderstanding of the log message generated on a metadata field change.

<sup>40</sup>The user had previous knowledge about representing dependencies in [mathematical] documents and therefore did not miss certain inverse relations and was able to deal with incoming/outgoing links.

- ☹N random post ordering
- ☹N “What if I want to change something [i.e. a post] later on? As a user or as an admin?”
  - ⚙ allow an admin to turn e.g. an untyped reply into what it actually is, e.g. an elaboration
- ☹E picks a *Decision* post, intends to say “I don’t like this that much.”
- ☹U “*The community will adapt its discussion culture to the prescribed structure.*”
- ☹U “*The discussion threads are already pre-structured. This is better than in Wikipedia.*”

E-mail notification:

- ☹E found that “auto-watch” rather belongs on a preference page
- ☹U found mail text OK: “*When I want to jump in, I have to go into the system anyway.*”
  - ⚙ filter notifications by argumentative type: “*I would only like arguments, no positions. I would like to know when a decision has been taken.*”

Queries:

- ▷ “Explained problem-solving assistance idea”
  - ☹D “*Specific Issue types restrict me in that I cannot take a Position on them.*”<sup>41</sup>
  - ☹U considered such a feature helpful for Wikipedia
- ☹U considered the querying support generally helpful<sup>42</sup>
  - ▷ “*I would like to watch a whole CD with a query tailored to that CD*”:
    - ⚙ prepare a query template for “a given CD or a part of it is being discussed”
    - ⚙ allow users to copy that template to their user page and adapt it.

---

<sup>41</sup>No, that was a misunderstanding.

<sup>42</sup>The test person moderates a MediaWiki-driven site.



# Bibliography

In citations of online resources, such as standard specifications, wiki articles, or blog posts, permanent links to the exact version are used where possible.

- [AA05] David Aumüller and Sören Auer. “Towards a Semantic Wiki Experience – Desktop Integration and Interactivity in WikSAR.” In: *Proc. of 1st Workshop on The Semantic Desktop – Next Generation Personal Information Management and Collaboration Infrastructure*. (Galway, Ireland, Nov. 2005). 2005. URL: [http://www.semanticdesktop.org/SemanticDesktopWS2005/final/22\\_aumueller\\_semanticwikiexperience\\_final.pdf](http://www.semanticdesktop.org/SemanticDesktopWS2005/final/22_aumueller_semanticwikiexperience_final.pdf) (cit. on p. 21).
- [AAH+10] Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, eds. *The Semantic Web: Research and Applications (Part I)*. 7th Extended Semantic Web Conference (ESWC) (Heraklion, Crete, Greece, May 30–June 3, 2010). Lecture Notes in Computer Science 6088. Springer Verlag, 2010 (cit. on pp. 474, 490).
- [AAL+08] Hal Abelson, Ben Adida, Mike Linksvayer, and Nathan Yergler. *ccREL: The Creative Commons Rights Expression Language*. Tech. rep. Creative Commons, Mar. 3, 2008. URL: <http://wiki.creativecommons.org/images/d/d6/Ccrel-1.0.pdf> (visited on 2009-10-22) (cit. on pp. 47, 124, 168).
- [ABC+03] Ron Ausbrooks, Stephen Buswell, David Carlisle, Stéphane Dalmas, Stan Devitt, Angel Diaz, Max Froumentin, Roger Hunter, Patrick Ion, Michael Kohlhase, Robert Miner, Nico Poppelier, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. *Mathematical Markup Language (MathML) Version 2.0 (second edition)*. W3C Recommendation. World Wide Web Consortium (W3C), Oct. 21, 2003. URL: <http://www.w3.org/TR/MathML2> (cit. on p. 382).
- [ABC+08] Ron Ausbrooks, Bert Bos, Olga Caprotti, David Carlisle, Giorgi Chavchanidze, Ananth Coorg, Stéphane Dalmas, Stan Devitt, Sam Dooley, Margaret Hinchcliffe, Patrick Ion, Michael Kohlhase, Azzeddine Lazrek, Dennis Leas, Paul Libbrecht, Manolis Mavrikis, Bruce Miller, Robert Miner, Murray Sargent, Kyle Siegrist, Neil Soiffer, Stephen Watt, and Mohamed Zergaoui. *Mathematical Markup Language (MathML) Version 3.0*. W3C Working Draft. World Wide Web Consortium (W3C),

- Apr. 9, 2008. URL: <http://www.w3.org/TR/2010/REC-MathML3-20101021> (cit. on p. 288).
- [ABC+10] Ron Ausbrooks, Stephen Buswell, David Carlisle, Giorgi Chavchanidze, Stéphane Dalmas, Stan Devitt, Angel Diaz, Sam Dooley, Roger Hunter, Patrick Ion, Michael Kohlhase, Azzeddine Lazrek, Paul Libbrecht, Bruce Miller, Robert Miner, Murray Sargent, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. *Mathematical Markup Language (MathML) Version 3.0*. W3C Recommendation. World Wide Web Consortium (W3C), 2010. URL: <http://www.w3.org/TR/MathML3> (cit. on pp. 63–65, 72, 139, 210, 213–219, 321, 330, 382).
- [ABD03] Andrea Asperti, Bruno Buchberger, and James Harold Davenport, eds. *Mathematical Knowledge Management, MKM’03*. LNCS 2594. Springer Verlag, 2003 (cit. on pp. 472, 485).
- [ABM+08] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. *RDFa in XHTML: Syntax and Processing*. W3C Recommendation. World Wide Web Consortium (W3C), Oct. 2008. URL: <http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014/> (cit. on p. 58).
- [ABM+10] Ben Adida, Mark Birbeck, Shane McCarron, and Ivan Herman. *RDFa Core 1.1. Syntax and processing rules for embedding RDF through attributes*. W3C Working Draft. World Wide Web Consortium (W3C), Oct. 26, 2010. URL: <http://www.w3.org/TR/2010/WD-rdfa-core-20101026/> (cit. on p. 169).
- [ABM+11] Ben Adida, Mark Birbeck, Shane McCarron, and Ivan Herman. *RDFa Core 1.1. Syntax and processing rules for embedding RDF through attributes*. W3C Working Draft. World Wide Web Consortium (W3C), Mar. 31, 2011. URL: <http://www.w3.org/TR/2011/WD-rdfa-core-20110331/> (cit. on pp. 56, 58, 59, 89, 165).
- [Abr] Per Abrahamsen. *vi Tutorial*. URL: <http://www.dina.kvl.dk/~abraham/religion/vi-tutorial.html> (visited on 2010-08-18) (cit. on p. 418).
- [ACD+10] Serge Autexier, Jacques Calmet, David Delahaye, Patrick D. F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton, eds. *Intelligent Computer Mathematics*. LNAI 6167. Springer Verlag, 2010. ISBN: 3642141277 (cit. on pp. 460, 462, 476–478, 480, 483, 501).
- [Acm] *The 1998 ACM Computing Classification System*. 1998. URL: <http://www.acm.org/about/class/ccs98> (visited on 2009-11-18) (cit. on p. 46).
- [ACN+07] Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, eds. *The Semantic Web*. 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007 (Busan, Korea, Nov. 11–15, 2007). Lecture Notes in Computer Science 4825. Springer Verlag, 2007. ISBN: 978-3-540-76297-3 (cit. on pp. 471, 499).

- [ACR+08] Serge Autexier, John Campbell, Julio Rubio, Volker Sorge, Masakazu Suzuki, and Freek Wiedijk, eds. *Intelligent Computer Mathematics*. 9th International Conference, AISC, 15th Symposium, Calculemus, 7th International Conference MKM (Birmingham, UK, July 28–Aug. 1, 2008). LNAI 5144. Springer Verlag, 2008 (cit. on pp. 466, 478, 496).
- [Act] ACTIVEMATH. URL: <http://www.activemath.org> (visited on 2010-06-05) (cit. on pp. 15, 41, 65, 184, 240).
- [Ada09] Nico Adams. *Semantic Chemistry*. Jan. 14, 2009. URL: [http://semanticweb.com/semantic-chemistry\\_b10684](http://semanticweb.com/semantic-chemistry_b10684) (visited on 2010-11-26) (cit. on p. 351).
- [Ado] Adobe – XMP Developer Center. Adobe. URL: <http://www.adobe.com/devnet/xmp/> (visited on 2010-08-25) (cit. on p. 223).
- [AEBo7] Miguel A. Abánades, Jesús Escribano, and Francisco Botana. “First Steps on Using OpenMath to Add Proving Capabilities to Standard Dynamic Geometry Systems.” In: [KKM+07], pp. 131–145. ISBN: 978-3-540-73083-5 (cit. on pp. 65, 310).
- [AFN+07] Serge Autexier, Armin Fiedler, Thomas Neumann, and Marc Wagner. “Supporting User-Defined Notations When Integrating Scientific Text-Editors with Proof Assistance Systems.” In: [KKM+07], pp. 176–190. ISBN: 978-3-540-73083-5 (cit. on pp. 25, 74, 195, 221, 301).
- [AGC+06] Andrea Asperti, Ferruccio Guidi, Claudio Sacerdoti Coen, Enrico Tassi, and Stefano Zacchiroli. “A Content Based Mathematical Search Engine: Whelp.” In: *Types for Proofs and Programs, International Workshop, TYPES 2004, revised selected papers*. Ed. by Jean-Christophe Filliâtre, Christine Paulin-Mohring, and Benjamin Werner. Lecture Notes in Computer Science 3839. Springer Verlag, 2006, pp. 17–32 (cit. on p. 17).
- [AGNo9] Andrea Asperti, Herman Geuvers, and Raja Natarajan. “Social Processes, Program Verification and all that.” In: *Mathematical Structures in Computer Science* 19.5 (Oct. 2009), pp. 877–896 (cit. on pp. 4, 5, 14, 18, 22).
- [AHM+06] Serge Autexier, Dieter Hutter, Till Mossakowski, and Axel Schairer. “Maya: Maintaining Structured Documents.” In: [Koh06b]. Chap. 26.12. URL: <http://omdoc.org/pubs/omdoc1.2.pdf> (cit. on p. 39).
- [AKK+08] Waseem Akhtar, Jacek Kopecký, Thomas Krennwallner, and Axel Polleres. “XSPARQL: Traveling between the XML and RDF worlds – and avoiding the XSLT pilgrimage.” In: [BHH+08] (cit. on pp. 60, 95, 270).
- [AKT+08] Anupriya Ankolekar, Markus Krötzsch, Thanh Tran, and Denny Vrandečić. “The two cultures: Mashing up Web 2.0 and the Semantic Web.” In: *Web Semantics* 6.1 (2008), pp. 70–75 (cit. on pp. 8–10, 340).
- [Alfo7] Ron Alford. *PROPOSAL: Deprecate membershipClass, add memberOf*. e-mail to [foaf-dev@lists.foaf-project.org](mailto:foaf-dev@lists.foaf-project.org). May 25, 2007. URL: <http://lists.foaf-project.org/pipermail/foaf-dev/2007-May/008551.html> (cit. on p. 148).

- [AM10] Serge Autexier and Normen Müller. “Semantics-based Change Impact Analysis for Heterogeneous Collections of Documents.” In: *Proceedings of the 10th ACM symposium on Document engineering*. Ed. by Michael Gormish and Rolf Ingold. DocEng ’10. Manchester, United Kingdom: ACM, 2010, pp. 97–106. ISBN: 978-1-4503-0231-9. DOI: <http://doi.acm.org/10.1145/1860559.1860580>. URL: <http://doi.acm.org/10.1145/1860559.1860580> (cit. on pp. 184, 308).
- [Anc09] Ștefan Anca. “Natural Language and Mathematics Processing for Applicable Theorem Search.” MA thesis. Jacobs University Bremen, 2009. URL: <https://svn.eecs.jacobs-university.de/svn/eecs/archive/msc-2009/aanca.pdf> (cit. on pp. 224, 353).
- [And] Eric Andrès. *LaTeX2OQMath*. URL: <http://www.activemath.org/~eandres/120.php> (visited on 2010-09-09) (cit. on p. 194).
- [APSC+03] Andrea Asperti, Luca Padovani, Claudio Sacerdoti Coen, Ferruccio Guidi, and Irene Schena. “Mathematical Knowledge Management in HELM.” In: *Annals of Mathematics and Artificial Intelligence, Special Issue on Mathematical Knowledge Management, Kluwer Academic Publishers* 38.1–3 (May 2003), pp. 27–46 (cit. on p. 16).
- [Arq] ARQ – A SPARQL Processor for Jena. URL: <http://jena.sourceforge.net/ARQ/> (visited on 2010-09-12) (cit. on pp. 228, 386, 396).
- [Arx] *arxiv.org e-Print archive*. URL: <http://www.arxiv.org> (visited on 2010-01-08) (cit. on pp. 11, 261, 352, 356).
- [ASCT+07] Andrea Asperti, Claudio Sacerdoti Coen, Enrico Tassi, and Stefano Zacchiroli. “User Interaction with the Matita Proof Assistant.” In: *Journal of Automated Reasoning* 39.2 (2007), pp. 109–139 (cit. on p. 18).
- [ASH+06] Anupriya Ankolekar, Katia Sycara, James Herbsleb, Robert Kraut, and Chris Welty. “Supporting Online Problem-Solving Communities with the Semantic Web.” In: *[Www]*, pp. 575–584 (cit. on p. 231).
- [ATC+09] Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl, eds. *The Semantic Web: Research and Applications*. 6th European Semantic Web Conference (ESWC) (Hersonissos, Crete, Greece, May 31–June 4, 2009). Lecture Notes in Computer Science 5554. Springer Verlag, 2009 (cit. on p. 473).
- [AYBB+10] Sihem Amer-Yahia, Chavdar Botev, Stephen Buxton, Pat Case, Jochen Doerre, Michael Dyck, Mary Holstege, Jim Melton, Michael Rys, and Jayavel Shanmugasundaram. *XQuery and XPath Full Text 1.0*. W3C Candidate Recommendation. World Wide Web Consortium (W3C), Jan. 28, 2010. URL: <http://www.w3.org/TR/2010/CR-xpath-full-text-10-20100128> (cit. on p. 225).

- [BAB+10] Sean Bechhofer, John Ainsworth, Jiten Bhagat, Iain Buchan, Philip Couch, Don Cruickshank, David De Roure, Mark Delderfield, Ian Dunlop, Matthew Gamble, Carole Goble, Danus Michaelides, Paolo Missier, Stuart Owen, David Newman, and Shoaib Sufi. “Why Linked Data is Not Enough for Scientists.” In: [Esc] (cit. on pp. 20, 354).
- [Bae10] John Baez. “Math Blogs.” In: *Notices of the AMS* (Mar. 2010), p. 333. URL: <http://www.ams.org/notices/201003/rtx100300333p.pdf> (cit. on pp. 12, 13, 19).
- [BAG09] Chris Bizer, Sören Auer, and Gunnar Aastrand Grimnes, eds. *Scripting and Development for the Semantic Web (SFSW)*. CEUR Workshop Proceedings 449. Aachen, May 2009. URL: <http://CEUR-WS.org/Vol-449/> (cit. on pp. 474, 482).
- [Bano6] Grzegorz Bancerek. “Information Retrieval and Rendering with MML Query.” In: [BFo6], pp. 266–279 (cit. on pp. 15, 230).
- [Bar] Luc Barthelet. *Mathematica-users Wiki Homepage*. URL: <http://www.mathematica-users.org> (visited on 2010-10-07) (cit. on p. 303).
- [Baro6] Rebhi S. Baraka. “A Framework for Publishing and Discovering Mathematical Web Services.” PhD thesis. Johannes Kepler Universität Linz, 2006. URL: [http://www.risc.uni-linz.ac.at/publications/download/risc\\_2945/06-04.pdf](http://www.risc.uni-linz.ac.at/publications/download/risc_2945/06-04.pdf) (cit. on p. 18).
- [Bar10] Michael J. Barany. “[B]ut this is blog maths and we’re free to make up conventions as we go along’: Polymath1 and the modalities of ‘massively collaborative mathematics’.” In: *Proceedings of the 6th International Symposium on Wikis and Open Collaboration (WikiSym)*. (Gdansk, Poland, July 7–9, 2010). Ed. by Phoebe Ayers and Felipe Ortega. ACM Press, 2010. URL: <http://www.wikisym.org/ws2010/Proceedings/> (cit. on pp. 10, 13).
- [Bato6] Steve Battle. “Gloze: XML to RDF and back again.” In: *Jena User Conference*. May 2006. URL: <http://jena.hpl.hp.com/juc2006/proceedings.html> (cit. on p. 140).
- [BB07] Uldis Bojārs and John G. Breslin. *SIOC Core Ontology Specification*. W3C Member Submission. World Wide Web Consortium (W3C), June 2007. URL: <http://www.w3.org/Submission/2007/SUBM-sioc-spec-20070612/> (cit. on pp. 47, 124, 128).
- [BB08] Dave Beckett and Jeen Broekstra. *SPARQL Query Results XML Format*. W3C Recommendation. World Wide Web Consortium (W3C), Jan. 15, 2008. URL: <http://www.w3.org/TR/2008/REC-rdf-sparql-XMLres-20080115/> (cit. on p. 392).
- [BB09] Lou Burnard and Syd Bauman. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Tech. rep. Version 1.5.0. TEI Consortium, Nov. 8, 2009. URL: <http://www.tei-c.org/release/doc/tei-p5-doc/en/html/> (cit. on pp. 77, 78, 80, 92).
- [BBC+07] Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernández, Michael Kay, Jonathan Robie, and Jérôme Siméon. *XML Path Language (XPath) 2.0*. W3C Recommendation. World Wide Web Consortium (W3C), Jan. 23, 2007. URL: <http://www.w3.org/TR/2007/REC-xpath20-20070123/> (cit. on p. 55).



- [BBD+10] Diego Berrueta, Dan Brickley, Stefan Decker, Sergio Fernández, Christoph Görn, Andreas Harth, Tom Heath, Kingsley Idehen, Kjetil Kjernsmo, Alistair Miles, Alexandre Passant, Axel Polleres, and Luis Polo. *SIOC Core Ontology Specification*. Ed. by Uldis Bojārs and John G. Breslin. Version 1.35. Mar. 25, 2010. URL: <http://rdfs.org/sioc/spec/> (visited on 2010-05-02) (cit. on pp. 47, 124, 127–129, 131).
- [BBG+08] Chris Bizer, Mark Butler, Stephen Garland, David Huynh, David Karger, Ryan Lee, Stefano Mazzocchi, Emmanuel Pietriga, Dennis Quan, and Karun Bakshi. *Fresnel – Display Vocabulary for RDF*. Tech. rep. World Wide Web Consortium (W3C), Apr. 2, 2008. URL: <http://www.w3.org/2005/04/fresnel-info/> (cit. on p. 212).
- [BBL08] David Beckett and Tim Berners-Lee. *Turtle – Terse RDF Triple Language*. W3C Team Submission. World Wide Web Consortium (W3C), Jan. 4, 2008. URL: <http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/> (cit. on p. 56).
- [BBP+08] Uldis Bojārs, John G. Breslin, Vassilios Peristeras, Giovanni Tummarello, and Stefan Decker. “Interlinking the Social Web with Semantics.” In: *IEEE Intelligent Systems* 23.3 (May/June 2008) (cit. on pp. 47, 124, 128).
- [BCC+04] Stephen Buswell, Olga Caprotti, David P. Carlisle, Michael C. Dewar, Marc Gaëtano, and Michael Kohlhase. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20> (cit. on pp. 65, 67, 114, 115, 143, 179, 210, 372).
- [BCF+07] Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, and Jérôme Siméon. *XQuery 1.0: An XML Query Language*. W3C Recommendation. World Wide Web Consortium (W3C), Jan. 23, 2007. URL: <http://www.w3.org/TR/2007/REC-xquery-20070123> (cit. on p. 55).
- [BCH+10] Dan Brickley, Vinay K. Chaudhri, Harry Halpin, and Deborah L. McGuinness, eds. *Proceedings of the AAAI Spring Symposium on Linked Data Meets Artificial Intelligence*. 2010 (cit. on pp. 464, 475).
- [BCH07] Chris Bizer, Richard Cyganiak, and Tom Heath. *How to Publish Linked Data on the Web*. July 27, 2007. URL: <http://sites.wiwiiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/20070727/> (cit. on p. 210).
- [BD78] P. Bateman and H. Diamond. “John E. Littlewood (1885–1977). An Informal Obituary.” In: *Mathematical Intelligencer* 1.1 (1978), pp. 28–33 (cit. on p. 4).
- [BDH+09] Jie Bao, Li Ding, Rui Huang, Paul Smart, Dave Braines, and Gareth Jones. “A Semantic Wiki based Light-Weight Web Application Model.” In: *Proceedings of the 4th Asian Semantic Web Conference*. 2009, pp. 168–183 (cit. on p. 282).
- [BDM09] Jie Bao, Li Ding, and Deborah L. McGuinness. “Semantic History: Towards Modeling and Publishing Changes of Online Semantic Data.” In: *Social Data on the Web (SDoW), Workshop at the 8th International Semantic Web Conference*. (Washington DC, USA, Oct. 25, 2009). Ed. by John Breslin, Uldis Bojārs, Alexandre Passant, and Sergio Fernández. CEUR Workshop Proceedings 520. Aachen, 2009. URL: <http://ceur-ws.org/Vol-520/paper07.pdf> (cit. on p. 328).

- [Beco4a] Dave Beckett. “Modernising Semantic Web Markup.” In: *XML Europe*. (Amsterdam, The Netherlands, Apr. 18–21, 2004). 2004. URL: <http://www.dajobe.org/papers/xml europe2004/> (cit. on pp. 56, 391).
- [Beco4b] Dave Beckett. *RDF/XML Syntax Specification (Revised)*. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 10, 2004. URL: <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/> (cit. on pp. 56, 149).
- [BEK+09] François Bry, Michael Eckert, Jakub Kotowski, and Klara Weiand. “What the User Interacts with: Reflections on Conceptual Models for Semantic Wikis.” In: [LSSM+09]. URL: <http://ceur-ws.org/Vol-464/> (cit. on pp. 283, 310).
- [Bel09] John L. Bell. “The Axiom of Choice.” In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2009. 2009. URL: <http://plato.stanford.edu/archives/spr2009/entries/axiom-choice/> (cit. on p. 38).
- [Ber] *Berkeley DB XML*. URL: <http://www.oracle.com/database/berkeley-db/xml/> (visited on 2009-10-22) (cit. on p. 225).
- [BFo6] Jon Borwein and William M. Farmer, eds. *Mathematical Knowledge Management (MKM)*. LNAI 4108. Springer Verlag, 2006 (cit. on pp. 455, 473, 494).
- [BFM+10] Anders Berglund, Mary Fernández, Ashok Malhotra, Jonathan Marsh, Marton Nagy, and Norman Walsh. *XQuery 1.0 and XPath 2.0 Data Model (XDM) (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C), Dec. 14, 2010. URL: <http://www.w3.org/TR/2010/REC-xpath-datamodel-20101214> (cit. on p. 215).
- [BG02] Dan Brickley and Ramanathan V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Working Draft. World Wide Web Consortium (W3C), Apr. 30, 2002. URL: <http://www.w3.org/TR/2002/WD-rdf-schema-20020430/> (cit. on p. 85).
- [BG04] Dan Brickley and Ramanathan V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 10, 2004. URL: <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/> (cit. on pp. 61, 148, 199, 203, 208).
- [BGE+08] Michel Buffa, Fabien Gandon, Guillaume Erete, Peter Sander, and Catherine Faron. “SweetWiki: A semantic wiki.” In: *Web Semantics: Science, Services and Agents on the World Wide Web* (2008). DOI: <http://doi.acm.org/10.1016/j.websem.2007.11.003> (cit. on p. 283).
- [BHBL+10] Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, eds. *Linked Data on the Web (LDOW)*. CEUR Workshop Proceedings 628. Aachen, Apr. 2010. URL: <http://CEUR-WS.org/Vol-628/> (cit. on pp. 461, 498).
- [BHH+08] Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, eds. *The Semantic Web: Research and Applications*. 5th European Semantic Web Conference (ESWC) (Tenerife, Spain, June 1–5, 2008). Lecture Notes in Computer Science 5021. Springer Verlag, 2008 (cit. on pp. 453, 464, 471).

- [BHI+08] Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee, eds. *Linked Data on the Web (LDOW)*. CEUR Workshop Proceedings 369. Aachen, Apr. 2008. URL: <http://CEUR-WS.org/Vol-369/> (cit. on pp. 462, 496).
- [BHL+09] Tim Bray, Dave Hollander, Andrew Layman, Richard Tobin, and Henry S. Thompson. *Namespaces in XML 1.0 (Third Edition)*. W3C Recommendation. World Wide Web Consortium (W3C), Dec. 8, 2009. URL: <http://www.w3.org/TR/2009/REC-xml-names-20091208/> (cit. on pp. 54, 59).
- [Bilo8] William H. Billingsley. “The Intelligent Book: technologies for intelligent and adaptive textbooks, focussing on Discrete Mathematics.” PhD thesis. University of Cambridge, June 2008. URL: <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-719.pdf> (cit. on p. 317).
- [Biro8] Philipp Birken. *Mathematik in der deutschsprachigen Wikipedia*. Oct. 27, 2008. URL: <http://www.wias-berlin.de/events/insk/herbst08/birken.pdf> (visited on 2010-08-11) (cit. on p. 14).
- [BK07] Grzegorz Bancerek and Michael Kohlhase. “Towards a Mizar Mathematical Library in OMDoc Format.” In: *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*. Ed. by R. Matuszewski and A. Zalewska. Vol. 10. Studies in Logic, Grammar and Rhetoric 23. University of Białystok, 2007, pp. 265–275. URL: <http://kwarc.info/kohlhase/papers/trybook.pdf> (cit. on p. 69).
- [BL06a] Tim Berners-Lee. *Design Issues: Linked Data*. 2006. URL: <http://www.w3.org/DesignIssues/LinkedData.html> (visited on 2010-01-20) (cit. on p. 53).
- [BL06b] Tim Berners-Lee. *Notation 3 (N3) – A readable RDF syntax*. Tech. rep. World Wide Web Consortium (W3C), Mar. 9, 2006. URL: <http://www.w3.org/DesignIssues/Notation3.html> (visited on 2010-08-10) (cit. on p. 82).
- [BL09] Tim Berners-Lee. *cwm. a general purpose data processor for the semantic web*. Oct. 20, 2009. URL: <http://www.w3.org/2000/10/swap/doc/cwm.html> (visited on 2010-08-10) (cit. on p. 82).
- [BL90] Tim Berners-Lee. *HyperText and CERN*. Tech. rep. CERN, May 1990. URL: <http://www.w3.org/Administration/HTandCERN.txt> (cit. on p. 7).
- [BLCC+06] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. “Tabulator: Exploring and Analyzing linked data on the Semantic Web.” English. In: *Proceedings of the The 3rd International Semantic Web User Interaction Workshop (SWUI06)*. Nov. 2006 (cit. on p. 209).
- [BLFM05] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. Internet Engineering Task Force (IETF), 2005. URL: <http://www.ietf.org/rfc/rfc3986.txt> (cit. on pp. 51, 137, 210).
- [BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web – Computers navigating tomorrow’s Web will understand more of what’s going on making it more likely that you’ll get what you really want.” In: *Scientific American* 284 (2001) (cit. on p. 8).

- [BMo4a] M. Bidoit and Peter D. Mosses. *CASL — the Common Algebraic Specification Language: User Manual*. LNCS 2900. Springer Verlag, 2004 (cit. on p. 69).
- [BMo4b] Paul V. Biron and Ashok Malhotra. *XML Schema Part 2: Datatypes Second Edition*. W3C Recommendation. World Wide Web Consortium (W3C), Oct. 28, 2004. URL: <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/> (cit. on pp. 168, 202).
- [BM10] Dan Brickley and Libby Miller. *FOAF Vocabulary Specification 0.98*. Tech. rep. ILRT Bristol, Aug. 9, 2010. URL: <http://xmlns.com/foaf/spec/20100809.html> (cit. on pp. 128, 155, 157).
- [BPo8a] Joachim Baumeister and Frank Puppe. “Web-based Knowledge Engineering using Knowledge Wikis.” In: *Proc. of the AAAI Spring Symposium on “Symbiotic Relationships between Semantic Web and Knowledge Engineering”*. 2008, pp. 1–13 (cit. on p. 283).
- [BPo8b] Diego Berrueta and Jon Phipps. *Best Practice Recipes for Publishing RDF Vocabularies*. W3C Working Group Note. World Wide Web Consortium (W3C), Aug. 28, 2008. URL: <http://www.w3.org/TR/2008/NOTE-swbp-vocab-pub-20080828/> (cit. on pp. 211, 363).
- [BPSM+08] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C Recommendation. World Wide Web Consortium (W3C), Nov. 26, 2008. URL: <http://www.w3.org/TR/2004/REC-xml-20081126> (cit. on pp. 53, 54).
- [BR00] M. Ted Boren and Judith Ramey. “Thinking Aloud: Reconciling Theory and Practice.” In: *IEEE Transactions on Professional Communication* 43.3 (Sept. 2000), pp. 261–278 (cit. on pp. 324, 337).
- [Bra03] Tim Bray. *On Semantics and Markup*. Apr. 9, 2003. URL: <http://www.tbray.org/ongoing/When/200x/2003/04/09/SemanticMarkup> (visited on 2010-08-11) (cit. on p. 62).
- [Bra97] Scott Bradner. *Key words for use in RFCs to indicate requirement levels*. RFC 2119. Internet Engineering Task Force (IETF), 1997. URL: <http://www.ietf.org/rfc/rfc2119.txt> (cit. on p. 51).
- [Bro96] John Brooke. “SUS – a quick and dirty usability scale.” In: *Usability Evaluation in Industry*. Ed. by Patrick W. Jordan, Bruce Thomas, Bernard A. Weerdmeester, and Ian L. McClelland. London: Taylor & Francis, 1996, pp. 189–194 (cit. on p. 342).
- [Bru87] N. G. de Bruijn. “The Mathematical Vernacular, a language for mathematics with typed sets.” In: *Proceedings of the Workshop on Programming Languages*. Ed. by P. Dybjer et al. 1987 (cit. on p. 35).
- [Brä05] Andreas Brändle. “Zu wenige Köche verderben den Brei. Eine Inhaltsanalyse der Wikipedia aus Perspektive der journalistischen Qualität, des Netzeffekts und der Ökonomie der Aufmerksamkeit.” MA thesis. Universität Zürich, 2005 (cit. on p. 286).

- [Brö07] Matthias Bröcheler. “A Mathematical Semantic Web.” Bachelor’s Thesis. Computer Science, Jacobs University, Bremen, 2007. URL: <http://www.eecs.jacobs-university.de/archive/bsc-2007/broechele.pdf> (cit. on pp. 141, 158).
- [BS05] Jonathan Borwein and Terry Stanway. “Knowledge and Community in Mathematics.” In: *The Mathematical Intelligencer* 27.2 (2005), pp. 7–16 (cit. on p. 4).
- [BTS+] Abraham Bernstein, Jonas Tappolet, Henry Story, et al. *baetle – Bug And Enhancement Tracking Language*. URL: <http://baetle.googlecode.com> (visited on 2009-10-27) (cit. on p. 135).
- [Bug] *Bugzilla*. URL: <http://www.bugzilla.org> (visited on 2009-10-27) (cit. on p. 231).
- [Bus01] Stephen Buswell. *RDF/XML Test cases for RDF Logic, Web Ontology and Maths content*. Deliverable 5.3b. Semantic Web Advanced Development for Europe (SWAD-Europe), 2001. URL: [http://www.w3.org/2001/sw/Europe/reports/xml\\_test\\_cases/wp53.html](http://www.w3.org/2001/sw/Europe/reports/xml_test_cases/wp53.html) (cit. on pp. 85, 117).
- [BW10] François Bry and Klara A. Weiland. “Flavors of KWQL, a Keyword Query Language for a Semantic Wiki.” In: *SOFSEM. 36th Conference on Current Trends in Theory and Practice of Computer Science* (Špindlerův Mlýn, Czech Republic, Jan. 23–29, 2010). Ed. by Jan van Leeuwen, Anca Muscholl, David Peleg, Jaroslav Pokorný, and Bernhard Rumpe. Vol. 5901. Lecture Notes in Computer Science. Springer, 2010, pp. 247–258. ISBN: 978-3-642-11265-2. DOI: 10.1007/978-3-642-11266-9\_21 (cit. on pp. 284, 310).
- [BZO+07] Nathan Bos, Ann Zimmermann, Judith Olson, Jude Yew, Jason Yerkie, Erik Dahl, and Gary Olson. “From shared databases to communities of practice: A taxonomy of collaboratories.” In: *Journal of Computer-Mediated Communication* 12.2 (2007). article 16. URL: <http://jcmc.indiana.edu/vol12/issue2/bos.html> (cit. on p. 354).
- [C2p] *People, Projects, and Patterns*. URL: <http://c2.com/cgi/wiki?PeopleProjectsAndPatterns> (visited on 2010-10-03) (cit. on p. 282).
- [Car10] Pierre Cartier. “Can we make Mathematics universal as well as fully reliable?” In: [ACD+10]. ISBN: 3642141277 (cit. on p. 4).
- [CB87] Jeff Conklin and Michael L. Begeman. “gIBIS: A Hypertext Tool for Team Design Deliberation.” In: *ACM Hypertext*. ACM Press, 1987, pp. 247–251 (cit. on pp. 49, 50).
- [CCJ+04] Arjeh Cohen, Hans Cuypers, Dorina Jibetean, and Mark Spanbroek. *LeActiveMath Exercise Language*. Deliverable D7. LeActiveMath Consortium, Dec. 2004. URL: [http://www.leactivemath.org/deliverables/D7\\_Exercise\\_Language.pdf](http://www.leactivemath.org/deliverables/D7_Exercise_Language.pdf) (cit. on p. 48).
- [CCJ+06] Arjeh M. Cohen, Hans Cuypers, Dorina Jibetean, and Mark Spanbroek. “Interactive Learning and Mathematical Calculus.” In: [Koho06a], pp. 330–345 (cit. on p. 240).

- [CCK+08] Hans Cuypers, Arjeh M. Cohen, Jan Willem Knopper, Rikko Verrijzer, and Mark Spanbroek. “MathDox, a system for interactive Mathematics.” In: *Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecommunications 2008 (ED-MEDIA’08)*. (Vienna, Austria, June 2008). AACE, June 2008, pp. 5177–5182. URL: <http://go.editlib.org/p/29092> (cit. on pp. 24, 65, 80, 240, 241).
- [CCV10] Arjeh M. Cohen, Hans Cuypers, and Rikko Verrijzer. “Mathematical Context in Interactive Documents.” In: *Mathematics in Computer Science 3.3 (2010): Special Issue on Authoring, Digitalization and Management of Mathematical Knowledge*. Ed. by Serge Autexier, Petr Sojka, and Masakazu Suzuki, pp. 331–347 (cit. on pp. 18, 48).
- [CDF+09] Don Chamberlin, Michael Dyck, Daniela Florescu, Jim Melton, Jonathan Robie, and Jérôme Siméon. *XQuery Update Facility 1.0*. W3C Candidate Recommendation. World Wide Web Consortium (W3C), June 9, 2009. URL: <http://www.w3.org/TR/2009/CR-xquery-update-10-20090609> (cit. on p. 225).
- [CDSC+09] Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, eds. *MKM/Calculus Proceedings*. LNAI 5625. Springer Verlag, July 2009. ISBN: 9783642026133 (cit. on pp. 465, 470, 474, 477, 483).
- [CDTo4a] Olga Caprotti, Mike Dewar, and Daniele Turi. “Mathematical Service Matching Using Description Logic and OWL.” In: *Mathematical Knowledge Management, MKM’04*. Ed. by Andrea Asperti, Grzegorz Bancerek, and Andrej Trybulec. LNAI 3119. Springer Verlag, 2004, pp. 73–87 (cit. on pp. 17, 85).
- [CDTo4b] Olga Caprotti, Mike Dewar, and Daniele Turi. *Mathematical Service Matching Using Description Logic and OWL*. Tech. rep. The MONET Consortium, 2004. URL: [http://monet.nag.co.uk/monet/publicdocs/monet\\_onts.pdf](http://monet.nag.co.uk/monet/publicdocs/monet_onts.pdf) (visited on 2010-08-11) (cit. on p. 18).
- [Cer] Davide P. Cervone. *jsMath. A Method of Including Mathematics in Web Pages*. URL: <http://www.math.union.edu/~dpvc/jsMath/> (visited on 2010-10-04) (cit. on p. 284).
- [CFG+10] Richard Cyganiak, Simon Field, Arofan Gregory, Wolfgang Halb, and Jeni Tennison. “Semantic Statistics: Bringing Together SDMX and SCOVO.” In: [BHBL+10]. URL: <http://CEUR-WS.org/Vol-628/> (cit. on p. 101).
- [CGo6] Paul Cairns and Jeremy Gow. “Literate Proving: Presenting and Documenting Formal Proofs.” In: [Koho6a], pp. 159–173 (cit. on pp. 11, 82).
- [CGHM+08] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. “OWL 2: The next step for OWL.” In: *Web Semantics: Science, Services and Agents on the World Wide Web 6.4 (2008)*, pp. 309–322 (cit. on p. 148).
- [CHB+05] Jeremy J. Carroll, Pat Hayes, Christian Bizer, and Patrick Stickler. “Named Graphs, Provenance and Trust.” In: [EHo5], pp. 613–622. ISBN: 1-59593-046-9 (cit. on p. 149).



- [Che10] Xiaoyu Chen. “Electronic Geometry Textbook: A Geometric Textbook Knowledge Management System.” In: [ACD+10], pp. 278–292. ISBN: 3642141277. arXiv:1005.0080v1 [cs.AI] (cit. on pp. 35, 142).
- [CHMo8] Peter Coetzee, Tom Heath, and Enrico Motta. “SparqPlug: Generating Linked Data from Legacy HTML, SPARQL and the DOM.” In: [BHI+08]. URL: <http://CEUR-WS.org/Vol-369/> (cit. on p. 270).
- [CJo9] Richard Cyganiak and Anja Jentzsch. *About the Linking Open Data dataset cloud*. Sept. 22, 2009. URL: <http://lod-cloud.net> (visited on 2010-09-23) (cit. on p. 9).
- [CKo7] Pierre Corbineau and Cezary Kaliszyk. “Cooperative Repositories for Formal Proofs.” In: [KKM+07], pp. 221–234. ISBN: 978-3-540-73083-5 (cit. on pp. 14, 302).
- [CKKC+09] Olivier Corby, Leila Kefi-Khelif, Hacène Cherfi, Fabien Gandon, and Khaled Khelif. *Querying the Semantic Web of Data using SPARQL, RDF and XML*. Tech. rep. 6847. INRIA Sophia Antipolis, Feb. 2009. URL: <http://hal.inria.fr/docs/00/36/23/81/PDF/RR-6847.pdf> (cit. on p. 60).
- [CLM+09] Tim Clark, Joanne S. Luciano, M. Scott Marshall, Eric Prud’hommeaux, and Susie Stephens, eds. *Semantic Web Applications in Scientific Discourse (SWASD)*. CEUR Workshop Proceedings 523. Aachen, 2009. URL: <http://CEUR-WS.org/Vol-523/> (cit. on pp. 470, 492).
- [CMo1] James Clark and Makoto Murata. *RELAX NG Specification*. Tech. rep. OASIS, Dec. 3, 2001. URL: <http://www.relaxng.org/spec-20011203.html> (cit. on p. 54).
- [Cnxa] *Connexions*. URL: <http://cnx.org> (visited on 2009-10-22) (cit. on p. 14).
- [Cnxb] *Connexions – FAQ*. URL: <http://cnx.org/help/faq#OpenAccess> (visited on 2010-07-14) (cit. on p. 14).
- [Cnxc] *Connexions – XML Languages*. URL: <http://cnx.org/help/authoring/xml> (visited on 2010-07-31) (cit. on pp. 78, 80).
- [Cnxd] *Connexions MathML Editor*. URL: <http://cnx.org/matheditor> (visited on 2010-07-31) (cit. on pp. 186, 195).
- [COo1] Olga Caprotti and Martijn Oostdijk. “On Communicating Proofs in Interactive Mathematical Documents.” In: *Proceedings of Artificial Intelligence and Symbolic Computation, AISC’2000*. Ed. by Eugenio Roanes Lozano. LNAI 1930. Springer Verlag, 2001, pp. 53–64 (cit. on p. 67).
- [Cono7] Dan Connolly. *Gleaning Resource Descriptions from Dialects of Languages (GRDDL)*. W3C Recommendation. World Wide Web Consortium (W3C), Sept. 11, 2007. URL: <http://www.w3.org/TR/2007/REC-grddl-20070911/> (cit. on p. 268).
- [Coq] *The Coq Proof Assistant*. URL: <http://coq.inria.fr/> (visited on 2010-07-31) (cit. on pp. 17, 81).

- [CP10] Pierre Antoine Champin and Alexandre Passant. “SIOC in Action. Representing the Dynamics of Online Communities.” In: [PHP+10]. ISBN: 978-1-4503-0014-8. URL: <http://portal.acm.org/citation.cfm?id=1839707> (cit. on pp. 48, 124, 127).
- [CS04] Jeremy J. Carroll and Patrick Stickler. *TriX: RDF Triples in XML*. Tech. rep. HPL-2004-56. HP Laboratories Bristol, May 13, 2004. URL: <http://www.hpl.hp.com/techreports/2004/HPL-2004-56.pdf> (cit. on p. 126).
- [CS07] Olga Caprotti and Mika Seppälä. *Evaluation criteria for eContent quality*. Deliverable D2.1. JEM, Nov. 1, 2007. URL: <http://jem-thematic.net/en/D2.1> (cit. on p. 201).
- [CSH+07] Milena C. Caires, Simon Scerri, Siegfried Handschuh, Michael Sintek, and Ludger van Elst. “A Protégé Plug-in Development to Support the NEPOMUK Representational Language.” In: *10th International Protégé Conference*. Ed. by Tania Tudorache and Timothy Redmond. 2007 (cit. on p. 149).
- [CT04] *XML Information Set (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 4, 2004. URL: <http://www.w3.org/TR/2004/REC-xml-infoset-20040204> (cit. on p. 54).
- [Cun+] Ward Cunningham et al. *Wiki Design Principles*. URL: <http://c2.com/cgi/wiki?WikiDesignPrinciples> (visited on 2009-10-29) (cit. on p. 282).
- [Cun+02] Ward Cunningham et al. *What is Wiki*. June 27, 2002. URL: <http://wiki.org/wiki.cgi?WhatIsWiki> (visited on 2009-10-28) (cit. on p. 8).
- [CX05] Isabel F. Cruz and Huiyong Xiao. “The Role of Ontologies in Data Integration.” In: *Engineering Intelligent Systems for Electrical Engineering and Communication* 13.4 (2005), pp. 245–252. ISSN: 1363-2078 (cit. on pp. 61, 118, 143).
- [CX09] Isabel F. Cruz and Huiyong Xiao. “Ontology Driven Data Integration in Heterogeneous Networks.” In: *Complex Systems in Knowledge-Based Environments*. Ed. by Andreas Tolk and Lakhmi Jain. Studies in Computational Intelligence 168. Springer, 2009, pp. 75–97 (cit. on pp. 118, 143).
- [Dai] *Specifications for the Digital Talking Book*. Tech. rep. ANSI/NISO Z39.86-2005. DAISY Consortium, Apr. 21, 2005. URL: <http://www.daisy.org/z3986/2005/Z3986-2005.html> (cit. on p. 78).
- [Dav05] Ian Davis. *The Sixteen Faces of Eve*. Sept. 27, 2005. URL: <http://iandavis.com/blog/2005/09/the-sixteen-faces-of-eve> (visited on 2009-10-22) (cit. on p. 391).
- [Dav09] James H. Davenport, ed. *22nd OpenMath Workshop*. July 2009. URL: <http://staff.bath.ac.uk/masjhd/OM2009.html> (cit. on pp. 480, 482).
- [Dav99] James H. Davenport. *A Small OpenMath Type System*. Tech. rep. The OpenMath Esprit Project, 1999. URL: <http://www.openmath.org/standard/sts.pdf> (cit. on pp. 67, 200).



- [Dbl] D2R Server publishing the DBLP Bibliography Database. URL: <http://dblp.13s.de/d2r/> (visited on 2010-11-15) (cit. on p. 356).
- [Dbp] DBpedia. URL: <http://dbpedia.org> (visited on 2010-01-23) (cit. on pp. 8, 355).
- [Dcm] Dublin Core Metadata Element Set. DCMI Recommendation. Version 1.1. Dublin Core Metadata Initiative, Jan. 14, 2008. URL: <http://dublincore.org/documents/2008/01/04/dces/> (cit. on p. 45).
- [DD09] Frederico Durão and Peter Dolog. “Analysis of Tag-Based Recommendation Performance for a Semantic Wiki.” In: [LSSM+09]. URL: <http://ceur-ws.org/Vol-464/> (cit. on p. 310).
- [DDG+10] Li Ding, Dominic DiFranzo, Alvaro Graves, James R. Michaelis, Xian Li, Deborah L. McGuinness, and Jim Hendler. “Data-gov Wiki: Towards Linking Government Data.” In: [BCH+10] (cit. on p. 20).
- [Deco7] David Decraene. *Online Ontology Visualisation: Embedding OWL-RDFS syntax in XHTML with RDFa*. Nov. 2007. URL: <http://ontologyonline.blogspot.com/2007/11/embedding-owl-rdfs-syntax-in-xhtml-with.html> (cit. on p. 149).
- [DEM+08] Klaas Dellschaft, Hendrik Engelbrecht, José Monte Barreto, Sascha Rutenbeck, and Steffen Staab. “Cicero: Tracking Design Rationale in Collaborative Ontology Engineering.” In: [BHH+08], pp. 782–786 (cit. on p. 303).
- [dGCL+10] Mathieu d’Aquin, Alexander García Castro, Christoph Lange, and Kim Viljanen, eds. *Proceedings of the 1st Workshop on Ontology Repositories and Editors, Extended Semantic Web Conference*. (Hersonissos, Crete, Greece, May 31, 2010). CEUR Workshop Proceedings 596. Aachen, 2010. URL: <http://ceur-ws.org/Vol-596/> (cit. on pp. 484, 504).
- [DGG+08] Klaas Dellschaft, Aldo Gangemi, Jose Manuel Gomez, Holger Lewen, Valentina Presutti, and Margherita Sini. *Practical Methods to Support Collaborative Ontology Design*. Ed. by Klaas Dellschaft. NEON EU-IST-2005-027595 Deliverable D2.3.1. Feb. 2008. URL: [http://www.neon-project.org/web-content/images/Publications/neon\\_2008\\_d2.3.1.pdf](http://www.neon-project.org/web-content/images/Publications/neon_2008_d2.3.1.pdf) (cit. on pp. 232, 303).
- [DGK+10] Catalin David, Deyan Ginev, Michael Kohlhase, and Joe Corneli. “eMath 3.0: Building Blocks for a social and semantic Web for online mathematics & ELearning.” In: *1st International Workshop on Mathematics and ICT: Education, Research and Applications*. (Bucharest, Romania, Nov. 3, 2010). Ed. by Ion Mierlus-Mazilu. 2010. URL: <http://kwarc.info/kohlhase/papers/malog10.pdf> (cit. on p. 352).
- [DGN+03] Peter Dolog, Rita Gavrioloaie, Wolfgang Nejdl, and Jan Brase. “Integrating Adaptive Hypermedia Techniques and Open RDF-based Environments.” In: *Proceedings of the 12th WWW conference*. (Budapest, Hungary, May 20–24, 2005). ACM Press, 2003 (cit. on p. 125).
- [DH81] Philip J. Davis and Reuben Hersh. *The Mathematical Experience*. Boston: Birkhäuser, 1981 (cit. on pp. 4, 5, 37).

- [Dij86] Edsger W. Dijkstra. “On a Cultural Gap.” In: *Mathematical Intelligencer* 8.1 (1986), pp. 48–52 (cit. on pp. 3, 192).
- [Dit] OASIS Darwin Information Typing Architecture (DITA). URL: <http://www.oasis-open.org/committees/dita/> (visited on 2010-02-02) (cit. on p. 78).
- [DK09] James H. Davenport and Michael Kohlhase. “Unifying Math Ontologies: A tale of two standards.” In: [CDSC+09], pp. 263–278. ISBN: 9783642026133. URL: <http://kwarc.info/kohlhase/papers/mkm09-MML0M3.pdf> (cit. on p. 65).
- [DKL+10a] Catalin David, Michael Kohlhase, Christoph Lange, Florian Rabe, and Vyacheslav Zholudev. “JOBAD/MMT – Interactive Mathematics.” In: [GENL+10]. URL: <http://sites.google.com/a/fh-hannover.de/aimashup/home/jobad> (cit. on p. 388).
- [DKL+10b] Catalin David, Michael Kohlhase, Christoph Lange, Florian Rabe, Nikita Zhiltsov, and Vyacheslav Zholudev. “Publishing Math Lecture Notes as Linked Data.” In: *The Semantic Web: Research and Applications (Part II)*. 7th Extended Semantic Web Conference (ESWC) (Hersonissos, Crete, Greece, May 30–June 3, 2010). Ed. by Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache. Lecture Notes in Computer Science 6089. Springer Verlag, 2010, pp. 370–375. arXiv:1004.3390v1 [cs.DL] (cit. on pp. 237, 260, 349).
- [DL08] James H. Davenport and Paul Libbrecht. “The Freedom to Extend OpenMath and its Utility.” In: *Mathematics in Computer Science* 2.2 (2008). Ed. by Manfred Kerber, pp. 253–277 (cit. on pp. 65, 310).
- [DLR10] Catalin David, Christoph Lange, and Florian Rabe. “Interactive Documents as Interfaces to Computer Algebra Systems: JOBAD and Wolfram|Alpha.” In: *CALCULEMUS (Emerging Trends)*. Ed. by David Delahaye and Renaud Rioboo. Centre d’Étude et de Recherche en Informatique du CNAM (Cédric), 2010, pp. 13–30. URL: <https://svn.omdoc.org/repos/jomdoc/doc/pubs/calculumus10/jobad-cas.pdf> (cit. on pp. 255, 258, 260, 390).
- [DMLP79] Richard A. De Millo, Richard J. Lipton, and Alan J. Perlis. “Social Processes and Proofs of Theorems and Programs.” In: *Communications of the ACM* 22.5 (1979), pp. 271–280 (cit. on pp. 4, 5, 176).
- [DN03] James H. Davenport and William A. Naylor. *Units and Dimensions in OpenMath*. 2003. URL: <http://www.openmath.org/documents/Units.pdf> (visited on 2009-10-22) (cit. on p. 256).
- [DR10] Jos De Roo. *EulerSharp*. Aug. 5, 2010. URL: <http://eulersharp.sourceforge.net/> (visited on 2010-08-10) (cit. on p. 82).
- [DRGA+10] David De Roure, Carole Goble, Sergejs Aleksejevs, Sean Bechhofer, Jiten Bhagat, Don Cruickshank, Paul Fisher, Nandkumar Kollara, Danus Michaelides, Paolo Missier, David Newman, Marcus Ramsden, Marco Roos, Katy Wolstencroft, Ed Zaluska, and Jun Zhao. “The Evolution of myExperiment.” In: [Esc] (cit. on p. 354).

- [DRGS07] David De Roure, Carole Goble, and Robert Stevens. “The design and realisation of the myExperiment virtual research environment for social sharing of workflows.” In: *Future Generation Computer Systems* 25 (2007), pp. 561–567. DOI: [10 . 101 6/j . future . 2008 . 06 . 010](https://doi.org/10.1016/j.future.2008.06.010) (cit. on p. 354).
- [DRS+06] Nicholas Drummond, Alan Rector, Robert Stevens, Georgina Moulton, Matthew Horridge, Hai Wang, and Julian Sedenberg. “Putting OWL in Order: Patterns for sequences in OWL.” In: *OWL: Experiences and Directions (OWLED)*. Ed. by Bernardo Cuenca Grau, Pascal Hitzler, Connor Shankey, and Evan Wallace. Nov. 2006 (cit. on p. 58).
- [DS05] Martin Dürst and Michel Suignard. *Internationalized Resource Identifiers (IRIs)*. RFC 3987. Internet Engineering Task Force (IETF), 2005. URL: <http://www.ietf.org/rfc/rfc3987.txt> (cit. on p. 53).
- [DsG+08] Duane Degler, mc schraefel, Jennifer Golbeck, Abraham Bernstein, and Lloyd Rutledge, eds. *Semantic Web User Interaction (SWUI), CHI. Exploring HCI Challenges*. (Florence, Italy, Apr. 5, 2008). CEUR Workshop Proceedings 543. Aachen, 2008. URL: <http://CEUR-WS.org/Vol-543/> (cit. on pp. 490, 493).
- [DSN01] Wolfgang Dalitz, Wolfram Sperber, and Winfried Neun. “Math-Net, a model for information and communication systems in sciences.” In: *EUNIS. 7th International Conference of European University Information Systems* (Berlin, Germany, Mar. 28–30, 2001). Ed. by Jan von Knop, Peter Schirmbacher, and Viljan Mahnic. LNI 13. GI, 2001, pp. 140–145. ISBN: 3-88579-339-3 (cit. on p. 16).
- [DSS+09] Renata Dividino, Simon Schenk, Sergej Sizov, and Steffen Staab. “Provenance, Trust, Explanations – and all that other Meta Knowledge.” In: *Künstliche Intelligenz* 2 (Feb. 2009), pp. 24–30 (cit. on p. 149).
- [DSWo8] Dominik Dietrich, Ewaryst Schulz, and Marc Wagner. “Authoring Verified Documents by Interactive Proof Construction and Verification in Text-Editors.” In: [\[ACR+08\]](#), pp. 398–414 (cit. on pp. 25, 301).
- [DuCo9] Bob DuCharme. *Using RDFa with DITA and DocBook*. Aug. 20, 2009. URL: <http://www.devx.com/semantic/Article/42543/> (visited on 2010-08-11) (cit. on pp. 81, 168).
- [DVR07] Michel Dumontier and Natalia Villanueva-Rosales. “Three-Layer OWL Ontology Design.” In: *WoMO*. (Whistler, Canada, Oct. 28, 2007). Ed. by Bernardo Cuenca Grau, Vasant Honavar, Anne Schlicht, and Frank Wolter. CEUR Workshop Proceedings 315. Aachen: CEUR-WS.org, 2007. URL: <http://ceur-ws.org/Vol-315/> (cit. on p. 154).
- [EGHo8] Anja Ebersbach, Markus Glaser, and Richard Heigl. *Wiki: Web Collaboration*. New York: Springer Verlag, 2008 (cit. on p. 282).
- [EH05] Allan Ellis and Tatsuya Hagino, eds. *Proceedings of the 14th international World Wide Web conference (WWW)*. (Chiba, Japan, May 10–14, 2005). ACM Press, 2005. ISBN: 1-59593-046-9 (cit. on pp. 461, 494).

- [Eix] Ramon Eixarch. *WIRIS Plugin for Moodle*. URL: <http://www.wiris.com/content/view/96/> (visited on 2009-11-10) (cit. on p. 195).
- [Enz99] Hans Magnus Enzensberger. *Drawbridge up. Mathematics — a cultural anathema*. German original: Zugbrücke außer Betrieb. Die Mathematik im Jenseits der Kultur. English translation by Tom Artin. A K PETERS, LTD., 1999 (cit. on pp. 19, 21).
- [Erio7] Henrik Eriksson. “The semantic-document approach to combining documents and ontologies.” In: *International Journal of Human-Computer Studies* 65.7 (2007), pp. 624–639 (cit. on pp. 158, 223).
- [Esc] 6th IEEE e-Science conference. (Brisbane, Australia, Dec. 2010). 2010 (cit. on pp. 455, 465).
- [Esw] *Custom RDF Dialects*. URL: <http://esw.w3.org/topic/CustomRdfDialects> (visited on 2010-02-02) (cit. on p. 60).
- [Evo] *EvoOnt – A Software Evolution Ontology*. URL: <http://www.ifi.uzh.ch/ddis/evo/> (visited on 2009-10-27) (cit. on p. 135).
- [Exi] *eXist database*. URL: <http://exist.sourceforge.net/> (visited on 2009-10-22) (cit. on p. 228).
- [Far04] William M. Farmer. “MKM: A new Interdisciplinary Field of Research.” In: *Bulletin of the ACM Special Interest Group on Symbolic and Automated Mathematics (SIGSAM)* 38.2 (2004), pp. 47–52 (cit. on p. 6).
- [FBS] Sergio Fernández, Uldis Bojārs, and Christopher Schmidt. *SpecGen v5 – ontology specification generator tool*. URL: <http://forge.morfeo-project.org/wiki/en/index.php/SpecGen> (visited on 2009-10-22) (cit. on p. 149).
- [Fei] Lee Feigenbaum. *SPARQL Protocol and Query Language: SPARQL Frequently Asked Questions*. URL: <http://www.thefigtrees.net/lee/sw/sparql-faq> (visited on 2010-04-17) (cit. on p. 384).
- [FF94] Andrea Fontana and James H. Frey. “Handbook of Qualitative Research.” In: ed. by Norman K. Denzin and Yvonna S. Lincoln. London: Sage, 1994, pp. 361–376 (cit. on p. 325).
- [FFB+07] Jill Freyne, Rosta Farzan, Peter Brusilovsky, Barry Smyth, and Maurice Coyle. “Collecting community wisdom: integrating social search & social navigation.” In: *Proceedings of the 12th international conference on Intelligent user interfaces (IUI)*. New York, NY, USA: ACM, 2007, pp. 52–61 (cit. on p. 316).
- [FG90] Gerhard Fischer and Andreas Girgensohn. “End-user modifiability in design environments.” In: *Conference on Human Factors in Computing Systems (CHI)*. (Seattle, WA, USA, 1990). Ed. by Jane Carrasco Chew and John Whiteside. New York, NY, USA: ACM, 1990, pp. 183–192. ISBN: 0-201-50932-6 (cit. on p. 343).
- [Fgd] *Content Standard for Digital Geospatial Metadata Workbook*. Workbook. Version 2.0. Federal Geographic Data Committee, May 1, 2000. URL: [http://www.fgdc.gov/metadata/documents/workbook\\_0501\\_bmk.pdf](http://www.fgdc.gov/metadata/documents/workbook_0501_bmk.pdf) (visited on 2010-08-11) (cit. on p. 36).

- [FGIo9] Sergio Fernández, Frédérick Giasson, and Kingsley Idehen. *SIOC Ontology: Application and Implementation Status*. Tech. rep. DERI Galway, May 15, 2009. URL: <http://rdfs.org/sioc/applications/> (visited on 2010-09-24) (cit. on p. 229).
- [FGT92] William Farmer, Josuah Guttman, and Xavier Thayer. “Little Theories.” In: *Proceedings of the 11th Conference on Automated Deduction*. Ed. by D. Kapur. LNCS 607. Saratoga Springs, NY, USA: Springer Verlag, 1992, pp. 467–581 (cit. on p. 38).
- [FHTo6] Colin Fraser, Harry Halpin, and Kavita E. Thomas. “Developing an Argumentation Ontology for Mailing Lists.” In: *AIMSA*. Ed. by Jérôme Euzenat and John Domingue. LNAI 4183. Springer Verlag, 2006, pp. 150–161 (cit. on p. 135).
- [Fie00] Roy T. Fielding. “Architectural Styles and the Design of Network-based Software Architectures.” PhD thesis. University of California, Irvine, 2000. URL: <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> (cit. on pp. 242, 249).
- [Fie01] Armin Fiedler. “*Prex*: An Interactive Proof Explainer.” In: *Automated Reasoning — 1st International Joint Conference, IJCAR 2001*. Ed. by Rajeev Goré, Alexander Leitsch, and Tobias Nipkow. LNAI 2083. Siena, Italy: Springer Verlag, 2001, pp. 416–420 (cit. on pp. 6, 39).
- [FKo6] Andreas Franke and Michael Kohlhase. “MBase, an Open Mathematical Knowledge Base.” In: [Koho6b]. Chap. 26.4. URL: <http://omdoc.org/pubs/omdoc1.2.pdf> (cit. on p. 225).
- [FLGPJ97] Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. “METHONTOLOGY: from Ontological Art towards Ontological Engineering.” In: *Proceedings of the Fourteenth National Conference on Artificial Intelligence AAAI-97*. (Stanford, USA, Mar. 1997). MIT Press, 1997, pp. 33–40 (cit. on p. 62).
- [FLT03] Frédéric Fürst, Michel Leclère, and Francky Trichet. “Ontological engineering and mathematical knowledge management: A formalization of projective geometry.” In: *Annals of Mathematics and Artificial Intelligence* 38 (2003), pp. 65–89 (cit. on p. 141).
- [Foc] *FoCaLiZe*. URL: <http://focalize.inria.fr/> (visited on 2010-08-29) (cit. on p. 82).
- [For] *Formalized Mathematics. a computer assisted approach*. URL: <http://fm.mizar.org> (visited on 2009-12-02) (cit. on pp. 11, 81).
- [Fre] *Freebase*. URL: <http://www.freebase.com> (visited on 2010-07-17) (cit. on p. 10).
- [FTA+07] Norbert Fuhr, Giannis Tsakonas, Trond Aalberg, Maristella Agosti, Preben Hansen, Sarantos Kapidakis, Claus-Peter Klas, László Kovács, Monica Landoni, András Micsik, Christos Papatheodorou, Carol Peters, and Ingeborg Sølvberg. “Evaluation of digital libraries.” In: *International Journal of Digital Libraries* 8 (2007), pp. 21–38 (cit. on pp. 314, 315).

- [Gam] GAMS: *Guide to Available Mathematical Software*. National Institute of Standards and Technology (NIST). URL: <http://gams.nist.gov> (visited on 2009-12-16) (cit. on p. 47).
- [Gan94] Mike Gancarz. *The UNIX Philosophy*. Digital Press, 1994. ISBN: 978-1555581237 (cit. on p. 175).
- [Gar05] Jesse James Garrett. *Ajax: A new approach to web applications*. Tech. rep. Seen February 2006. Adaptive Path, Feb. 18, 2005. URL: <http://www.adaptivepath.com/publications/essays/archives/000385.php> (cit. on p. 241).
- [GBo4] Jan Grant and Dave Beckett. *RDF Test Cases*. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 10, 2004. URL: <http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/> (cit. on p. 56).
- [GBO+08] Michael Gruninger, Olivier Bodenreider, Frank Olken, Leo Obrst, and Peter Yim. “Ontology Summit 2007 – Ontology, taxonomy, folksonomy: Understanding the distinctions.” In: *Applied Ontology* 3 (2008), pp. 191–200. DOI: [10.3233/A0-2008-0052](https://doi.org/10.3233/A0-2008-0052) (cit. on p. 61).
- [GC05] Roberto García and Òscar Celma. “Semantic Integration and Retrieval of Multimedia Metadata.” In: *Proceedings of the ISWC Workshop on Knowledge Markup and Semantic Annotation (SemAnnot)*. (Galway, Ireland, Nov. 7, 2005). Ed. by Siegfried Handschuh, Thierry Declerck, and Marja-Riitta Koivunen. CEUR Workshop Proceedings 185. Aachen: CEUR-WS.org, 2005. URL: <http://ceur-ws.org/Vol-185/> (cit. on p. 140).
- [GC07] Jeremy Gow and Paul Cairns. “Closing the Gap Between Formal and Digital Libraries of Mathematics.” In: *Studies in Logic, Grammar and Rhetoric* 10.23 (2007) (cit. on pp. 12, 35).
- [GENL+10] Adrian Giurca, Brigitte Endres-Niggemeyer, Christoph Lange, Lutz Maicher, and Pascal Hitzler, eds. *AI Mashup Challenge*. June 2010. URL: <http://sites.google.com/a/fh-hannover.de/aimashup/> (cit. on pp. 465, 499, 502).
- [GF+92] M. Genesereth, R. Fikes, et al. *Knowledge Interchange Format: Version 3.0 Reference Manual*. Tech. rep. Computer Science Department, Stanford University, 1992 (cit. on p. 62).
- [GGPM05] Giorgi Gogvadze, Alberto González Palomo, and Erice Melis. “Interactivity of Exercises in ActiveMath.” In: *Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences. Sharing Good Practices of Research, Experimentation and Innovation*. Ed. by Chee-Kit Looi, David Jonassen, and Mitsuru Ikeda. Frontiers in Artificial Intelligence and Applications 133. IOS Press, 2005, pp. 109–115 (cit. on pp. 48, 71).
- [GH09a] Tudor Groza and Siegfried Handschuh. *SALT Annotation Ontology*. Tech. rep. Digital Enterprise Research Institute (DERI), 2009. URL: <http://salt.semanticauthoring.org/ontologies/sao> (visited on 2010-08-21) (cit. on p. 119).



- [GH09b] Tudor Groza and Siegfried Handschuh. *SALT Document Ontology*. Tech. rep. Digital Enterprise Research Institute (DERI), 2009. URL: <http://salt.semanticauthoring.org/ontologies/sdo> (visited on 2010-08-21) (cit. on p. 119).
- [GH09c] Tudor Groza and Siegfried Handschuh. *SALT Rhetorical Ontology*. Tech. rep. Digital Enterprise Research Institute (DERI), 2009. URL: <http://salt.semanticauthoring.org/ontologies/sro> (visited on 2010-08-21) (cit. on p. 120).
- [GHC+09] Tudor Groza, Siegfried Handschuh, Tim Clark, Simon Buckingham Shum, and Anita de Waard. “A Short Survey of Discourse Representation Models.” In: [CLM+09]. URL: <http://CEUR-WS.org/Vol-523/> (cit. on pp. 39, 49, 119, 136).
- [GHJ+08] Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and Sylvia Stuurman. *Feedback Services for Exercise Assistants*. Tech. rep. UU-CS-2008-018. Utrecht University, July 2008 (cit. on pp. 240, 241).
- [GHM+07] Tudor Groza, Siegfried Handschuh, Knud Möller, and Stefan Decker. “SALT – Semantically Annotated L<sup>A</sup>T<sub>E</sub>X for Scientific Publications.” In: *The Semantic Web: Research and Applications*. 4th European Semantic Web Conference (ESWC) (Innsbruck, Austria, June 3–7, 2007). Ed. by Enrico Franconi, Michael Kifer, and Wolfgang May. Lecture Notes in Computer Science 4519. Springer Verlag, 2007, pp. 518–532. ISBN: 978-3-540-72666-1 (cit. on pp. 76, 118).
- [Gico8] Jana Giceva. *Capturing Rhetorical Aspects in Mathematical Documents using OM-Doc and SALT*. Technical Report. Jacobs University and DERI Galway, 2008. URL: [https://svn.kwarc.info/repos/supervision/intern/2008/giceva\\_jana/project/internship%20report.pdf](https://svn.kwarc.info/repos/supervision/intern/2008/giceva_jana/project/internship%20report.pdf) (visited on 2010-08-11) (cit. on pp. 144, 254, 260).
- [GJA+09] Deyan Ginev, Constantin Jucovschi, Stefan Anca, Mihai Grigore, Catalin David, and Michael Kohlhase. “An Architecture for Linguistic and Semantic Analysis on the arXMLiv Corpus.” In: *Applications of Semantic Technologies (AST) Workshop at Informatik 2009*. 2009. URL: <http://www.kwarc.info/projects/lamapun/pubs/AST09-LaMaPUn+appendix.pdf> (cit. on pp. 39, 356).
- [GK97] Thomas F. Gordon and Nikos Karacapilidis. “The Zeno Argumentation Framework.” In: *Sixth International Conference on Artificial Intelligence and Law*. ACM Press, 1997, pp. 10–18 (cit. on p. 234).
- [GKX09] James Gardner, Aaron Krowne, and Li Xiong. “NNexus: Towards an Automatic Linker for a Massively-Distributed Collaborative Corpus.” In: *IEEE Transactions on Knowledge and Data Engineering* 21.6 (June 2009) (cit. on p. 301).
- [GLR09] Jana Giceva, Christoph Lange, and Florian Rabe. “Integrating Web Services into Active Mathematical Documents.” In: [CDSC+09], pp. 279–293. ISBN: 9783642026133. URL: <https://svn.omdoc.org/repos/jomdoc/doc/pubs/mkm09/jobad/jobad-server.pdf> (cit. on pp. 237, 260).

- [GM05] Stephan Grimm and Boris Motik. “Closed World Reasoning in the Semantic Web through Epistemic Operators.” In: *OWL: Experiences and Directions (OWLED)*. Ed. by Bernardo Cuenca Grau, Ian Horrocks, Bijan Parsia, and Peter Patel-Schneider. Nov. 2005 (cit. on p. 204).
- [GM08] George Goguadze and Erica Melis. “Feedback in ActiveMath Exercises.” In: *International Conference on Mathematics Education (ICME)*. 2008 (cit. on p. 240).
- [GM99] Ashish Gupta and Iderpal Singh Mumick, eds. *Materialized views: techniques, implementations, and applications*. Cambridge, MA, USA: MIT Press, 1999. ISBN: 0-262-57122-6 (cit. on p. 295).
- [GMB08] Aurona Gerber, Alta van der Merwe, and Andries Barnard. “A Functional Semantic Web Architecture.” In: [BHH+08] (cit. on p. 51).
- [GMH+07] Tudor Groza, Knud Möller, Siegfried Handschuh, Diana Trif, and Stefan Decker. “SALT: Weaving the Claim Web.” In: [ACN+07], pp. 197–210. ISBN: 978-3-540-76297-3 (cit. on pp. 76, 118, 119).
- [GMM+03] Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. *W3C XPointer Framework*. W3C Recommendation. World Wide Web Consortium (W3C), Mar. 25, 2003. URL: <http://www.w3.org/TR/2003/REC-xptr-framework-20030325/> (cit. on p. 54).
- [GN09] Timothy Gowers and Michael Nielsen. “Massively collaborative mathematics.” In: *Nature* 461.15 (2009), pp. 879–881 (cit. on pp. 4, 13, 49, 230).
- [GO10] Birte Glimm and Chimezie Ogbuji. *SPARQL 1.1 Entailment Regimes*. W3C Working Draft. World Wide Web Consortium (W3C), Oct. 14, 2010. URL: <http://www.w3.org/TR/2010/WD-sparql11-entailment-20101014/> (cit. on p. 228).
- [GO94] Thomas R. Gruber and Gregory R. Olsen. “An Ontology for Engineering Mathematics.” In: *Fourth International Conference on Principles of Knowledge Representation and Reasoning*. Ed. by Jon Doyle, Piero Torasso, and Erik Sandewall. Morgan Kaufmann, 1994 (cit. on p. 141).
- [Goe06] Johann Wolfgang von Goethe. *Maximen und Reflexionen*. Ed. by Helmut Koopmann. Kleine Bibliothek der Weltweisheit 14. München: Deutscher Taschenbuch Verlag, Dec. 2006. ISBN: 978-3-423-34378-7 (cit. on p. 261).
- [Gogo3a] George Goguadze. *Metadata for Mathematical Libraries*. Deliverable D3.a. MoWGLI, 2003. URL: [http://mowgli.cs.unibo.it/misc/deliverables/metadata/D3a\\_metadata\\_for\\_math/math\\_metadata.pdf](http://mowgli.cs.unibo.it/misc/deliverables/metadata/D3a_metadata_for_math/math_metadata.pdf) (cit. on p. 43).
- [Gogo3b] George Goguadze. *Metadata Model*. Deliverable D3.b. MoWGLI, 2003. URL: [http://mowgli.cs.unibo.it/misc/deliverables/metadata/D3bmetadata\\_model/metadata\\_model.pdf](http://mowgli.cs.unibo.it/misc/deliverables/metadata/D3bmetadata_model/metadata_model.pdf) (cit. on p. 86).
- [Gon05] Roberto García González. “A Semantic Web approach to Digital Rights Management.” PhD thesis. Barcelona: Universitat Pompeu Fabra, Nov. 2005. URL: <http://rhizomik.net/html/~roberto/thesis/> (cit. on p. 140).



- [Goo] Google. *Google Wave*. URL: <http://wave.google.com> (visited on 2010-10-03) (cit. on p. 282).
- [Gow09a] Timothy Gowers. *Is massively collaborative mathematics possible?* Jan. 27, 2009. URL: <http://gowers.wordpress.com/2009/01/27/is-massively-collaborative-mathematics-possible/> (visited on 2010-08-11) (cit. on p. 13).
- [Gow09b] Timothy Gowers. *Questions of procedure*. Feb. 1, 2009. URL: <http://gowers.wordpress.com/2009/02/01/questions-of-procedure/> (visited on 2010-08-11) (cit. on p. 13).
- [GPo6a] Alberto González Palomo. “QMath: A Human-Oriented Language and Batch Formatter for OMDoc.” In: [Koh06b]. Chap. 26.2. URL: <http://omdoc.org/pubs/omdoc1.2.pdf> (cit. on pp. 73, 185).
- [GPo6b] Alberto González Palomo. “Sentido: an Authoring Environment for OMDoc.” In: [Koh06b]. Chap. 26.3. URL: <http://omdoc.org/pubs/omdoc1.2.pdf> (cit. on pp. 191, 379).
- [Gro09] Tudor Groza. *SALT Syntax*. Tech. rep. Digital Enterprise Research Institute (DERI), 2009. URL: <http://salt.semanticauthoring.org/salt-syntax.html> (visited on 2010-08-21) (cit. on p. 122).
- [Gru07] Klaus Grue. “The Layers of Logiweb.” In: [KKM+07], pp. 250–264. ISBN: 978-3-540-73083-5 (cit. on p. 303).
- [GS03] Ferruccio Guidi and Irene Schena. “A Query Language for a Metadata Framework about Mathematical Resources.” In: [ABD03], pp. 105–118 (cit. on p. 17).
- [GSC03] Ferruccio Guidi and Claudio Sacerdoti Coen. “Querying Distributed Digital Libraries of Mathematics.” In: [HR03]. URL: <http://www.calculumus.net/meetings/rome03/Proceedings/final.pdf> (cit. on p. 86).
- [GSMT09] Shudi (Sandy) Gao, C. M. Sperberg-McQueen, and Henry S. Thompson. *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. W3C Candidate Recommendation. World Wide Web Consortium (W3C), Apr. 2009. URL: <http://www.w3.org/TR/2009/CR-xschema11-1-20090430/> (cit. on p. 54).
- [Gui03] Ferruccio Guidi. “Searching and Retrieving in Content-based Repositories of Formal Mathematical Knowledge.” PhD thesis. Università di Bologna, 2003 (cit. on p. 17).
- [GUM+04] George Gogvadze, Carsten Ullrich, Erica Melis, Jörg Siekmann, Chistian Gross, and Rafael Morales. *LeActiveMath Structure and Metadata Model*. Deliverable D6. LeActiveMath Consortium, 2004. URL: <http://www.activemath.org/pubs/LeAM-D6.pdf> (cit. on p. 162).
- [Har40] Godfrey H. Hardy. *A Mathematician’s Apology*. Cambridge University Press, 1940 (cit. on p. 34).
- [Hay04] Patrick Hayes. *RDF Semantics*. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 10, 2004. URL: <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/> (cit. on p. 55).

- [HC09] Aidan Hogan and Richard Cyganiak. *Frequently Observed Problems on the Web of Data*. Tech. rep. Version vo.3. Pedantic Web Group, Nov. 13, 2009. URL: <http://pedantic-web.org/fops.html> (cit. on p. 166).
- [Hea+] Tom Heath et al. *Linked Data – Connect Distributed Data across the Web – Glossary*. URL: <http://linkeddata.org/glossary> (visited on 2010-09-28) (cit. on p. 138).
- [Hea09] David Heath. “OpenMath Content Dictionary Manager.” MA thesis. University of Birmingham, Sept. 2009. URL: <http://www.david-heath.co.uk/wp-content/uploads/2009/09/OpenMathCDManager.pdf> (cit. on pp. 187, 302).
- [Heio0] Bettina Heintz. *Die Innenwelt der Mathematik. Zur Kultur und Praxis einer beweisenden Disziplin*. Springer Verlag Wien, 2000 (cit. on pp. 4–6, 41, 176).
- [Hel] HELM. *Hypertextual Electronic Library of Mathematics*. URL: <http://helm.cs.unibo.it> (visited on 2010-07-15) (cit. on pp. 16, 17).
- [HHK+10] Kevin Hammond, Peter Horn, Alexander Konovalov, Steve Linton, Dan Roozmond, Abdallah Al Zain, and Phil Trinder. “Easy Composition of Symbolic Computation Software: A New Lingua Franca for Symbolic Computation.” In: *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation (ISSAC)*. ACM Press, 2010, pp. 339–346 (cit. on p. 18).
- [HHM+09] Peter Haase, Daniel Herzig, Mark Musen, and Thanh Tran. “Semantic Wiki Search.” In: [ATC+09], pp. 445–460 (cit. on p. 317).
- [HHM+10] Thomas C. Hales, John Harrison, Sean McLaughlin, Tobias Nipkow, Steven Obua, and Roland Zumkeller. “A Revision of the Proof of the Kepler Conjecture.” In: *Discrete and Computational Geometry* (2010). arXiv:0902.0350v1 [math.MG] (cit. on p. 5).
- [HHR+09] Michael Hausenblas, Wolfgang Halb, Yves Raimond, Lee Feigenbaum, and Danny Ayers. “SCOVO: Using Statistics on the Web of Data.” In: [ATC+09] (cit. on p. 130).
- [Hic10] Ian Hickson. *HTML5 (including next generation additions still in development)*. Draft Standard. Web Hypertext Application Technology Working Group (WHATWG), 2010. URL: <http://whatwg.org/html5> (visited on 2010-02-02) (cit. on p. 60).
- [HKMo7] David Huynh, David Karger, and Rob Miller. “Exhibit: Lightweight Structured Data Publishing.” In: *16th International World Wide Web Conference*. (Banff, Alberta, Canada, 2007). ACM, 2007. URL: <http://www2007.org/paper161.php> (cit. on p. 209).
- [HKSo6] Eberhard Hilf, Michael Kohlhase, and Heinrich Stamerjohanns. “Capturing the Content of Physics: Systems, Observables, and Experiments.” In: [BFo6], pp. 165–178. URL: <http://kwarc.info/kohlhase/papers/mkm06physml.pdf> (cit. on pp. 48, 162).

- [HLRO+10] Ralf Heese, Markus Luczak-Rösch, Radoslaw Oldakowski, Olga Streibel, and Adrian Paschke. “One Click Annotation.” In: *Scripting and Development for the Semantic Web (SFSW)*. Ed. by Gunnar Aastrand Grimnes, Sören Auer, and Gregory Todd Williams. May 2010. URL: <http://www.semanticscripting.org/SFSW2010/> (cit. on p. 186).
- [HLS10] Philipp Heim, Steffen Lohmann, and Timo Stegemann. “Interactive Relationship Discovery via the Semantic Web.” In: [AAH+10], pp. 303–317 (cit. on p. 9).
- [HM+] Thomas C. Hales, Sean McLaughlin, et al. *The Flyspeck project*. URL: <http://flyspeck.googlecode.com> (visited on 2010-10-04) (cit. on p. 5).
- [HMF09] Olaf Hartig, Hannes Mühleisen, and Johann-Christoph Freytag. “Linked Data for Building a Map of Researchers.” In: [BAG09]. URL: <http://CEUR-WS.org/Vol-449/> (cit. on p. 9).
- [Hor] Matthew Horridge. *OWL 2 Validator*. Powered by the OWL API Version 3.0.0.1272. URL: <http://owl.cs.manchester.ac.uk/validator/> (visited on 2010-06-29) (cit. on pp. 202, 363).
- [Horo2] Ian Horrocks. “Reasoning with Expressive Description Logics: Theory and Practice.” In: *Automated Deduction — CADE-18*. Ed. by Andrei Voronkov. LNAI 2392. Springer Verlag, 2002, pp. 1–15 (cit. on p. 61).
- [Hov+] Jean-François Hovinne et al. *WYMEditor – web-based XHTML editor*. URL: <http://www.wymentor.org> (visited on 2009-11-11) (cit. on p. 186).
- [HPH+] Jens Hartmann, Raúl Palma, Peter Haase, and Asunción Gómez-Pérez. *Ontology Metadata Vocabulary – OMV*. URL: <http://omv2.sourceforge.net> (visited on 2010-01-12) (cit. on pp. 48, 124).
- [HPR+08] Ivan Herman, Eric Prud’hommeaux, Thomas Roessler, and Rigo Wenning. *Team Comment on ccREL: The Creative Commons Rights Expression Language Member Submission*. W3C Team Comment. World Wide Web Consortium (W3C), Feb. 2008. URL: <http://www.w3.org/Submission/2008/02/Comment> (cit. on p. 126).
- [HPS09] Matthew Horridge and Peter F. Patel-Schneider. *OWL 2 Web Ontology Language: Manchester Syntax*. W3C Candidate Recommendation. World Wide Web Consortium (W3C), Oct. 27, 2009. URL: <http://www.w3.org/TR/2009/NOTE-owl2-manchester-syntax-20091027/> (cit. on p. 154).
- [HPSH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. “From SHIQ and RDF to OWL: The Making of a Web Ontology Language.” In: *Web Semantics 1.1* (2003), pp. 7–26 (cit. on pp. 61, 104).
- [HR03] Thérèse Hardin and Renaud Rioboo, eds. *11th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus 2003)*. Rome, Italy, Sept. 2003. URL: <http://www.calculemus.net/meetings/rome03/Proceedings/final.pdf> (cit. on pp. 472, 488).
- [HR09a] Peter Horn and Dan Roozmond. “OpenMath in SCIENCE: SCSCP and POP-CORN.” In: [CDSC+09], pp. 474–479. ISBN: 9783642026133 (cit. on p. 185).

- [HR09b] Peter Horn and Dan Roozemon, eds. *The Popcorn OpenMath Representation*. Version 1.0. Apr. 2009. URL: <http://java.symcomp.org/FormalPopcorn.html> (visited on 2009-11-12) (cit. on pp. 185, 380).
- [HRH+] Ian Horrocks, Alan Ruttenberg, Sandro Hawke, Ivan Herman, et al. *OWL Working Group*. URL: <http://www.w3.org/2007/OWL/> (visited on 2009-10-22) (cit. on p. 61).
- [HS10a] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language*. W3C Working Draft. World Wide Web Consortium (W3C), Oct. 14, 2010. URL: <http://www.w3.org/TR/2010/WD-sparql11-query-20101014/> (cit. on pp. 58, 228).
- [HS10b] Steve Harris and Andy Seaborne. *SPARQL 1.1 Query Language*. W3C Working Draft. World Wide Web Consortium (W3C), Oct. 14, 2010. URL: <http://www.w3.org/TR/2010/WD-sparql11-query-20101014/> (cit. on p. 271).
- [HSB+05] Dominik Heckmann, Tim Schwartz, Boris Brandherm, Michael Schmitz, and Margeritta von Wilamowitz-Moellendorff. “Gumo – The General User Model Ontology.” In: *User Modeling 2005*. Ed. by L. Ardissono, P. Brna, and A. Mitrovic. 2005, pp. 428–432. DOI: [http://dx.doi.org/10.1007/11527886\\_58](http://dx.doi.org/10.1007/11527886_58). URL: [http://dx.doi.org/10.1007/11527886\\_58](http://dx.doi.org/10.1007/11527886_58) (cit. on p. 130).
- [HV] Jónathan Heras Vicente. *An OpenMath Content Dictionary Editor*. URL: <http://www.unirioja.es/cu/joheras/openmath-editor.html> (visited on 2009-10-18) (cit. on p. 187).
- [HY07] Michael Hausenblas and Wing C Yung. *RDFa Test Suite*. W3C Editor’s Draft. World Wide Web Consortium (W3C), June 29, 2007. URL: <http://www.w3.org/2006/07/SWD/RDFa/testsuite/> (cit. on p. 391).
- [Iano2] Renato Iannella. *Open Digital Rights Language (ODRL)*. W3C Note. Version 1.1. World Wide Web Consortium (W3C), Sept. 19, 2002. URL: <http://www.w3.org/TR/2002/NOTE-odrl-20020919/> (cit. on p. 47).
- [IL10] Toby A. Inkster and Christoph Lange. *RDFa Host Languages*. Feb. 23, 2010. URL: [http://rdfa.info/wiki/?title=RDFa\\_Host\\_Languages&oldid=1032](http://rdfa.info/wiki/?title=RDFa_Host_Languages&oldid=1032) (visited on 2010-08-27) (cit. on p. 58).
- [Ink] Toby A. Inkster. *Swignition*. URL: <http://buzzword.org.uk/swignition/> (visited on 2009-10-22) (cit. on p. 269).
- [Iso] *Ergonomic Requirements for Office Work With Visual Display Terminals (VDT). Part 11: Guidance in Usability*. ISO 9241–11. London, 1997 (cit. on p. 315).
- [Jano6] Peter Jansen. “An Emacs mode for editing OMDoc Documents.” In: [Koh06b]. Chap. 26.16. URL: <http://omdoc.org/pubs/omdoc1.2.pdf> (cit. on pp. 24, 185).
- [Jen] *Jena — a Semantic Web Framework for Java*. URL: <http://jena.sf.net> (visited on 2010-05-10) (cit. on pp. 394, 396).
- [JHY+10] Prateek Jain, Pascal Hitzler, Peter Z. Yeh, Kunal Verma, and Amit P. Sheh. “Linked Data Is Merely More Data.” In: [BCH+10] (cit. on pp. 20, 94).

- [Jip] Peter Jipsen. *ASciencePad – a TiddlyWiki suitable for scientific notes*. URL: <http://math.chapman.edu/~jipsen/asciencepad/asciencepad.html> (visited on 2009-11-10) (cit. on pp. 195, 303).
- [JIVE+10] María José Ibáñez, Gabriela Vulcu, Joaquin Ezpeleta, and Sami Bhiri. “Semantically Enabled Business Process Discovery.” In: *Symposium on Applied Computing*. (Sierre, Switzerland, 2010). ACM, 2010, pp. 1396–1403. ISBN: 978-1-60558-639-7 (cit. on p. 272).
- [JK10] Constantin Jucovski and Michael Kohlhase. “sTeXIDE: An Integrated Development Environment for sTeX Collections.” In: [ACD+10], pp. 336–344. ISBN: 3642141277. arXiv:1005.5489v1 [cs.OH] (cit. on p. 185).
- [Job] *JOBAD Framework – JavaScript API for OMDoc-based active documents*. URL: <http://jomdoc.omdoc.org/wiki/JOBAD> (visited on 2010-08-05) (cit. on p. 258).
- [Jom] *JOMDoc Project — Java Library for OMDoc documents*. URL: <http://jomdoc.omdoc.org> (visited on 2011-05-07) (cit. on pp. 41, 102, 114, 199, 220, 274, 380, 395).
- [JWo4] Ian Jacobs and Norman Walsh. *Architecture of the World Wide Web*. W3C Recommendation. World Wide Web Consortium (W3C), Dec. 15, 2004. URL: <http://www.w3.org/TR/2004/REC-webarch-20041215/> (cit. on p. 8).
- [KAo8] C. Maria Keet and Alessandro Artale. “Representing and Reasoning over a Taxonomy of Part-Whole Relations.” In: *Applied Ontology* 3.1–2 (2008): *Special Issue on Ontological Foundations for Conceptual Modelling*, pp. 91–110 (cit. on p. 112).
- [KAJ+08] Michael Kohlhase, Ștefan Anca, Constantin Jucovski, Alberto González Palomo, and Ioan A. Șucan. “MathWebSearch 0.4, A Semantic Search Engine for Mathematics.” manuscript. 2008. URL: <http://mathweb.org/projects/mws/pubs/mkm08.pdf> (cit. on pp. 224, 379).
- [Kay07] Michael Kay. *XSL Transformations (XSLT) Version 2.0*. W3C Recommendation. World Wide Web Consortium (W3C), Jan. 23, 2007. URL: <http://www.w3.org/TR/2007/REC-xslt20-20070123/> (cit. on pp. 54, 363, 364, 391).
- [Kay08] Michael Kay. *Saxonica: XSLT and XQuery Processing*. 2008. (Visited on 2009-10-22) (cit. on p. 391).
- [Kay10] Michael Kay. *XSL Transformations (XSLT) Version 3.0*. W3C Working Draft. possibly still accessible via <http://www.w3.org/TR/xslt-21/>. World Wide Web Consortium (W3C), Dec. 14, 2010. URL: <http://www.w3.org/TR/2010/WD-xslt-30-20101214/> (cit. on p. 391).
- [KBT07] C. Kiefer, A. Bernstein, and J. Tappolet. “Analyzing Software with iSPARQL.” In: *Proc. 3rd International Workshop on Semantic Web Enabled Software Engineering (SWESE ’07)*. 2007 (cit. on p. 135).
- [Kep04] Stephan Kepser. “A Simple Proof of the Turing-Completeness of XSLT and XQuery.” In: *Extreme Markup Languages*. Ed. by Tommie Usdin. 2004 (cit. on pp. 72, 228, 392).

- [Ker10] Manfred Kerber. “Proofs, proofs, proofs, and proofs.” In: [ACD+10], pp. 345–354. ISBN: 3642141277. arXiv:1005.5124v1 [cs.AI] (cit. on pp. 4, 6).
- [KGL+09] Michael Kohlhase, Jana Giceva, Christoph Lange, and Vyacheslav Zholudev. “JOBAD – Interactive Mathematical Documents.” In: *AI Mashup Challenge at KI Conference*. Ed. by Brigitte Endres-Niggemeyer, Valentin Zacharias, and Pascal Hitzler. Sept. 2009. URL: <https://svn.omdoc.org/repos/jomdoc/doc/pubs/ai-mashup09/jobad.pdf> (cit. on p. 260).
- [KKo8] Andrea Kohlhase and Michael Kohlhase. “Semantic Knowledge Management for Education.” In: *Proceedings of the IEEE; Special Issue on Educational Technology* 96.6 (June 2008), pp. 970–989. URL: <http://kwarc.info/kohlhase/papers/sekm4ed.pdf> (cit. on pp. 37, 97).
- [KKo9a] Andrea Kohlhase and Michael Kohlhase. “Semantic Transparency in User Assistance Systems.” In: *Proceedings of the 27th annual ACM international conference on Design of communication (SIGDOC)*. (Bloomington, Indiana, USA, 2009). Ed. by Brad Mehlenbacher, Aristidis Protopsaltis, Ashley Williams, and Shaun Slaterey. ACM Special Interest Group for Design of Communication. New York, NY, USA: ACM Press, 2009, pp. 89–96. DOI: 10.1145/1621995.1622013. URL: <http://kwarc.info/kohlhase/papers/sigdoc09-semtrans.pdf> (cit. on pp. 340, 477).
- [KKo9b] Andrea Kohlhase and Michael Kohlhase. “Semantic Transparency in User Assistance Systems.” In: [KKo9a], pp. 89–96. DOI: 10.1145/1621995.1622013. URL: <http://kwarc.info/kohlhase/papers/sigdoc09-semtrans.pdf> (cit. on p. 343).
- [KKo9c] Andrea Kohlhase and Michael Kohlhase. “Spreadsheet Interaction with Frames: Exploring a Mathematical Practice.” In: [CDSC+09], pp. 341–356. ISBN: 9783642026133. URL: <http://kwarc.info/kohlhase/papers/mkm09-framing.pdf> (cit. on pp. 41, 341, 343).
- [KKK+08] Takayuki Kawata, Masaaki Kataoka, Hiroshi Kai, and Yasushi Tamura. “A MathML content markup editor on the xfy.” In: *Applications for Computer Algebra*. (RISC, Linz, Austria, July 27–30, 2008). Ed. by Elena Smirnova and Stephen M. Watt. 2008 (cit. on p. 317).
- [KKL10a] Andrea Kohlhase, Michael Kohlhase, and Christoph Lange. “Dimensions of Formality: A Case Study for MKM in Software Engineering.” In: [ACD+10], pp. 355–369. ISBN: 3642141277. arXiv:1004.5071v1 [cs.DL] (cit. on pp. 37, 47, 48, 97, 161, 181, 182, 237).
- [KKL10b] Andrea Kohlhase, Michael Kohlhase, and Christoph Lange. “sTeX – A System for Flexible Formalization of Linked Data.” In: [PHP+10]. ISBN: 978-1-4503-0014-8. DOI: 10.1145/1839707.1839712. arXiv:1006.4474v1 [cs.SE]. URL: <http://portal.acm.org/citation.cfm?id=1839707> (cit. on pp. 48, 161, 187, 198).



- [KKM+07] Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, eds. *MKM/Calculamus*. LNAI 4573. Springer Verlag, 2007. ISBN: 978-3-540-73083-5 (cit. on pp. 453, 462, 472, 478, 487).
- [KL] Yaron Koren and Niklas Laxström. *MediaWiki – Replace Text*. URL: [http://www.mediawiki.org/w/index.php?title=Extension:Replace\\_Text&oldid=354868](http://www.mediawiki.org/w/index.php?title=Extension:Replace_Text&oldid=354868) (visited on 2010-10-10) (cit. on p. 309).
- [KLM+08] Oliver Kutz, Dominik Lücke, Till Mossakowski, and Immanuel Normann. “The OWL in the CASL – Designing Ontologies Across Logics.” In: [SDRo8] (cit. on pp. 69, 158).
- [KLM+09] Michael Kohlhase, Christoph Lange, Christine Müller, Normen Müller, and Florian Rabe. *Notations for Active Mathematical Documents*. KWARC Report 2009-1. Jacobs University Bremen, Feb. 2009. URL: [http://kwarc.info/publications/papers/KLMR\\_NfAD.pdf](http://kwarc.info/publications/papers/KLMR_NfAD.pdf) (cit. on pp. 74, 97, 193, 217, 237).
- [KLR07] Michael Kohlhase, Christoph Lange, and Florian Rabe. “Presenting Mathematical Content With Flexible Elisions.” In: *OpenMath/JEM Workshop 2007*. Ed. by Olga Caprotti, Michael Kohlhase, and Paul Libbrecht. June 2007. URL: <http://www.openmath.org/meetings/linz2007/> (cit. on pp. 74, 97).
- [KLS+10] Andriy Kovalchuk, Vyacheslav Levitsky, Igor Samolyuk, and Valentyn Yanchuk. “The Formulator MathML Editor Project: User-Friendly Authoring of Content Markup Documents.” In: [ACD+10], pp. 385–397. ISBN: 3642141277. arXiv:1005.0146v1 [cs.DL] (cit. on p. 317).
- [KLW95] Michael Kifer, Georg Lausen, and James Wu. “Logical Foundations of Object-Oriented and Frame-Based Languages.” In: *Journal of the ACM* 42.4 (1995), pp. 741–843 (cit. on pp. 62, 148).
- [KMH08] Chrysovalanto Kousetti, David E. Millard, and Yvonne Howard. “A Study of Ontology Convergence in a Semantic Wiki.” In: *Proceedings of the 4th International Symposium on Wikis (WikiSym)*. (Porto, Portugal, Sept. 8–10, 2008). Ed. by Ademar Aguiar and Mark Bernstein. ACM Press, 2008. URL: <http://www.wikisym.org/ws2008/proceedings/> (cit. on p. 316).
- [KMR] Michael Kohlhase, Till Mossakowski, and Florian Rabe. *LATIN: Logic Atlas and Integrator*. URL: <http://latin.omdoc.org> (visited on 2010-09-15) (cit. on pp. 150, 248, 258, 271, 273, 330, 355, 388).
- [KMR+07] Fairouz Kamareddine, Manuel Maarek, Krzysztof Retel, and J. B. Wells. “Narrative Structure of Mathematical Texts.” In: [KKM+07], pp. 296–312. ISBN: 978-3-540-73083-5 (cit. on pp. 43, 204).
- [KMR08] Michael Kohlhase, Christine Müller, and Florian Rabe. “Notations for Living Mathematical Documents.” In: [ACR+08], pp. 504–519. URL: <http://omdoc.org/pubs/mkm08-notations.pdf> (cit. on pp. 71, 72, 220, 381).
- [Knu92] Donald E. Knuth. *Literate Programming*. The University of Chicago Press, 1992 (cit. on p. 11).

- [Koh+a] Michael Kohlhase et al. *Lectures as Linked Data*. URL: <http://kwarc.info/LinkedLectures/> (visited on 2010-09-22) (cit. on p. 212).
- [Koh+b] Michael Kohlhase et al. *Planet GenCS*. URL: <http://gencs.kwarc.info> (visited on 2010-09-22) (cit. on p. 259).
- [Koho6a] Michael Kohlhase, ed. *Mathematical Knowledge Management, MKM'05*. LNAI 3863. Springer Verlag, 2006 (cit. on pp. 460, 461, 487, 498).
- [Koho6b] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf> (cit. on pp. 37–39, 45, 67, 69–71, 73, 88, 97, 101, 102, 106, 109, 112, 113, 120, 124, 128, 161, 162, 164, 198, 210, 251, 273, 363, 381, 453, 468, 472, 475, 479, 488).
- [Koho6c] Michael Kohlhase. “Standardizing Context in System Interoperability.” In: [Koho6b]. Chap. 26.18. URL: <http://omdoc.org/pubs/omdoc1.2.pdf> (cit. on pp. 69, 200).
- [Koho8a] Andrea Kohlhase. “Semantic Interaction Design: Composing Knowledge with CPoint.” PhD thesis. Computer Science, Universität Bremen, Apr. 2008. URL: [http://kwarc.info/ako/pubs/AKo\\_Promo.pdf](http://kwarc.info/ako/pubs/AKo_Promo.pdf) (cit. on pp. 21, 23, 306).
- [Koho8b] Michael Kohlhase. “Compiling OpenMath Type systems to Relax NG Grammars.” In: *3rd JEM Workshop – Joining Educational Mathematics*. Ed. by Olga Caprotti, Sebastian Xambó, Maria-Antonia Huertas, Michael Kohlhase, and Mika Seppälä. 2008. URL: <http://jem-thematic.net/workshop3> (cit. on p. 200).
- [Koho8c] Michael Kohlhase. *Generic Metadata Element*. e-mail to [project-omdoc-dev@jacobs-university.de](mailto:project-omdoc-dev@jacobs-university.de). July 1, 2008. URL: <http://lists.jacobs-university.de/mailman/private/project-omdoc-dev/2008-July/thread.html#73> (cit. on p. 161).
- [Koho8d] Michael Kohlhase. “Using L<sup>A</sup>T<sub>E</sub>X as a Semantic Markup Format.” In: *Mathematics in Computer Science 2.2* (2008), pp. 279–304. URL: <https://svn.kwarc.info/repos/stex/doc/mcs08/stex.pdf> (cit. on p. 76).
- [Koh10a] Michael Kohlhase. *dcm.sty: An Infrastructure for marking up Dublin Core Metadata in L<sup>A</sup>T<sub>E</sub>X documents*. Self-documenting L<sup>A</sup>T<sub>E</sub>X package. Comprehensive T<sub>E</sub>X Archive Network (CTAN), 2010. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/dcm/dcm.pdf> (cit. on p. 187).
- [Koh10b] Michael Kohlhase. *OAF: FlexiForms*. July 23, 2010. URL: <http://trac.kwarc.info/oaf/wiki/FlexiForms?version=5> (visited on 2010-08-11) (cit. on p. 35).
- [Koh10c] Michael Kohlhase. *reqdoc.sty: Semantic Markup for Requirements Specification Documents*. Self-documenting L<sup>A</sup>T<sub>E</sub>X package. Comprehensive T<sub>E</sub>X Archive Network (CTAN), 2010. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/reqdoc/reqdoc.pdf> (cit. on p. 187).



- [Koh10d] Michael Kohlhase. *sproof.sty: Structural Markup for Proofs*. Self-documenting L<sup>A</sup>T<sub>E</sub>X package. Comprehensive T<sub>E</sub>X Archive Network (CTAN), 2010. URL: <http://www.ctan.org/tex-archive/macros/latex/contrib/stex/sproof/sproof.pdf> (cit. on p. 77).
- [Kor+10] Yaron Koren et al. *MediaWiki – Semantic Forms*. Aug. 26, 2010. URL: [http://www.mediawiki.org/w/index.php?title=Extension:Semantic\\_Forms&oldid=347072](http://www.mediawiki.org/w/index.php?title=Extension:Semantic_Forms&oldid=347072) (cit. on p. 187).
- [Kor10] Yaron Koren. *MediaWiki – Semantic Internal Objects*. Sept. 26, 2010. URL: [http://www.mediawiki.org/w/index.php?title=Extension:Semantic\\_Internal\\_Objects&oldid=352992](http://www.mediawiki.org/w/index.php?title=Extension:Semantic_Internal_Objects&oldid=352992) (cit. on p. 283).
- [KPH+08] Dionysios D. Kehagias, Ioannis Papadimitriou, Joana Hois, Dimitrios Tzovaras, and John Bateman. “A Methodological Approach for Ontology Evaluation and Refinement.” In: *ASK-IT Final Conference*. June 2008 (cit. on p. 147).
- [KPS+06] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca Grau, and James A. Hendler. “Swoop: A Web Ontology Editing Browser.” In: *Web Semantics 4.2* (2006), pp. 144–153 (cit. on p. 363).
- [KPV+09] Konstantinos Kotis, Andreas Papasalourous, George A. Vouros, Pappas Nikolaos, and Zoumpatianos Konstantinos. “e-Class in Ontology Engineering: Integrating Ontologies to Argumentation and Semantic Wiki technology.” In: *Workshop on Intelligent and Innovative Support for Collaborative Learning Activities (WIISCOLA)*. (University of the Aegean, Rhodes, Greece, June 8–13, 2009). Ed. by Jacqueline Bourdeau, Riichiro Mizoguchi, Seiji Isotani, Barbara Wasson, WeiQin Chen, and Jelena Jovanovic. 2009, pp. 9–18 (cit. on p. 304).
- [KR09] Michael Kohlhase and Florian Rabe. “Semantics of OpenMath and MathML<sub>3</sub>.” In: [Dav09]. URL: <http://kwarc.info/kohlhase/papers/om09-semantic.pdf> (cit. on pp. 64, 67, 91, 118).
- [KR70] Werner Kunz and Horst W. J. Rittel. *Issues as elements of information systems*. Working paper 131. Institute of Urban and Regional Development, University of California, Berkeley, July 1970 (cit. on pp. 49, 132).
- [Kro03] Aaron Krowne. “An Architecture for Collaborative Math and Science Digital Libraries.” MA thesis. Virginia Tech, 2003. URL: <http://scholar.lib.vt.edu/theses/available/etd-09022003-150851/> (cit. on p. 301).
- [Kru09] Sebastian Ryszard Kruk. “Semantic Digital Libraries. Improving Usability of Information Discovery with Semantic and Social Services.” PhD thesis. National University of Ireland, Galway, 2009 (cit. on pp. 314, 316).
- [KRZ10] Michael Kohlhase, Florian Rabe, and Vyacheslav Zholudev. “Towards MKM in the Large: Modular Representation and Scalable Software Architecture.” In: [ACD+10], pp. 370–384. ISBN: 3642141277. arXiv:1005.5232v2 [cs.OH] (cit. on pp. 225, 226).

- [KSP+07] Aniket Kittur, Bongwon Suh, Bryan A. Pendleton, and Ed H. Chi. “He says, she says: conflict and coordination in Wikipedia.” In: *CHI*. Ed. by Mary Beth Rosson and David J. Gilmore. ACM, 2007, pp. 453–462. ISBN: 978-1-59593-593-9 (cit. on p. 285).
- [KSV07] Markus Krötzsch, Sebastian Schaffert, and Denny Vrandečić. “Reasoning in Semantic Wikis.” In: *Proceedings of the 3rd Reasoning Web Summer School*. (Dresden, Germany, Sept. 3–7, 2007). Ed. by Grigoris Antoniou, Uwe Aßmann, Cristina Baroglio, Stefan Decker, Nicola Henze, Paula-Lavinia Pătrânjan, and Robert Tolksdorf. LNCS 4636. Springer, 2007, pp. 310–329. URL: [http://korrekt.org/talks/2007/Kroetzsch\\_Reasoning\\_Web\\_Semantic\\_Wikis\\_2007.pdf](http://korrekt.org/talks/2007/Kroetzsch_Reasoning_Web_Semantic_Wikis_2007.pdf) (cit. on p. 284).
- [KU08] Ralf Klischewski and Stefan Ukena. “Test Strategies for Evaluation of Semantic eGovernment Applications.” In: *Electronic Government (EGOV)*. Ed. by M. A. Wimmer, H. J. Scholl, and E. Ferro. LNCS 5184. Springer Verlag, 2008, pp. 291–302 (cit. on p. 324).
- [Kuh09] Tobias Kuhn. “How Controlled English can Improve Semantic Wikis.” In: [LSSM+09]. URL: <http://ceur-ws.org/Vol-464/> (cit. on p. 317).
- [Kuro9] Siarhei Kuryla. *OMDoc as an Ontology Language: OWL→OMDoc translation implementation*. Project report. 2009. URL: <https://svn.kwarc.info/repos/kwarc/rabe/Teaching/Seminar09/Reports/siarhei.pdf> (cit. on pp. 274, 279, 394).
- [KWZo8] Fairouz Kamareddine, J. B. Wells, and Christoph Zengler. “Computerising Mathematical Text with MathLang.” In: *Electron. Notes Theor. Comput. Sci.* 205 (2008), pp. 5–30. ISSN: 1571-0661. DOI: <http://dx.doi.org/10.1016/j.entcs.2008.03.063>. URL: <http://www.cedar-forest.org/forest/papers/drafts/mathlang-coq-short.pdf> (cit. on pp. 22, 75).
- [KŞ06] Michael Kohlhase and Ioan Şucan. “A Search Engine for Mathematical Formulae.” In: *Proceedings of Artificial Intelligence and Symbolic Computation, AISC’2006*. Ed. by Tetsuo Ida, Jacques Calmet, and Dongming Wang. LNAI 4120. Springer Verlag, 2006, pp. 241–253. URL: <http://kwarc.info/kohlhase/papers/aisc06.pdf> (cit. on p. 224).
- [Lak76] Imre Lakatos. *Proofs and Refutations. The Logic of Mathematical Discovery*. Cambridge University Press, 1976 (cit. on pp. 5, 176).
- [Lan] Christoph Lange. *OpenMath Wiki*. URL: <http://wiki.openmath.org> (visited on 2009-10-22) (cit. on p. 221).
- [Lan+] Christoph Lange et al. *Krextor – The KWARC RDF Extractor*. URL: <http://kwarc.info/projects/krextor/> (visited on 2010-12-06) (cit. on p. 390).
- [Lano6a] Christoph Lange. “A Semantic Wiki for Mathematical Knowledge Management.” Diploma thesis. Universität Trier, Aug. 2006. URL: <http://kwarc.info/projects/swim/pubs/swim-thesis-final.pdf> (cit. on pp. 311, 482).
- [Lano6b] Christoph Lange, ed. *Wikis und Blogs – Planen, Einrichten, Verwalten*. C&L Computer- und Literaturverlag, Sept. 2006. ISBN: 3-936546-44-4 (cit. on pp. 281, 282).

- [Lano7a] Christoph Lange. *SWiM – A Semantic Wiki for Mathematical Knowledge Management*. Tech. rep. 5. Revised, updated and reviewed version of [Lano6a]. Jacobs University Bremen, Mar. 2007. URL: <http://jpubs.jacobs-university.de/handle/579/143> (cit. on pp. 287, 311, 381).
- [Lano7b] Christoph Lange. “Towards Scientific Collaboration in a Semantic Wiki.” In: *Bridging the Gap between Semantic Web and Web 2.0 (SemNet)*. Ed. by Andreas Hotho and Bettina Hoser. June 2007 (cit. on p. 310).
- [Lano8a] Christoph Lange. “Mathematical Semantic Markup in a Wiki: The Roles of Symbols and Notations.” In: [LSSM+08]. URL: <http://ceur-ws.org/Vol-360/> (cit. on p. 311).
- [Lano8b] Christoph Lange. *SWiM Subversion client*. Sept. 24, 2008. URL: [http://mathweb.org/w/index.php?title=SWiM/Subversion\\_client&oldid=8284](http://mathweb.org/w/index.php?title=SWiM/Subversion_client&oldid=8284) (visited on 2010-10-06) (cit. on p. 398).
- [Lano9a] Christoph Lange. “Krextor – An Extensible XML→RDF Extraction Framework.” In: [BAG09]. URL: <http://ceur-ws.org/Vol-449/ShortPaper2.pdf> (cit. on pp. 272, 279).
- [Lano9b] Christoph Lange. “wiki.openmath.org – how it works, how you can participate.” In: [Dav09]. arXiv:1003.5192v1 [cs.DL]. URL: <http://staff.bath.ac.uk/masjhd/OM2009.html> (cit. on p. 311).
- [Lan10] Christoph Lange. “Towards OpenMath Content Dictionaries as Linked Data.” In: *23rd OpenMath Workshop*. Ed. by Michael Kohlhase and Christoph Lange. July 2010. arXiv:1006.4057v1 [cs.DL]. URL: <http://cicm2010.cnam.fr/om/> (cit. on pp. 20, 210, 355).
- [LBo8] Charlene Li and Josh Bernoff. *Groundswell. Winning in a World Transformed by Social Technologies*. Harvard Business Press, 2008 (cit. on p. 23).
- [LBG+08] Christoph Lange, Uldis Bojārs, Tudor Groza, John Breslin, and Siegfried Handschuh. “Expressing Argumentative Discussions in Social Media Sites.” In: *Social Data on the Web (SDoW), Workshop at the 7th International Semantic Web Conference*. (Karlsruhe, Germany, Oct. 27, 2008). Ed. by John Breslin, Uldis Bojārs, Alexandre Passant, and Sergio Fernández. CEUR Workshop Proceedings 405. Aachen, 2008. URL: <http://ceur-ws.org/Vol-405/paper4.pdf> (cit. on pp. 97, 131, 132, 144, 237).
- [LGPo8] Christoph Lange and Alberto González Palomo. “Easily Editing and Browsing Complex OpenMath Markup with SWiM.” In: *Mathematical User Interfaces Workshop*. Ed. by Paul Libbrecht. July 2008. URL: <http://www.activemath.org/workshops/MathUI/08/proceedings/LangeGonzales-OMEdit.html> (cit. on pp. 191, 192, 237).

- [LHCo8] Christoph Lange, Tuukka Hastrup, and Stéphane Corlosquet. “Arguing on Issues with Mathematical Knowledge Items in a Semantic Wiki.” In: *Wissens- und Erfahrungsmanagement LWA (Lernen, Wissensentdeckung und Adaptivität) Conference Proceedings*. Ed. by Joachim Baumeister and Martin Atzmüller. Vol. 448. Oct. 2008 (cit. on pp. 97, 237, 311).
- [Libo8] Paul Libbrecht. “Cross curriculum search through the GeoSkills Ontology.” In: *Proceedings of the 2nd International Workshop on Search and Exchange of e-learning Material (SE@M), located at EC-TEL-2008*. (Maastricht, Netherlands, Sept. 17, 2008). Ed. by D. Massart, J.-N. Colin, F. Van Asche, and M. Wolpers. CEUR Workshop Proceedings. Aachen: CEUR-WS.org, 2008. URL: <http://ceur-ws.org/Vol-385/> (cit. on p. 130).
- [Libo9] Paul Libbrecht. *Collection Management in ActiveMath*. presented without publication at [CDSC+09]. 2009. URL: [http://www.activemath.org/~paul/copy\\_left/Content-Storage-and-Patterns.pdf](http://www.activemath.org/~paul/copy_left/Content-Storage-and-Patterns.pdf) (visited on 2009-11-18) (cit. on pp. 44, 45, 71, 124).
- [Lib10a] Paul Libbrecht. “Notations Around the World: Census and Exploitation.” In: [ACD+10], pp. 398–410. ISBN: 3642141277. arXiv:1004.5165v1 [cs.DL] (cit. on pp. 40, 355).
- [Lib10b] Paul Libbrecht. “What You Check is What You Get: Authoring with jEditOQMath.” In: *10th International Conference on Advanced Learning Technologies (ICALT)*. (Sousse, Tunisia, July 5–7, 2010). IEEE, 2010, pp. 682–686 (cit. on pp. 24, 185, 301).
- [LKo8] Christoph Lange and Michael Kohlhase. “A Semantic Wiki for Mathematical Knowledge Management.” In: *Emerging Technologies for Semantic Work Environments: Techniques, Methods, and Applications*. Ed. by Jörg Rech, Björn Decker, and Eric Ras. IGI Global, Apr. 2008, pp. 47–68. URL: <http://www.igi-global.com/Bookstore/Chapter.aspx?TitleId=10143> (cit. on pp. 308, 351).
- [LKo9] Christoph Lange and Michael Kohlhase. “A Mathematical Approach to Ontology Authoring and Documentation.” In: [CDSC+09], pp. 389–404. ISBN: 9783642026133. URL: <http://kwarc.info/kohlhase/papers/mkm09-omdoc4onto.pdf> (cit. on pp. 97, 160, 169, 237).
- [LMo6] Paul Libbrecht and Erica Melis. “Methods for Access and Retrieval of Mathematical Content in ActiveMath.” In: *Proceedings of ICMS-2006*. Ed. by N. Takayama and A. Iglesias. LNAI 4151. Springer Verlag, 2006. URL: <http://www.activemath.org/publications/Libbrecht-Melis-Access-and-Retrieval-ActiveMath-ICMS-2006.pdf> (cit. on p. 224).
- [LMK+09] Roberto A. Lotufo, Rubens C. Machado, André Körbes, and Rafael G. Ramos. “Adessowiki On-line Collaborative Scientific Programming Platform.” In: *Proceedings of the 5th International Symposium on Wikis and Open Collaboration (WikiSym)*. (Orlando, Florida, Oct. 25–27, 2009). Ed. by Dirk Riehle and Amy Bruckman. ACM Press, 2009. URL: <http://www.wikisym.org/ws2009/proceedings/> (cit. on p. 353).

- [LMRo8] Christoph Lange, Sean McLaughlin, and Florian Rabe. “Flyspeck in a Semantic Wiki – Collaborating on a Large Scale Formalization of the Kepler Conjecture.” In: [LSSM+o8]. URL: <http://ceur-ws.org/Vol-360/> (cit. on pp. 181, 237, 288, 311).
- [LMY+o4] Shengping Liu, Jing Mei, Anbu Yue, and Zuoquan Lin. “XSDL: Making XML Semantics Explicit.” In: *SWDB*. (Toronto, Canada, Aug. 29–30, 2004). Ed. by Christoph Bussler, Val Tannen, and Iрин Fundulaki. LNCS 3372. Springer Verlag, 2004, pp. 64–83. ISBN: 3-540-24576-6 (cit. on pp. 140, 269).
- [LPPH+o9] Danh Le-Phuoc, Axel Polleres, Manfred Hauswirth, Giovanni Tummarello, and Christian Morbidoni. “Rapid Prototyping of Semantic Mash-Ups through Semantic Web Pipes.” In: *Proceedings of the 17th WWW conference*. (Madrid, Spain, Apr. 20–24, 2009). Ed. by Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl. ACM Press, 2009, pp. 581–590. ISBN: 978-1-60558-487-4 (cit. on pp. 241, 272).
- [LSo9] James R. Lewis and Jeff Sauro. “The Factor Structure of the System Usability Scale.” In: *Proceedings of the Human Computer Interaction International Conference (HCII)*. (San Diego, CA, USA, 2009). 2009 (cit. on p. 342).
- [LSD+o6] Yuan Fang Li, Jing Sun, Gillian Dobbie, Jun Sun, and Hai Wang. “Validating Semistructured Data using OWL.” In: *7th International Conference on Web-Age Information Management (WAIM’06)*. LNCS 4016. Springer Verlag, June 2006, pp. 520–531 (cit. on p. 204).
- [LSSM+o8] Christoph Lange, Sebastian Schaffert, Hala Skaf-Molli, and Max Völkel, eds. *Proceedings of the 3rd Workshop on Semantic Wikis, European Semantic Web Conference*. (Costa Adeje, Tenerife, Spain, June 2, 2008). CEUR Workshop Proceedings 360. Aachen, 2008. URL: <http://ceur-ws.org/Vol-360/> (cit. on pp. 482, 484).
- [LSSM+o9] Christoph Lange, Sebastian Schaffert, Hala Skaf-Molli, and Max Völkel, eds. *Proceedings of the 4th Workshop on Semantic Wikis, European Semantic Web Conference*. (Hersonissos, Crete, Greece, June 1, 2009). CEUR Workshop Proceedings 464. Aachen, 2009. URL: <http://ceur-ws.org/Vol-464/> (cit. on pp. 457, 464, 481, 491, 496, 498).
- [Lur] Lurch. *open-source mathematics validation software*. URL: <http://lurch.sourceforge.net> (visited on 2010-09-09) (cit. on pp. 65, 301).
- [LZ10] Christoph Lange and Vyacheslav Zholudev. “Previewing OWL Changes and Refactorings Using a Flexible XML Database.” In: [dGCL+10]. URL: <http://ceur-ws.org/Vol-596/paper-08.pdf> (cit. on pp. 225, 271).
- [MAB+o1] E. Melis, E. Andrés, J. Büdenbender, Adrian Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. “ActiveMath: A generic and adaptive web-base learning environment.” In: *International Journal of Artificial Intelligence in Education* 12.4 (2001), pp. 385–407 (cit. on p. 48).

- [Mad+] Stewart Mader et al. *Seed it with content*. URL: <http://www.wikipatterns.org/display/wikipatterns/Seed+it+with+content> (visited on 2010-10-20) (cit. on p. 319).
- [Mado7] Stewart Mader. *wikipatterns*. Wiley, Dec. 2007. ISBN: 978-0470223628 (cit. on p. 197).
- [Mad92] Nazim H. Madhavji. “Environment Evolution: The Prism Model of Changes.” In: *IEEE Transactions on Software Engineering* 18.5 (May 1992), pp. 380–392 (cit. on pp. 41, 43).
- [MAF+03] E. Melis, J. Buedenbender E. Andres, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. “Knowledge Representation and Management in ACTIVE-MATH.” In: *International Journal on Artificial Intelligence and Mathematics, Special Issue on Management of Mathematical Knowledge* 38.1–3 (2003), pp. 47–64 (cit. on p. 162).
- [Mar] *MARC code list for Relators, Sources, Description Conventions*. 2003. URL: <http://www.loc.gov/marc/relators> (visited on 2009-10-22) (cit. on p. 45).
- [Mar03] Massimo Marchiori. “The Mathematical Semantic Web.” In: [ABDo3] (cit. on pp. 16, 83).
- [Mata] *Math-Net RDF Collection*. URL: <http://www.iwi-iuk.org/material/RDF/1.1/> (visited on 2009-12-12) (cit. on p. 125).
- [Matb] *MathDox – Interactive Mathematics*. URL: <http://www.mathdox.org> (visited on 2010-01-23) (cit. on pp. 15, 18, 24, 80, 184, 186, 240).
- [Matc] *MathDox – OpenMath Translation Servlet*. URL: <http://mathdox.org/phrasebook/> (visited on 2010-03-16) (cit. on p. 390).
- [Matd] *MathML Software – Editors*. URL: [http://www.w3.org/Math/Software/mathml\\_software\\_cat\\_editors.html](http://www.w3.org/Math/Software/mathml_software_cat_editors.html) (visited on 2009-11-11) (cit. on p. 186).
- [Mate] *MathOverflow*. URL: <http://mathoverflow.net> (visited on 2010-09-13) (cit. on pp. 13, 49, 233).
- [MBG+03] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. *Ontology Library*. WonderWeb Deliverable 18. Laboratory for Applied Ontology – ISTC-CNR, Dec. 2003. URL: <http://www.loa-cnr.it/Papers/D18.pdf> (cit. on p. 62).
- [MCGH+09] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. *OWL 2 Web Ontology Language: Profiles*. W3C Recommendation. World Wide Web Consortium (W3C), Oct. 27, 2009. URL: <http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/> (cit. on pp. 61, 154).



- [MDS+07] Deborah L. McGuinness, Li Ding, Paulo Pinheiro da Silva, and Cynthia Chang. “PML 2: A Modular Explanation Interlingua.” In: *Proceedings of the AAAI Workshop on Explanation-Aware Computing (ExaCt)*. (Vancouver, British Columbia, Canada, July 22–23, 2007). Ed. by Thomas Roth-Berghofer, Stefan Schulz, and David B. Leake. 2007. URL: <http://www.aaai.org/Papers/Workshops/2007/WS-07-06/WS07-06-008.pdf> (cit. on p. 87).
- [MEC+06] Daniel Marquès, Ramon Eixarch, Glòria Casanellas, and Bruno Martínez. “WIRIS OM Tools: a Semantic Formula Editor.” In: *Mathematical User Interfaces Workshop 2006*. Ed. by Paul Libbrecht. 2006. URL: <http://www.activemath.org/~paul/MathUI06> (cit. on pp. 186, 301, 317).
- [Med] *MediaWiki*. URL: <http://www.mediawiki.org> (visited on 2009-10-22) (cit. on p. 285).
- [MEG+03] Luke McDowell, Oren Etzioni, Steven D. Gribble, Alon Halevy, Henry Levy, William Pentney, Deepak Verma, and Stani Vlasheva. “Mangrove: Enticing Ordinary People onto the Semantic Web via Instant Gratification.” In: *3rd International Semantic Web Conference (ISWC 2003)*. 2003 (cit. on p. 21).
- [Meta] *Dummy edit. From Meta, a Wikimedia project coordination wiki*. Dec. 23, 2008. URL: [http://meta.wikimedia.org/w/index.php?title=Help:Dummy\\_edit&diff=prev&oldid=1320951](http://meta.wikimedia.org/w/index.php?title=Help:Dummy_edit&diff=prev&oldid=1320951) (cit. on p. 293).
- [Metb] *Flagged Revisions. From Meta, a Wikimedia project coordination wiki*. June 26, 2010. URL: [http://meta.wikimedia.org/w/index.php?title=Flagged\\_Revisions&oldid=2020634](http://meta.wikimedia.org/w/index.php?title=Flagged_Revisions&oldid=2020634) (cit. on p. 201).
- [Metc] *Minor edit. From Meta, a Wikimedia project coordination wiki*. Sept. 25, 2010. URL: [http://meta.wikimedia.org/w/index.php?title=Help:Minor\\_edit&oldid=2132961](http://meta.wikimedia.org/w/index.php?title=Help:Minor_edit&oldid=2132961) (cit. on p. 177).
- [MG10] David McCabe and Andrew Garrett. *MediaWiki – LiquidThreads*. Sept. 29, 2010. URL: <http://www.mediawiki.org/w/index.php?title=Extension:LiquidThreads&oldid=353654> (cit. on p. 285).
- [MGH+06] Erica Melis, Giorgi Gogvadze, Martin Homik, Paul Libbrecht, Carsten Ullrich, and Stefan Winterstein. “Semantic-aware components and services of ActiveMath.” In: *British Journal of Educational Technology* 37.3 (May 2006), pp. 405–423 (cit. on p. 240).
- [MGL+09] Erica Melis, Giorgi Gogvadze, Paul Libbrecht, and Carsten Ullrich. “Culturally adapted mathematics education with ActiveMath.” In: *AI & Society* 24.3 (2009), pp. 251–265 (cit. on pp. 40, 41).
- [MH04] Deborah L. McGuinness and Frank van Harmelen. *OWL Web Ontology Language Overview*. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 10, 2004. URL: <http://www.w3.org/TR/2004/REC-owl-features-20040210/> (cit. on p. 148).
- [Mic] *Microformats*. URL: <http://microformats.org> (visited on 2010-02-01) (cit. on p. 60).

- [Mil] Bruce Miller. *LaTeXML: A L<sup>A</sup>T<sub>E</sub>X to XML Converter*. URL: <http://dlmf.nist.gov/LaTeXML/> (visited on 2011-03-03) (cit. on p. 77).
- [Mino5] Robert Miner. “The Importance of MathML to Mathematics Communication.” In: *Notices of the AMS* 52.5 (2005), pp. 532–538 (cit. on p. 224).
- [Mis10] Dimitar Misev. “Integrating SUMO and OMDoc.” Bachelor’s Thesis. Computer Science, Jacobs University, Bremen, 2010 (cit. on pp. 159, 274).
- [Miza] *Mizar Mathematical Library*. URL: <http://www.mizar.org/library> (visited on 2009-12-02) (cit. on p. 11).
- [Mizb] *Mizar System*. URL: <http://mizar.org/system/> (visited on 2010-07-31) (cit. on pp. 11, 81).
- [MLU+06] Shahid Manzoor, Paul Libbrecht, Carsten Ullrich, and Erica Melis. “Authoring Presentation for OPENMATH.” In: [Koho6a], pp. 33–48 (cit. on pp. 65, 71, 72, 179, 196).
- [MMo7] Robert Miner and Rajesh Munavalli. “An Approach to Mathematical Search Through Query Formulation and Data Normalization.” In: [KKM+07], pp. 342–355. ISBN: 978-3-540-73083-5 (cit. on p. 224).
- [MMLo7] Till Mossakowski, Christian Maeder, and Klaus Lüttich. “The Heterogeneous Tool Set.” In: *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS-2007*. Ed. by Orna Grumberg and Michael Huth. LNCS 4424. Berlin, Germany: Springer Verlag, 2007, pp. 519–522 (cit. on pp. 69, 93, 271).
- [Mod] *ModelDriven.org Architecture Ontology*. URL: <http://modeldriven.org/2008/ArchitectureOntology/doc/> (visited on 2010-08-27) (cit. on pp. 48, 124, 127).
- [Mona] *MKMNET (Mathematical Knowledge Management Network)*. URL: <http://monet.nag.co.uk/mkm/> (visited on 2009-10-22) (cit. on pp. 17, 125, 184).
- [Monb] *MONET – Mathematics on the net*. URL: <http://monet.nag.co.uk> (visited on 2010-07-15) (cit. on p. 17).
- [Mo009] Ross Moore. “Ongoing efforts to generate “tagged PDF” using pdfTeX.” In: *Towards Digital Mathematics Library, DML 2009 workshop*. Ed. by Petr Sojka. Masaryk University, Brno, 2009 (cit. on p. 223).
- [Mos] Till Mossakowski. *Hets: the Heterogeneous Tool Set*. URL: <http://www.dfki.de/sks/hets> (visited on 2011-08-05) (cit. on pp. 69, 93, 271).
- [MOV06] Jonathan Marsh, David Orchard, and Daniel Veillard. *XML Inclusions (XInclude) Version 1.0 (Second Edition)*. W3C Recommendation. World Wide Web Consortium (W3C), Nov. 15, 2006. URL: <http://www.w3.org/TR/2006/REC-xinclude-20061115/> (cit. on pp. 276, 382).
- [Mow] *MoWGLI (Mathematics on the Web – Get it by Logic and Interfaces)*. URL: <http://mowgli.cs.unibo.it/> (visited on 2010-07-15) (cit. on p. 16).



- [MPo3] Manuel Maarek and Virgile Prevosto. “FoCDoc: The Documentation System of FoC.” In: [HRo3], pp. 31–43. URL: <http://www.calculumus.net/meetings/rome03/Proceedings/final.pdf> (cit. on p. 82).
- [MPPSo9] Boris Motik, Bijan Parsia, and Peter F. Patel-Schneider. *OWL 2 Web Ontology Language: XML Serialization*. W3C Recommendation. World Wide Web Consortium (W3C), Oct. 27, 2009. URL: <http://www.w3.org/TR/2009/REC-owl2-xml-serialization-20091027/> (cit. on pp. 271, 272).
- [MPSPo9] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. W3C Recommendation. World Wide Web Consortium (W3C), Oct. 27, 2009. URL: <http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/> (cit. on pp. 61, 107, 139, 148, 150, 157, 202, 208, 271).
- [MRo8] Steve McKay and Jason Robbins. *Looks Good To Me – Source Code Review Tools*. July 30, 2008. URL: <http://googlecode.blogspot.com/2008/07/looks-good-to-me-source-code-review.html> (visited on 2009-10-27) (cit. on p. 231).
- [MRB] Arno Mittelbach, Sebastian Rahtz, and Ioan Bernevig. *Roma: generating validators for the TEI*. URL: <http://www.tei-c.org/Roma/> (visited on 2010-02-03) (cit. on p. 78).
- [MRRo3] Peter Murray-Rust and Henry S. Rzepa. “Chemical Markup, XML, and the World Wide Web. 4. CML Schema.” In: *Journal of Chemical Information and Computer Sciences* 43 (2003), pp. 757–772. DOI: [10.1021/ci0256541](https://doi.org/10.1021/ci0256541) (cit. on p. 351).
- [Msc] *Mathematics Subject Classification MSC2010*. 2010. URL: <http://msc2010.org> (visited on 2009-11-16) (cit. on pp. 11, 46).
- [MSZ+10] Paolo Missier, Satya S. Sahoo, Jun Zhao, Carole Goble, and Amit Sheth. “Janus: from Workflows to Semantic Provenance and Linked Open Data.” In: *Proceedings of the 3rd Provenance and Annotation Workshop*. (Troy, NY, USA, June 15–16, 2010). 2010 (cit. on p. 20).
- [MT] William C. Mann and Maite Taboada. *Rhetorical Structure Theory*. URL: <http://www.sfu.ca/rst/> (visited on 2010-07-24) (cit. on pp. 39, 122).
- [MVWo5] Jonathan Marsh, Daniel Veillard, and Norman Walsh. *xml:id Version 1.0*. W3C Recommendation. World Wide Web Consortium (W3C), Sept. 9, 2005. URL: <http://www.w3.org/TR/2005/REC-xml-id-20050909/> (cit. on pp. 54, 91, 137).
- [MWAo3] Erica Melis, Markus Weber, and Eric Andr  s. “Lessons for (Pedagogic) Usability of eLearning Systems.” In: *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*. (Chesapeake, VA, USA, 2003). Ed. by A. Rossett. AACE, 2003, pp. 281–284 (cit. on p. 317).
- [M  lo6] Normen M  ller. “OMDoc as a Data Format for VeriFun.” In: [Koho6b]. Chap. 26.20, pp. 329–332. URL: <http://omdoc.org/pubs/omdoc1.2.pdf> (cit. on p. 69).

- [Mül10a] Christine Müller. “Adaptation of Mathematical Documents.” PhD thesis. Jacobs University Bremen, 2010. URL: <http://kwarc.info/cmuller/papers/thesis.pdf> (cit. on pp. 40, 41, 49, 71, 72, 110, 217, 220, 239, 253, 381).
- [Mül10b] Normen Müller. “Change Management on Semi-Structured Documents.” PhD thesis. Jacobs University Bremen, 2010. URL: <http://kwarc.info/nmueller/papers/nrmphd.pdf> (cit. on pp. 41, 43, 184, 262, 308).
- [Nca] *The n-Category Café. A group blog on math, physics and philosophy.* URL: <http://golem.ph.utexas.edu/category/> (visited on 2010-07-14) (cit. on p. 12).
- [Neo] *NeOn Toolkit.* URL: <http://neon-toolkit.org> (visited on 2009-10-26) (cit. on pp. 149, 160, 303).
- [Nev10] Zuzana Nevěřilová. “Implementing Dynamic Visualization as an Alternative Interface to a Digital Mathematics Library.” In: *Towards Digital Mathematics Library, DML workshop*. Ed. by Petr Sojka. Masaryk University, Brno, 2010, pp. 63–68 (cit. on p. 209).
- [Nie00] Jakob Nielsen. *Why You Only Need to Test with 5 Users*. Mar. 19, 2000. URL: <http://www.useit.com/alertbox/20000319.html> (visited on 2010-04-18) (cit. on p. 337).
- [Nie08] Michael Nielsen. *The Future of Science*. July 17, 2008. URL: <http://michaelnielsen.org/blog/the-future-of-science-2/> (cit. on p. 7).
- [Nie93] Jakob Nielsen. *Usability Engineering*. London: Academic Press, 1993 (cit. on pp. 315, 324, 325, 430).
- [NJSSM+09] Hala Naja-Jazzar, Nishadi de Silva, Hala Skaf-Molli, Charbel Rahhal, and Pascal Molli. “OntoReST: A RST-based Ontology for Enhancing Documents Content Quality in Collaborative Writing.” In: *INFOCOMP Journal of Computer Science* 8.3 (Sept. 2009), pp. 1–10 (cit. on pp. 119, 120, 122).
- [Nla] *nLab*. URL: <http://ncatlab.org/> (visited on 2010-07-14) (cit. on p. 12).
- [Noro8] Immanuel Normann. “Automated Theory Interpretation.” PhD thesis. Bremen, Germany: Jacobs University, 2008. URL: <https://svn.eecs.jacobs-university.de/svn/eecs/archive/phd-2008/normann.pdf> (cit. on p. 230).
- [Noro9] Bryce Nordgren. *Workspace to develop a math domain for DITA*. Feb. 21, 2009. URL: <http://dita.xml.org/wiki/workspace-to-develop-a-math-domain-for-dita> (visited on 2010-07-30) (cit. on p. 80).
- [Nov] Dimitre Novatchev. *FXSL – the Functional Programming Library for XSLT*. URL: <http://fxsl.sourceforge.net/> (visited on 2010-09-24) (cit. on p. 391).
- [NPBo3] Mikael Nilsson, Matthias Palmér, and Jan Brase. “The LOM RDF binding – principles and implementation.” In: *3rd Annual Ariadne Conference*. (Katholieke Universiteit Leuven, Belgium, Nov. 20–21, 2003). 2003 (cit. on p. 124).

- [NPJ+08] Mikael Nilsson, Andy Powell, Pete Johnston, and Ambjörn Naeve. *Expressing Dublin Core metadata using the Resource Description Framework (RDF)*. DCMI Recommendation. Dublin Core Metadata Initiative, Jan. 14, 2008. URL: <http://dublincore.org/documents/2008/01/14/dc-rdf/> (cit. on p. 124).
- [NR06] Natasha Noy and Alan Rector. *Defining N-ary Relations on the Semantic Web*. W3C Working Group Note. World Wide Web Consortium (W3C), Apr. 12, 2006. URL: <http://www.w3.org/TR/2006/NOTE-swp-n-aryRelations-20060412/> (cit. on p. 57).
- [NS09] Arnold Neumaier and Peter Schodl. *A Semantic Turing Machine*. Manuscript. 2009. URL: <http://www.mat.univie.ac.at/~neum/ms/STM.pdf> (cit. on p. 84).
- [NS10] Arnold Neumaier and Peter Schodl. “A Framework for Representing and Processing Arbitrary Mathematics.” In: *Proceedings of the International Conference on Knowledge Engineering and Ontology Development*. 2010, pp. 476–479 (cit. on p. 84).
- [OAH08] Jacco van Ossenbruggen, Alia Amin, and Michiel Hildebrand. “Why Evaluating Semantic Web Applications is Difficult. Exploring HCI Challenges.” In: [DsG+08]. URL: <http://CEUR-WS.org/Vol-543/> (cit. on p. 315).
- [Obe99] James Oberg. “Why the Mars Probe went off Course.” In: *IEEE Spectrum* (Dec. 1999), pp. 34–39 (cit. on p. 256).
- [Ocf] *OEBPS Container Format (OCF)*. Recommended Specification. Version 1.0. International Digital Publishing Forum, Sept. 11, 2006. URL: <http://www.idpf.org/ocf/ocf1.0/download/ocf10.htm> (cit. on p. 78).
- [ODM+06] Eyal Oren, Renaud Delbru, Knud Möller, Max Völkel, and Siegfried Handschuh. “Annotation and Navigation in Semantic Wikis.” In: *1st Workshop on Semantic Wikis*. (Budva, Montenegro, June 12, 2006). Ed. by Max Völkel, Sebastian Schaffert, and Stefan Decker. CEUR Workshop Proceedings 206. Aachen, 2006. URL: <http://ceur-ws.org/Vol-206/> (cit. on p. 283).
- [OE09] Christian-Emil Ore and Øyvind Eide. “TEI and cultural heritage ontologies: Exchange of information?” In: *Literary and Linguistic Computing* 24.2 (2009), pp. 161–172 (cit. on p. 78).
- [Ogb05] Uche Ogbuji. *Thinking XML: State of the art in XML modeling. What do developers need to know about the various approaches to semantic transparency?* Mar. 11, 2005. URL: <http://www.ibm.com/developerworks/xml/library/x-think30.html> (visited on 2010-10-27) (cit. on p. 54).
- [OKP+10] Tope Omitola, Christos L. Koumenides, Igor O. Popov, Yang Yang, Manuel Salvadores, Martin Szomszor, Tim Berners-Lee, Nicholas Gibbins, Wendy Hall, mc schraefel, and Nigel Shadbolt. “Put in Your Postcode, Out Comes the Data: A Case Study.” In: [AAH+10], pp. 318–332 (cit. on pp. 20, 206).
- [Olv] OpenLink Software. *OpenLink Universal Integration Middleware – Virtuoso Product Family*. URL: <http://virtuoso.openlinksw.com> (visited on 2009-10-22) (cit. on pp. 60, 228).

- [Omd] *OMDoc*. URL: <http://omdoc.org> (visited on 2010-11-11) (cit. on p. 67).
- [OPo9] Fabrizio Orlandi and Alexandre Passant. “Enabling cross-wikis integration by extending the SIOC ontology.” In: [LSSM+09]. URL: <http://ceur-ws.org/Vol-464/> (cit. on p. 47).
- [Opea] *OpenMath 3 Trac*. URL: <http://trac.mathweb.org/OM3> (visited on 2010-10-10) (cit. on p. 180).
- [Opeb] *OpenMath Home*. URL: <http://www.openmath.org> (visited on 2009-10-22) (cit. on p. 65).
- [Opec] *OpenMath Jira*. URL: <http://jira.activemath.org/browse/OM> (visited on 2010-10-11) (cit. on p. 179).
- [Oped] *OpenMath Symbols*. URL: <http://www.openmath.org/cdindex.html> (visited on 2010-10-28) (cit. on p. 322).
- [Opf] *Open Packaging Format (OPF)*. Recommended Specification. Version 2.0 v1.0. International Digital Publishing Forum, Sept. 11, 2007. URL: [http://www.idpf.org/2007/opf/OPF\\_2.0\\_final\\_spec.html](http://www.idpf.org/2007/opf/OPF_2.0_final_spec.html) (cit. on pp. 78, 126).
- [Opp92] A. N. Oppenheim. *Questionnaire Design, Interviewing and Attitude Measurement*. London: Pinter Publishers, 1992 (cit. on p. 321).
- [Ops] *Open Publication Structure (OPS)*. Recommended Specification. Version 2.0 v1.0. International Digital Publishing Forum, Sept. 11, 2007. URL: [http://www.idpf.org/2007/ops/OPS\\_2.0\\_final\\_spec.html](http://www.idpf.org/2007/ops/OPS_2.0_final_spec.html) (cit. on p. 78).
- [Orto9] Felipe Ortega. “Wikipedia: A quantitative analysis.” PhD thesis. Universidad Rey Juan Carlos, Madrid, 2009 (cit. on p. 23).
- [Owla] *OWLDoc*. URL: <http://code.google.com/p/co-ode-owl-plugins/wiki/OWLDoc> (visited on 2010-12-20) (cit. on p. 149).
- [Owlb] *The OWL API*. URL: <http://owlapi.sourceforge.net> (visited on 2010-01-05) (cit. on pp. 202, 274).
- [O’Ro5] Tim O’Reilly. *What is Web 2.0*. Sept. 2005. URL: <http://oreilly.com/web2/archive/what-is-web-20.html> (visited on 2009-10-22) (cit. on p. 8).
- [Pad] Luca Padovani. *GtkMathView*. URL: <http://helm.cs.unibo.it/mml-widget/> (visited on 2010-07-16) (cit. on p. 17).
- [Pan] *panta rhei*. URL: <http://trac.kwarc.info/panta-rhei> (visited on 2009-10-22) (cit. on pp. 49, 352).
- [Pap80] Seymour Papert. *Mindstorms. Children, Computers, and Powerful Ideas*. Basic Books, 1980 (cit. on p. 306).

- [PBK+06] Emmanuel Pietriga, Chris Bizer, David Karger, and Ryan Lee. “Fresnel – A Browser-Independent Presentation Vocabulary for RDF.” In: *5th International Semantic Web Conference (ISWC)*. Ed. by Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo. Lecture Notes in Computer Science 4273. Springer Verlag, 2006, pp. 158–171. ISBN: 3-540-49029-9 (cit. on p. 212).
- [PCB+09] Alexandre Passant, Paolo Ciccarese, John G. Breslin, and Tim Clark. “SWAN/SIOC: Aligning Scientific Discourse Representation and Social Semantics.” In: [CLM+09]. URL: <http://CEUR-WS.org/Vol-523/> (cit. on p. 136).
- [PCSF08] C. Michael Pilato, Ben Collins-Sussman, and Brian W. Fitzpatrick. *Version Control With Subversion*. 2nd ed. Sebastopol, CA, USA: O’Reilly & Associates, Inc., 2008. ISBN: 978-0-596-51033-6. URL: <http://svnbook.red-bean.com> (cit. on pp. 278, 397).
- [Pes07] Darko Pesikan. “Coping with Content Representations of Mathematics in Editor Environments: nOMDoc mode.” Bachelor’s Thesis. Computer Science, Jacobs University, Bremen, 2007 (cit. on p. 185).
- [Pfe01] Frank Pfenning. “Logical Frameworks.” In: *Handbook of Automated Reasoning*. Ed. by Alan Robinson and Andrei Voronkov. Vol. I and II. Elsevier Science and MIT Press, 2001 (cit. on pp. 69, 388).
- [PHC+09] Raúl Palma, Peter Haase, Oscar Corcho, and Asunción Gómez-Pérez. “Change Representation For OWL 2 Ontologies.” In: *OWL: Experiences and Directions (OWLED)*. Ed. by Rinke Hoekstra and Peter F. Patel-Schneider. Oct. 2009 (cit. on pp. 48, 124).
- [PHP+10] Adrian Paschke, Nicola Henze, Tassilo Pellegrini, and Hans Weigand, eds. *6th International Conference on Semantic Systems (I-Semantics) and the 5th International Conference on Pragmatic Web*. ACM, 2010. ISBN: 978-1-4503-0014-8. URL: <http://portal.acm.org/citation.cfm?id=1839707> (cit. on pp. 463, 477).
- [PJ03] Andy Powell and Pete Johnston. *Guidelines for Implementing Dublin Core in XML*. DCMI Recommendation. Dublin Core Metadata Initiative, Apr. 2, 2003. URL: <http://dublincore.org/documents/2003/04/02/dc-xml-guidelines/> (cit. on p. 162).
- [Plaa] *Planetary Developer Forum*. URL: <http://trac.mathweb.org/planetary/> (visited on 2011-01-20) (cit. on p. 352).
- [Plab] *PlanetMath.org – Math for the people, by the people*. URL: <http://planetmath.org> (visited on 2011-01-06) (cit. on pp. 14, 301).
- [Plü04] Judith Plümer. *Erinnerungen an Prof. Dr. Schwänzl*. 2004. URL: <http://d-mathnet.preprints.org/research/wissinfo/NachrufRS.html> (visited on 2010-11-06) (cit. on p. 16).
- [PM04] Helena Sofia Pinto and João P. Martins. “Ontologies: How can They be Built?” In: *Knowledge and Information Systems 6* (2004), pp. 441–464 (cit. on pp. 62, 102).

- [PNJ+08] Frederik Pfisterer, Markus Nitsche, Anthony Jameson, and Catalin Barbu. “User-Centered Design and Evaluation of Interface Enhancements to the Semantic MediaWiki. Exploring HCI Challenges.” In: [DsG+08]. URL: <http://CEUR-W S.org/Vol-543/> (cit. on p. 316).
- [PNLo2] Adam Pease, Ian Niles, and John Li. “The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications.” In: *Working Notes of the AAAI Workshop on Ontologies and the Semantic Web*. 2002 (cit. on p. 159).
- [Pola] *Deolalikar P vs NP paper*. URL: [http://michaelnielsen.org/polymath1/index.php?title=Deolalikar\\_P\\_vs\\_NP\\_paper&oldid=3654](http://michaelnielsen.org/polymath1/index.php?title=Deolalikar_P_vs_NP_paper&oldid=3654) (visited on 2010-11-03) (cit. on p. 13).
- [Polb] *Polymath1Wiki*. URL: <http://michaelnielsen.org/polymath1/> (visited on 2010-07-12) (cit. on p. 13).
- [Polc] *The polymath blog*. URL: <http://polymathprojects.org/> (visited on 2010-07-12) (cit. on p. 13).
- [Proa] *ProgrammableWeb. Mashups, APIs, and the Web as Platform*. URL: <http://www.programmableweb.com> (visited on 2010-07-16) (cit. on p. 16).
- [Prob] *ProofWiki*. URL: <http://www.proofwiki.org> (visited on 2010-07-16) (cit. on p. 14).
- [Proc] *Protégé*. URL: <http://protege.stanford.edu> (visited on 2010-01-06) (cit. on pp. 160, 261, 363).
- [Prod] *Wiki for formalized mathematics based on ProofWeb*. URL: <http://prover.cs.ru.nl/wiki.php> (visited on 2010-10-07) (cit. on p. 302).
- [PSo6] Christian Pentzold and Sebastian Seidenglanz. “Foucault@Wiki – First Steps Towards a Conceptual Framework for the Analysis of Wiki Discourses.” In: *Proceedings of the International Symposium on Wikis (WikiSym)*. (Odense, Denmark, Aug. 21–23, 2006). Ed. by Dirk Riehle and James Noble. ACM Press. 2006. URL: <http://www.wikisym.org/ws2006/proceedings/p59.pdf> (cit. on p. 287).
- [PSo8] Eric Prud’hommeaux and Andy Seaborne. *SPARQL Query Language for RDF*. W3C Recommendation. World Wide Web Consortium (W3C), Jan. 15, 2008. URL: <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/> (cit. on p. 60).
- [PSMo9] Peter F. Patel-Schneider and Boris Motik. *OWL 2 Web Ontology Language: Mapping to RDF Graphs*. W3C Recommendation. World Wide Web Consortium (W3C), Oct. 27, 2009. URL: <http://www.w3.org/TR/2009/REC-owl2-mapping-to-rdf-20091027/> (cit. on p. 150).
- [PSSo3] Peter F. Patel-Schneider and Jérôme Siméon. “The Yin/Yang Web: A Unified Model for XML Syntax and RDF Semantics.” In: *IEEE Transactions on Knowledge and Data Engineering* 15.4 (2003), pp. 797–812 (cit. on p. 140).



- [PSTo4] Helena Sofia Pinto, Steffen Staab, and Christoph Tempich. “DILIGENT: Towards a fine-grained methodology for Distributed, Loosely-controlled and evolving Engineering of oNTologies.” In: *ECAI*. 2004, pp. 393–397 (cit. on p. 132).
- [PZo6] Luca Padovani and Stefano Zacchiroli. “From Notation to Semantics: There and Back Again.” In: [BFo6], pp. 194–207 (cit. on p. 63).
- [Pól73] George Pólya. *How to Solve it. A New Aspect of Mathematical Method*. Princeton University Press, 1973 (cit. on pp. 4, 5, 13, 176).
- [Raba] Florian Rabe. *MMT – A Module system for Mathematical Theories*. URL: <http://trac.kwarc.info/MMT/> (visited on 2010-08-18) (cit. on pp. 70, 74, 102, 155, 222, 386, 388, 389).
- [Rabb] Florian Rabe. *TNTBase – MMT Plugin*. URL: <http://tntbase.org/wiki/MMTPlugin> (visited on 2010-05-10) (cit. on p. 226).
- [Rabo8] Florian Rabe. “Representing Logics and Logic Translations.” PhD thesis. Jacobs University Bremen, 2008. URL: <http://kwarc.info/frabe/Research/phdthesis.pdf> (cit. on pp. 38, 43, 69, 70, 81, 93, 102, 199, 200).
- [Raio8] Yves Raimond. “A Distributed Music Information System.” PhD thesis. Queen Mary, University of London, Nov. 2008. URL: <http://moustaki.org/phd/> (cit. on p. 351).
- [RCD+10] Jonathan Robie, Don Chamberlin, Michael Dyck, and John Snelson. *XQuery 3.0: An XML Query Language*. W3C Working Draft. World Wide Web Consortium (W3C), Dec. 14, 2010. URL: <http://www.w3.org/TR/2010/WD-xquery-30-20101214> (cit. on p. 228).
- [Rdd] *Resource Directory Description Language (RDDL)*. Feb. 18, 2002. URL: <http://www.rddl.org/> (visited on 2010-08-07) (cit. on p. 54).
- [Rdfa] *RDFa Tools*. Aug. 3, 2010. URL: <http://rdfa.info/wiki/?title=Tools&oldid=1162> (visited on 2010-08-27) (cit. on p. 168).
- [Rdfb] *RDFa wiki*. URL: <http://rdfa.info/wiki/>.
- [RDSM+02] Allen Renear, David Dubin, C. M. Sperberg-McQueen, and Claus Huitfeldt. “Towards a Semantics for XML Markup.” In: *DocEng’02*. ACM, 2002 (cit. on p. 140).
- [Reio5] Gerald Reif. “WEESA – Web Engineering for Semantic Web Applications.” PhD thesis. Technische Universität Wien, May 2005 (cit. on p. 270).
- [Ren97] Linda van Rens. “Usability problem classifier.” MA thesis. Blacksburg, VA, USA: Virginia Polytechnic Institute and State University, 1997 (cit. on p. 325).
- [Ret09] Krzysztof Retel. “Gradual Computerisation and Verification of Mathematics. MathLang’s Path into Mizar.” PhD thesis. Edinburgh: Heriot-Watt University, Apr. 2009 (cit. on pp. 43, 75, 88, 92, 204).
- [RGJ05] Gerald Reif, Harald Gall, and Mehdi Jazayeri. “WEESA: Web engineering for semantic Web applications.” In: [EH05], pp. 722–729. ISBN: 1-59593-046-9 (cit. on p. 270).

- [Rha] *Rhaptos Trac: Collection Structure Redesign / Inception*. July 20, 2009. URL: <https://trac.rhaptos.org/trac/rhaptos/wiki/CollectionStructureRedesign/Inception?version=54> (visited on 2010-07-31) (cit. on p. 80).
- [Ria] *RIACA OpenMath products*. URL: <http://www.riaca.win.tue.nl/projects/openmath/> (visited on 2009-10-22) (cit. on p. 187).
- [RK11] Florian Rabe and Michael Kohlhase. “A Scalable Module System.” Manuscript, submitted to Information & Computation. 2011. URL: <http://kwarc.info/frabe/Research/mmt.pdf> (cit. on pp. 38, 69, 70, 155).
- [Rob09] Andrew Robbins. *Semantic MathML*. June 8, 2009. URL: <http://straymindcough.blogspot.com/2009/06/semantic-mathml.html> (visited on 2010-08-09) (cit. on p. 84).
- [RP05] Robert G. Raskin and Michael J. Pan. “Knowledge representation in the semantic web for Earth environmental terminology (SWEET).” In: *Computers & Geosciences* 31 (2005), pp. 1119–1125. DOI: [10.1016/j.cageo.2004.12.004](https://doi.org/10.1016/j.cageo.2004.12.004) (cit. on p. 130).
- [RSC09] Ricardo Radaelli-Sanchez and Connexions. *The Intermediate CNXML (Connexions Web Site)*. Apr. 10, 2009. URL: <http://cnx.org/content/m9006/2.22/> (visited on 2010-08-11) (cit. on p. 80).
- [Rus09] Chris Rusbridge. *Digital Curation Blog: Semantically richer PDF?* Apr. 6, 2009. URL: <http://digitalcuration.blogspot.com/2009/04/semantically-richer-pdf.html> (visited on 2010-08-25) (cit. on p. 223).
- [RW73] Horst W. J. Rittel and Melvin M. Webber. “Dilemmas in a General Theory of Planning.” In: *Policy Sciences* 4.2 (June 1973), pp. 155–169 (cit. on pp. 50, 233).
- [SAR+11] Manu Sporny, Benjamin Adrian, Nathan Rixham, Mark Birbeck, and Ivan Herman. *RDFa API. An API for extracting structured data from Web documents*. W3C Working Draft. World Wide Web Consortium (W3C), Apr. 19, 2011. URL: <http://www.w3.org/TR/2011/WD-rdfa-api-20110419/> (cit. on pp. 60, 218).
- [SB10] Aaron Schulz and Joerg Baach. *MediaWiki – Flagged Revisions*. June 27, 2010. URL: <http://www.mediawiki.org/w/index.php?title=Extension:FlaggedRevs&oldid=333905> (cit. on p. 201).
- [SBA05] Jörg Siekmann, Christoph Benz Müller, and Serge Autexier. “Computer Supported Mathematics with OMEGA.” In: *Journal of Applied Logic, special issue on Mathematics Assistance Systems* (Dec. 2005). Ed. by Christoph Benz Müller (cit. on p. 69).
- [SBF98] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. “Knowledge Engineering: Principles and Methods.” In: *Data & Knowledge Engineering* 25.1–2 (1998), pp. 161–198 (cit. on p. 60).
- [SCo8] Leo Sauermann and Richard Cyganiak. *Cool URIs for the Semantic Web*. W3C Interest Group Note. World Wide Web Consortium (W3C), Dec. 3, 2008. URL: <http://www.w3.org/TR/2008/NOTE-cooluris-20081203/> (cit. on pp. 53, 138, 210, 211).



- [Sch] *schematron*. URL: <http://schematron.com> (visited on 2009-12-22) (cit. on p. 198).
- [Scho2] Irene Schena. “Towards a Semantic Web for Formal Mathematics.” Technical Report UBLCS–2002–6. PhD thesis. University of Bologna, Mar. 2002 (cit. on pp. 86, 212).
- [Scho6] Sebastian Schaffert. “IkeWiki: A Semantic Wiki for Collaborative Knowledge Management.” In: *1st International Workshop on Semantic Technologies in Collaborative Applications (STICA)*. (Manchester, UK, June 2006). 2006 (cit. on pp. 208, 287, 386).
- [Scho9] Markus Schatten. “Programming Languages for Autopoiesis Facilitating Semantic Wiki Systems.” PhD thesis. University of Zagreb, 2009 (cit. on p. 353).
- [Sci] *The SCIENCE Project – Symbolic Computation Infrastructure for Europe*. URL: <http://www.symbolic-computation.org/> (visited on 2009-10-22) (cit. on pp. 18, 184).
- [SDo8] Jonathan Stratford and James H. Davenport. “Unit Knowledge Management.” In: [ACR+08], pp. 382–397 (cit. on pp. 256, 260).
- [SDRo8] Uli Sattler, Cathy Dolbear, and Alan Ruttenberg, eds. *OWL: Experiences and Directions (OWLED)*. Oct. 2008 (cit. on pp. 478, 501).
- [SEG+09] Sebastian Schaffert, Julia Eder, Szaby Grünwald, Thomas Kurz, Mihai Radulescu, Rolf Sint, and Stephanie Stroka. “KiWi – A Platform for Semantic Social Software.” In: [LSSM+09]. URL: <http://ceur-ws.org/Vol-464/> (cit. on pp. 309, 310).
- [Sema] *Semantic MediaWiki*. URL: <http://semantic-mediawiki.org> (visited on 2010-03-04) (cit. on pp. 283, 284).
- [Semb] *SemWiki.org – The Semantic Wiki Community*. URL: <http://semwiki.org> (visited on 2010-06-17) (cit. on p. 306).
- [Sero8] François-Paul Servant. “Linking Enterprise Data.” In: [BHI+08]. URL: <http://CEUR-WS.org/Vol-369/> (cit. on p. 20).
- [SFNT+08] Abraham Sebastian, Natalya Fridman Noy, Tania Tudorache, and Mark A. Musen. “A Generic Ontology for Collaborative Ontology-Development Workflows.” In: *Knowledge Engineering: Practice and Patterns*. 16th International Conference, EKAW (Acitrezza, Italy, Sept. 29–Oct. 2, 2008). Ed. by Aldo Gangemi and Jérôme Euzenat. Lecture Notes in Computer Science 5268. Springer Verlag, 2008, pp. 318–328. ISBN: 978-3-540-87695-3 (cit. on p. 50).
- [SGRo9] Josef Schneeberger, Günther Görz, and Jürgen Renn. “Interview mit Jürgen Renn.” In: *Künstliche Intelligenz* 4 (Dec. 2009), pp. 48–53 (cit. on p. 12).
- [Sio] *SIOC Types Ontology Module Namespace*. URL: <http://rdfs.org/sioc/types> (visited on 2009-10-27) (cit. on p. 131).
- [SKoo] Andreas Strotmann and Ladislav Kohout. “OpenMath: Compositionality Achieved at Last.” In: *Bulletin of the ACM Special Interest Group on Symbolic and Automated Mathematics (SIGSAM)* 34.2 (2000), pp. 66–72 (cit. on p. 74).

- [SKo8] Neil Soiffer and Kathy Kahl. *Specification for the Digital Talking Book Modular Extension for Mathematics*. Tech. rep. Version 1.1. DAISY Consortium, Sept. 24, 2008. URL: <http://www.daisy.org/projects/mathml/mathml-in-daisy-spec.html> (cit. on p. 80).
- [SKG+10] Heinrich Stamerjohanns, Michael Kohlhase, Deyan Ginev, Catalin David, and Bruce Miller. “Transforming large collections of scientific publications to XML.” In: *Mathematics in Computer Science 3.3 (2010): Special Issue on Authoring, Digitalization and Management of Mathematical Knowledge*. Ed. by Serge Autexier, Petr Sojka, and Masakazu Suzuki, pp. 299–307. URL: <http://kwarc.info/kohlhase/papers/mcs10.pdf> (cit. on pp. 352, 356).
- [Sloo3] Neil J. A. Sloane. “The On-Line Encyclopedia of Integer Sequences.” In: *Notices of the AMS* 50.8 (2003), p. 912 (cit. on p. 355).
- [SM93] Frank M. Shipman and Catherine C. Marshall. *Formality Considered Harmful: Experiences, Emerging Themes, and Directions*. Report. University of Colorado at Boulder, Department of Computer Science, 1993. URL: <http://www.cs.colorado.edu/departement/publications/reports/docs/CU-CS-648-93.pdf> (cit. on pp. 23, 234).
- [Smw] SMW+ *Semantic Enterprise Wiki*. URL: <http://wiki.ontoprise.de> (visited on 2010-07-17) (cit. on p. 10).
- [Sow10] John F. Sowa. *Re: using SKOS for controlled values for controlled vocabulary*. e-mail to [ontolog-forum@ontolog.cim3.net](mailto:ontolog-forum@ontolog.cim3.net). Oct. 9, 2010. URL: <http://ontolog.cim3.net/forum/ontolog-forum/2010-10/msg00019.html> (cit. on p. 99).
- [SP10] Ben Shneiderman and Catherine Plaisant. *Designing the user interface: strategies for effective human-computer interaction*. 5th ed. Addison-Wesley, 2010. ISBN: 978-0-321-53735-5 (cit. on p. 315).
- [SPB10] Jodi Schneider, Alexandre Passant, and John G. Breslin. “Enhancing MediaWiki Talk pages with Semantics for Better Coordination. A Proposal.” In: *5th Workshop on Semantic Wikis*. (Hersonissos, Crete, Greece, May 31, 2010). Ed. by Christoph Lange, Jochen Reutelshöfer, Sebastian Schaffert, and Hala Skaf-Molli. *CEUR Workshop Proceedings* 632. Aachen, 2010. URL: <http://ceur-ws.org/Vol-632/> (cit. on p. 287).
- [SPCG+06] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. “Pellet: A Practical OWL-DL Reasoner.” In: *Web Semantics: Science, Services and Agents on the World Wide Web* (2006). To appear. URL: <http://www.mindswap.org/papers/PelletJWS.pdf> (cit. on p. 396).
- [Spe03] Wolfram Sperber. “Math-Net International and the Math-Net Page.” In: *Electronic Information and Communication in Mathematics*. ICM International Satellite Conference (Beijing, China, Aug. 29–31, 2002). Ed. by Fengshan Bai and Bernd Wegner. *Lecture Notes in Computer Science* 2730. Springer, 2003, pp. 169–177 (cit. on p. 16).

- [SRPo7] Helen Sharp, Yvonne Rogers, and Jenny Preece. *Interaction design: beyond human-computer interaction*. 2nd ed. John Wiley & Sons, 2007. ISBN: 978-0-470-01866-8. URL: <http://www.id-book.com> (cit. on pp. 315, 321, 323, 325).
- [SSo6] Alan Sexton and Volker Sorge. “Processing Textbook-Style Matrices.” In: [Koho6a], pp. 111–125 (cit. on p. 245).
- [SSo9] Pavel Smrž and Marek Schmidt. “Information Extraction in Semantic Wikis.” In: [LSSM+09]. URL: <http://ceur-ws.org/Vol-464/> (cit. on pp. 186, 196).
- [SS10] Katharina Siorpaes and Elena Simperl. “Human Intelligence in the Process of Semantic Content Creation.” In: *World Wide Web* 13 (2010), pp. 33–59 (cit. on p. 21).
- [ST10] John Sheridan and Jeni Tennison. “Linking UK Government Data.” In: [BHBL+10]. URL: <http://CEUR-WS.org/Vol-628/> (cit. on p. 20).
- [Sta] *Stack Overflow*. URL: <http://stackoverflow.com> (visited on 2010-09-13) (cit. on pp. 13, 233).
- [Ste] *sTeX Emacs Mode*. URL: <https://svn.kwarc.info/repos/stex/projects/emacs> (visited on 2009-11-10) (cit. on p. 185).
- [Stoo6] Margaret-Anne Storey. “Theories, tools and research methods in program comprehension: past, present and future.” In: *Software Quality* 14 (2006), pp. 187–208 (cit. on p. 11).
- [Stro3] Andreas Strotmann. “Content Markup Language Design Principles.” PhD thesis. Florida State University, 2003. URL: <http://www.cs.fsu.edu/research/reports/TR-030702.pdf> (cit. on p. 62).
- [Stro8] Jonathan Stratford. *Creating an extensible Unit Converter using OpenMath as the Representation of the Semantics of the Units*. Tech. rep. 2008-02. University of Bath, June 2008. URL: <http://www.cs.bath.ac.uk/pubdb/download.php?resID=290> (cit. on pp. 256, 259, 260).
- [SVo8] Sebastian Schaffert and Max Völkel. *Semantic Wikis: The Wiki Way to the Semantic Web*. Presented in the Ontolog mini-series on Semantic Wikis, session 1 ([http://ontolog.cim3.net/cgi-bin/wiki.pl?ConferenceCall\\_2008\\_10\\_23](http://ontolog.cim3.net/cgi-bin/wiki.pl?ConferenceCall_2008_10_23)). Oct. 23, 2008. URL: <http://www.slideshare.net/xamde/a-survey-of-the-landscape-and-stateofart-in-semantic-wiki-presentation> (cit. on pp. 282, 306).
- [SVM+03] Sudhanshu Sipani, Kunal Verma, John A. Miller, and Boanerges Aleman-Meza. “Designing a high-performance database engine for the ‘DB4XML’ native XML database system.” In: *Journal of Systems and Software* 69 (2003), pp. 87–104 (cit. on p. 228).
- [Swe] *Semantic Web for Earth and Environmental Terminology (SWEET)*. NASA. URL: <http://sweet.jpl.nasa.gov/> (visited on 2010-08-22) (cit. on p. 130).

- [SX02] Daniel Suthers and Jun Xu. “Kūkākūkā: An Online Environment for Artifact-Centered Discourse.” In: *Education Track of the 11th World Wide Web Conference (WWW)*. May 2002, pp. 472–480 (cit. on p. 309).
- [Sys] SysMO-DB SEEK. URL: <http://www.sysmo-db.org/seek/> (visited on 2010-11-30) (cit. on p. 20).
- [TAFB+10] Carlos Tejo-Alonso, Sergio Fernández, Diego Berrueta, Luis Polo, María Jesús Fernández, and Víctor Morlán. “eZaragoza, a tourist promotional mashup.” In: [GENL+10]. URL: <http://sites.google.com/a/fh-hannover.de/aimashup/home/ezaragoza> (cit. on p. 9).
- [Tano3] Andrew S. Tanenbaum. *Computer networks*. 4th ed. Pearson Education, 2003 (cit. on p. 62).
- [TDO07] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. “Sindice.com: Weaving the Open Linked Data.” In: [ACN+07], pp. 552–565. ISBN: 978-3-540-76297-3 (cit. on p. 163).
- [Tei] TEI – Ontologies SIG. URL: <http://www.tei-c.org/Activities/SIG/Ontologies/> (visited on 2010-08-02) (cit. on p. 78).
- [Ten+] Jeni Tennison et al. *rdfQuery – RDF processing in your browser*. URL: <http://rdfquery.googlecode.com> (visited on 2010-01-20) (cit. on pp. 218, 389).
- [Ten09a] Jeni Tennison. *HTML5/RDFa Arguments*. Aug. 21, 2009. URL: <http://www.jenitennison.com/blog/node/124> (visited on 2010-02-02) (cit. on p. 60).
- [Ten09b] Jeni Tennison. *What You Can’t Do with HTML5 Microdata*. May 13, 2009. URL: <http://www.jenitennison.com/blog/node/103> (visited on 2010-02-02) (cit. on p. 60).
- [Texa] *TeX Document Center (TeXDocC)*. URL: <http://www.texdocc.de> (visited on 2010-09-11) (cit. on p. 184).
- [Texb] *GNU TeXMacS*. URL: <http://www.texmacs.org> (visited on 2010-09-07) (cit. on p. 186).
- [Thu94] William P. Thurston. “On Proof and Progress in Mathematics.” In: *Bulleting of the American Mathematical Society* 30.2 (1994), pp. 161–177 (cit. on p. 5).
- [Tik] Tiki Wiki CMS Groupware. *Software made the wiki way*. URL: <http://tikiwiki.org> (visited on 2010-10-03) (cit. on p. 283).
- [Tilo6] Paul van Tilburg. *Exploring the Core of MathLang*. Internship Report. ULTRA group, School of Mathematics and Computer Science, Heriot-Watt University, Edinburgh, 2006. URL: <http://paul.luon.net/writings/reports/HWU-MACS-MathLang.pdf> (visited on 2010-08-11) (cit. on p. 75).
- [Tin] *TinyMCE – JavaScript WYSIWYG editor*. URL: <http://tinymce.moxiecode.com/> (visited on 2009-11-10) (cit. on pp. 186, 379).

- [TL09] Vladimir Tomberg and Mart Laanpere. “RDFa versus Microformats: Exploring the Potential for Semantic Interoperability of Mash-up Personal Learning Environments.” In: *2nd International Workshop on Mashup Personal Learning Environments (MUPPLE)*. Ed. by Fridolin Wild, Marco Kalz, Matthias Palmér, and Daniel Müller. CEUR Workshop Proceedings 506. Aachen, Sept. 2009. URL: <http://CEUR-WS.org/Vol-506/tomberg.pdf> (cit. on p. 60).
- [TLC+09] Ha Manh Tran, Christoph Lange, Georgi Chulkov, Jürgen Schönwälder, and Michael Kohlhase. “Applying Semantic Techniques to Search and Analyze Bug Tracking Data.” In: *Journal of Network and Systems Management* 17.3 (May 2009): *Special Issue on Ontologies for Network and Service Management*, pp. 285–308. DOI: [10.1007/s10922-009-9134-4](https://doi.org/10.1007/s10922-009-9134-4) (cit. on p. 231).
- [TN07] Tania Tudorache and Natasha Noy. “Collaborative Protégé.” In: *Workshop on Social and Collaborative Construction of Structured Knowledge (CKC) at WWW*. 2007 (cit. on p. 232).
- [Tnt] *TNTBase*. URL: <http://tntbase.org/> (visited on 2010-01-11) (cit. on p. 225).
- [TNT+08] Tania Tudorache, Natalya F. Noy, Samson Tu, and Mark A. Musen. “Supporting Collaborative Ontology Development in Protégé.” In: *The Semantic Web*. 7th International Semantic Web Conference (ISWC). Ed. by Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy W. Finin, and Krishnaprasad Thirunarayan. LNCS 5318. Springer Verlag, Oct. 2008 (cit. on p. 50).
- [TPS+05] Christoph Tempich, H. Sofia Pinto, York Sure, and Steffen Staab. “An Argumentation Ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT).” In: *The Semantic Web: Research and Applications*. 2nd European Semantic Web Conference (ESWC) (Hersonissos, Crete, Greece, May 29–June 1, 2005). Ed. by Asunción Gómez-Pérez and Jérôme Euzenat. Lecture Notes in Computer Science 3532. Springer Verlag, 2005, pp. 241–256. ISBN: 3-540-26124-9 (cit. on pp. 50, 234, 334).
- [Traa] *The Trac Project*. URL: <http://trac.edgewall.org/> (visited on 2009-10-22) (cit. on pp. 180, 327).
- [Trab] *Trac and Subversion*. URL: <http://trac.edgewall.org/wiki/TracSubversion> (visited on 2009-10-27) (cit. on pp. 180, 231).
- [Tri] *Tricki. a repository of mathematical know-how*. URL: <http://www.tricki.org> (visited on 2010-07-12) (cit. on p. 13).
- [Trz95] Jerzy Trzeciak. *Writing Mathematical Papers in English. a practical guide*. Gdańskie Wydawnictwo Oświatowe, 1995 (cit. on pp. 39, 40).
- [TSB+10] Jiao Tao, Evren Sirin, Jie Bao, and Deborah L. McGuinness. “Extending OWL with Integrity Constraints.” In: *23rd Workshop on Description Logics (DL)*. (Waterloo, Canada, May 4–7, 2010). Ed. by Volker Haarslev, David Toman, and Grant Weddell. CEUR Workshop Proceedings 573. Aachen: CEUR-WS.org, 2010. URL: <http://ceur-ws.org/Vol-573/> (cit. on p. 204).

- [TSL+07] Christoph Tempich, Elena Simperl, Markus Luczak, Rudi Studer, and H. Sofia Pinto. “Argumentation-Based Ontology Engineering.” In: *IEEE Intelligent Systems* 22.6 (2007), pp. 52–59. ISSN: 1541-1672 (cit. on pp. 50, 303).
- [TVNo8] Tania Tudorache, Jennifer Vendetti, and Natalya Noy. “Web-Protégé: A Lightweight OWL Ontology Editor for the Web.” In: [SDRo8] (cit. on p. 50).
- [UAR+10] Josef Urban, Jesse Alama, Piotr Rudnicki, and Herman Geuvers. “A wiki for Mizar: Motivation, considerations, and initial prototype.” In: [ACD+10], pp. 455–469. ISBN: 3642141277. arXiv:1005.4552v1 [cs.DL] (cit. on p. 14).
- [Ull08] Carsten Ullrich. *Pedagogically Founded Courseware Generation for Web-Based Learning*. LNCS. Springer Verlag, Sept. 2008. ISBN: 978-3-540-88213-8. URL: <http://www.springerlink.com/content/k604618p5351/> (cit. on pp. 43, 48, 240).
- [Uni] Unicode. Version 5.2.0. Unicode, Inc. Oct. 21, 2009. URL: <http://www.unicode.org/versions/Unicode5.2.0/> (visited on 2010-08-11) (cit. on p. 53).
- [Urbo6] Josef Urban. “MoMM – Fast Interreduction and Retrieval in Large Libraries of Formalized Mathematics.” In: *International Journal on Artificial Intelligence Tools (IJAIT)* 15.1 (2006), pp. 109–130. DOI: 10.1142/S0218213006002588 (cit. on p. 229).
- [Urla] *Mathematica*. URL: <http://www.wolfram.com/products/mathematica/> (visited on 2010-06-05) (cit. on pp. 11, 353).
- [Urlb] *Wolfram|Alpha*. URL: <http://www.wolframalpha.com> (visited on 2011-05-05) (cit. on p. 255).
- [Urlc] *Wolfram|Alpha API*. URL: <http://www.wolframalpha.com/developers.html> (visited on 2011-05-05) (cit. on p. 255).
- [Van] *Vanilla Forums*. URL: <http://vanillaforums.org> (visited on 2010-09-22) (cit. on p. 352).
- [VDPM+08] Davy Van Deursen, Chris Poppe, Gaëtan Martens, Erik Mannens, and Rik Van de Walle. “XML to RDF Conversion: a Generic Approach.” In: *Fourth International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution, Proceedings*. 2008 (cit. on p. 270).
- [Vee01] Jeffrey Veen. *The Art & Science of Web Design*. New Riders, 2001. ISBN: 0-7897-2370-0 (cit. on p. 208).
- [VKR+10] Denny Vrandečić, Markus Krötzsch, Sebastian Rudolph, and Uta Lösch. “Leveraging Non-Lexical Knowledge for the Linked Open Data Web.” In: *The Fifth RAFT, the yearly bilingual publication on nonchalant research*. Ed. by Rodolphe Héliot and Antoine Zimmermann. 2010. URL: [http://km.aifb.kit.edu/projects/numbers/linked\\_open\\_numbers.pdf](http://km.aifb.kit.edu/projects/numbers/linked_open_numbers.pdf) (cit. on p. 355).
- [VKV+06] Max Völkel, Markus Krötzsch, Denny Vrandečić, Heiko Haller, and Rudi Studer. “Semantic Wikipedia.” In: [Www], pp. 585–594. URL: <http://www.aifb.uni-karlsruhe.de/WBS/hha/papers/SemanticWikipedia.pdf> (cit. on p. 283).



- [VL] Mario Volke and Thorsten Liebig. *Jigs for OWL*. URL: <http://www.jigs4owl.com> (visited on 2010-09-19) (cit. on p. 241).
- [VLH+10] Denny Vrandečić, Christoph Lange, Michael Hausenblas, Jie Bao, and Li Ding. “Semantics of Governmental Statistics Data.” In: *Proceedings of WebSci’10: Extending the Frontiers of Society On-Line*. Web Science Trust, 2010. URL: <http://journal.webscience.org/400/> (cit. on pp. 20, 206, 207, 237).
- [VLL10] Mario Volke, Thorsten Liebig, and Marko Luther. “Jigs4OWL.” In: [GENL+10]. URL: <http://sites.google.com/a/fh-hannover.de/aimashup/home/jigs4owl> (cit. on p. 241).
- [VMS99] A. Marie Vans, Anneliese von Mayrhauser, and Gabriel Somlo. “Program understanding behavior during corrective maintenance of large-scale software.” In: *International Journal of Human-Computer Studies* 51 (1999), pp. 31–70 (cit. on p. 11).
- [Vra10] Denny Vrandečić. “Ontology Evaluation.” PhD thesis. Karlsruhe Institute of Technology (KIT), 2010 (cit. on pp. 202, 204).
- [W3C] *Document Object Model DOM*. URL: <http://www.w3.org/DOM/> (visited on 2009-12-07) (cit. on pp. 60, 270).
- [W3c] *SKOS Simple Knowledge Organization System*. URL: <http://www.w3.org/2004/02/skos/> (visited on 2010-08-22) (cit. on p. 125).
- [W3Ca] World Wide Web Consortium (W3C), ed. *Cascading Style Sheets*. URL: <http://www.w3.org/Style/CSS/> (visited on 2009-10-22) (cit. on p. 55).
- [W3Cb] World Wide Web Consortium (W3C), ed. *Resource Description Framework (RDF)*. URL: <http://www.w3.org/RDF/> (visited on 2009-10-22) (cit. on p. 55).
- [WABo6] Marc Wagner, Serge Autexier, and Christoph Benzmüller. “PLATO: A Mediator between Text-Editors and Proof Assistance Systems.” In: *7th Workshop on User Interfaces for Theorem Provers (UITP)* 174.2 (2006), pp. 87–107 (cit. on pp. 25, 301).
- [Walo8] Norman Walsh. *The DocBook Schema*. Committee Specification. Version 5.0. OASIS, Aug. 2008. URL: <http://www.docbook.org/specs/docbook-5.0-spec-cs-01.html> (cit. on p. 77).
- [WBB+09] Makarius Wenzel, Clemens Ballarin, Stefan Berghofer, Timothy Bourke, Lucas Dixon, Florian Haftmann, Gerwin Klein, Alexander Krauss, Tobias Nipkow, David von Oheimb, Larry Paulson, and Sebastian Skalberg. *The Isabelle/Isar Reference Manual*. Ed. by Makarius Wenzel. Apr. 19, 2009. URL: <http://isabelle.in.tum.de/doc/isar-ref.pdf> (visited on 2009-11-11) (cit. on p. 81).
- [Weia] Martin Weissman. *SlugMath Wiki*. URL: <http://slugmath.ucsc.edu/mediawiki/> (visited on 2010-10-07) (cit. on p. 284).
- [Weib] Eric W. Weisstein, ed. *Wolfram MathWorld. the web’s most extensive mathematics resource*. Wolfram Research. URL: <http://mathworld.wolfram.com> (visited on 2009-12-02) (cit. on p. 11).
- [Wik] *wikipatterns.com*. URL: <http://www.wikipatterns.com> (visited on 2010-06-25) (cit. on p. 197).



- [Wir] *WIRIS Editor – a tool for graphical edition of mathematical formulas*. URL: <http://www.wiris.com/content/view/20/> (visited on 2009-11-10) (cit. on p. 186).
- [WJF09] Franz Weigl, Mirjana Jakšić, and Burkhard Freitag. “Towards the automated verification of semi-structures documents.” In: *Data & Knowledge Engineering* 68 (2009), pp. 292–317 (cit. on p. 205).
- [WMO8] Norman Walsh and Leonard Mueller. *DocBook 5.0: The Definitive Guide*. O’Reilly, 2008 (cit. on p. 77).
- [WMO9] Claudia Wagner and Enrico Motta. “Data Republishing on the Social Semantic Web.” In: *Proceedings of the 1st Workshop on Trust and Privacy on the Social and Semantic Web (SPOT)*. (June 1, 2009). CEUR Workshop Proceedings 447. Aachen, 2009. URL: <http://CEUR-WS.org/Vol-447/> (cit. on p. 186).
- [Wola] *Wolfram|Alpha*. URL: <http://www.wolframalpha.com> (visited on 2011-05-05) (cit. on p. 16).
- [Wolb] *Wolfram|Alpha Widgets*. URL: <http://developer.wolframalpha.com/widgets/> (visited on 2011-05-05) (cit. on p. 16).
- [WR10] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*. 2nd ed. Vol. I. Cambridge, UK: Cambridge University Press, 1910 (cit. on p. 6).
- [WS03] Christoph Walther and Stephan Schweitzer. “About Verifun.” In: *Proceedings of the 19th International Conference on Automated Deduction (CADE-19)*. Ed. by Franz Baader. LNCS 2741. Springer Verlag, 2003 (cit. on p. 69).
- [Wsm] *Web Service Modeling Ontology*. URL: <http://www.wsmo.org> (visited on 2009-11-25) (cit. on p. 62).
- [WVE99] Martijn van Welie, Gerrit C. van der Veer, and Anton Eliëns. “Breaking down Usability.” In: *Proceedings of Interact*. (Edinburgh, Scotland, Aug. 30–Sept. 3, 1999). 1999, pp. 613–620 (cit. on p. 315).
- [Www] *Proceedings of the 15th international World Wide Web conference (WWW)*. (Edinburgh, Scotland). ACM Press, 2006 (cit. on pp. 454, 501).
- [Yah] *Yahoo! Pipes*. URL: <http://pipes.yahoo.com> (visited on 2009-10-22) (cit. on p. 241).
- [You05] Abdou Youssef. “Search of Mathematical Contents: Issues and Methods.” In: *Proceedings of the ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE)*. Ed. by Richard T. Hurley and Wenying Feng. July 2005, pp. 100–105. ISBN: 1-880843-55-2 (cit. on p. 224).
- [YWK09] Takashi Yamamiya, Alessandro Warth, and Ted Kaehler. “Active Essays on the Web.” In: *7th Annual International Conference on Creating, Computing, Connecting, and Collaborating through Computing*. (Kyoto University, Kyoto, Japan, Jan. 2009). 2009 (cit. on p. 353).
- [Zac07] Stefano Zacchiroli. “User Interaction Widgets for Interactive Theorem Proving.” PhD thesis. Università di Bologna, 2007 (cit. on p. 19).

- [zAg] zAgile. *Wikidsmart for Atlassian Confluence*. URL: <http://www.zagile.com/products/wikidsmart.html> (visited on 2010-10-03) (cit. on p. 283).
- [Zal10] Michal Zalewski. *Browser Security Handbook*. Tech. rep. Google, 2010. URL: <http://browsersec.googlecode.com> (visited on 2010-09-21) (cit. on p. 386).
- [Zbl] *Zentralblatt MATH*. URL: <http://www.zentralblatt-math.org> (visited on 2011-03-03) (cit. on p. 11).
- [Zhi] Nikita Zhiltsov. *Mocassin. Mathematical Semantic Search Engine*. URL: <http://mocassin.googlecode.com> (visited on 2010-09-22) (cit. on p. 227).
- [Zho+] Vyacheslav Zholudev et al. *TNTBase – Integration*. URL: <http://tntbase.org/wiki/Integration> (visited on 2010-09-29) (cit. on pp. 272, 278).
- [Zimo8] Jürgen Zimmer. “MathServe – A Framework for Semantic Reasoning Services.” PhD thesis. Universität des Saarlandes, July 2008 (cit. on pp. 18, 356).
- [Zim10] Antoine Zimmermann. “Ontology Recommendations for the Data Publishers.” In: [dGCL+10]. URL: <http://ceur-ws.org/Vol-596/> (cit. on p. 20).
- [Zino4] Claus Zinn. “Understanding Informal Mathematical Discourse.” PhD thesis. Technischen Fakultät der Universität Erlangen-Nürnberg, 2004 (cit. on p. 39).
- [ZK09] Vyacheslav Zholudev and Michael Kohlhase. “TNTBase: a Versioned Storage for XML.” In: *Proceedings of Balisage: The Markup Conference*. Vol. 3. Balisage Series on Markup Technologies. Mulberry Technologies, Inc., 2009. DOI: [10.4242/BalisageVol3.Zholudev01](https://doi.org/10.4242/BalisageVol3.Zholudev01) (cit. on pp. 225, 388).
- [ZK10] Vyacheslav Zholudev and Michael Kohlhase. “Scripting Documents with XQuery: Virtual Documents in TNTBase.” In: *Proceedings of Balisage: The Markup Conference*. Vol. 5. Balisage Series on Markup Technologies. Mulberry Technologies, Inc., 2010. DOI: [10.4242/BalisageVol5.Zholudev01](https://doi.org/10.4242/BalisageVol5.Zholudev01). URL: <http://www.balisage.net/Proceedings/vol5/cover.html> (cit. on pp. 225, 278).
- [ZKR10] Vyacheslav Zholudev, Michael Kohlhase, and Florian Rabe. “A [insert XML Format] Database for [insert cool application].” In: *Proceedings of XML Prague 2010*. 2010, pp. 317–339. URL: <http://kwarc.info/vzholudev/pubs/XMLPrague.pdf> (cit. on p. 278).
- [Ado] Adobe Systems Inc., ed. *Adobe Flash Platform*. URL: <http://www.adobe.com/flashplatform/> (visited on 2010-09-19) (cit. on p. 386).
- [Adv] Advanced Knowledge Technologies (AKT), ed. *acm.rkbexplorer.com*. URL: <http://acm.rkbexplorer.com> (visited on 2010-11-30) (cit. on p. 20).
- [Ame] American Mathematical Society. *MathSciNet Mathematical Reviews on the Net*. URL: <http://www.ams.org/mathscinet/> (visited on 2010-08-05) (cit. on p. 11).
- [Apa] Apache Software Foundation. *Apache Subversion*. URL: <http://subversion.apache.org/> (visited on 2009-10-22) (cit. on pp. 225, 277, 289).
- [Apab] Apache Software Foundation. *Apache Velocity*. URL: <http://velocity.apache.org> (visited on 2010-09-30) (cit. on p. 394).

- [Apac] Apache Software Foundation. *Lucene*. URL: <http://lucene.apache.org/> (visited on 2010-05-04) (cit. on pp. 224, 284, 311).
- [Cre] Creative Commons, ed. *Creative Commons*. URL: <http://creativecommons.org> (visited on 2010-07-20) (cit. on p. 8).
- [DCMo5] DCMI Usage Board. *MARC Relator terms and Dublin Core*. Tech. rep. Dublin Core Metadata Initiative, Dec. 8, 2005. URL: <http://dublincore.org/usage/documents/relators/> (cit. on pp. 45, 124, 126).
- [DCMo8] DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, Jan. 14, 2008. URL: <http://dublincore.org/documents/2008/01/14/dcmi-terms/> (cit. on pp. 46, 128).
- [Danoo] Ron Daniel Jr. *Harvesting RDF Statements from XLinks*. W3C Note. World Wide Web Consortium (W3C), Sept. 29, 2000. URL: <http://www.w3.org/TR/2000/NOTE-xlink2rdf-20000929/> (cit. on p. 81).
- [IEEo2a] IEEE Learning Technology Standards Committee. *Standard for Learning Object Metadata*. Tech. rep. 1484.12.1. IEEE, 2002 (cit. on p. 48).
- [IEEo2b] IEEE Learning Technology Standards Committee. *Standard for Resource Description Framework (RDF) binding for Learning Object Metadata data model*. Tech. rep. 1484.12.4. IEEE, 2002 (cit. on p. 124).
- [Int] International Mathematical Union (IMU), ed. *Math-Net. an International Information and Communication System*. URL: <http://www.math-net.org> (visited on 2010-10-07) (cit. on p. 16).
- [Mic] Microsoft Corporation. *MSDN Library – Development Tools and Languages – Visual Studio 2008 – Visual Studio – Visual C++ – Visual C++ Reference – Visual C++ Libraries Reference – MFC – MFC Concepts – MFC COM – Active Document Containment – Active Documents*. URL: <http://msdn.microsoft.com/en-us/library/bx9c54kf.aspx> (visited on 2009-10-22) (cit. on p. 239).
- [Moz] Mozilla Labs. *Ubiquity*. URL: <http://mozillalabs.com/ubiquity> (visited on 2010-09-19) (cit. on p. 241).
- [Nat10] National Institute of Standards and Technology, ed. *Digital Library of Mathematical Functions*. May 7, 2010. URL: <http://dlmf.nist.gov> (cit. on pp. 11, 355).
- [ODRa] ODRL Initiative. *ODRL Version 2 Development Working Group*. URL: <http://odrl.net/2.0/> (visited on 2010-05-19) (cit. on p. 124).
- [ODRb] ODRL Initiative. *The ODRL Initiative. An Open Rights Language For the Digital Commons*. URL: <http://odrl.net> (visited on 2010-05-19) (cit. on p. 47).
- [Opea] OpenLink Software, ed. *Ping the Semantic Web*. URL: <http://pingthesemanticweb.com> (visited on 2010-08-15) (cit. on p. 100).
- [Opeb] OpenMath Society. URL: <http://www.openmath.org/standard/omxsl/> (visited on 2010-08-09) (cit. on p. 72).

- [Ora] Oracle (formerly Sun Microsystems), ed. *Community Equity*. URL: <http://community-equity.org> (visited on 2011-01-29) (cit. on p. 310).
- [PAU] PAUX Technologies GmbH, ed. *PAUX Technologies*. URL: <http://pauX.de> (visited on 2010-10-10) (cit. on p. 283).
- [Rec] Recordare LLC, ed. *MusicXML 2.0 Specification*. URL: <http://www.recordare.com/musicxml/specification> (visited on 2010-11-26) (cit. on p. 351).
- [The] The Dojo Foundation, ed. *The Dojo Toolkit*. URL: <http://dojotoolkit.org/> (visited on 2010-10-05) (cit. on p. 398).
- [Theo2] The W3C HTML Working Group. *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) – A Reformulation of HTML 4 in XML 1.0*. W3C Recommendation. World Wide Web Consortium (W3C), Aug. 1, 2002. URL: <http://www.w3.org/TR/2002/REC-xhtml1-20020801> (cit. on p. 55).
- [W3Co3] W3C Math Working Group. June 26, 2003. URL: <http://www.w3.org/Math/XSL/Overview-tech.html> (visited on 2010-08-09) (cit. on p. 72).
- [Wik] Wikimedia Foundation, ed. *Wikipedia, the free encyclopedia*. URL: <http://www.wikipedia.org> (visited on 2010-08-11) (cit. on p. 282).
- [Wik09a] Wikimedia Foundation, ed. *Help: Edit summary*. From *Wikipedia, the free encyclopedia*. Nov. 2, 2009. URL: [http://en.wikipedia.org/w/index.php?title=Help:Edit\\_summary&oldid=323403778](http://en.wikipedia.org/w/index.php?title=Help:Edit_summary&oldid=323403778) (visited on 2010-08-11) (cit. on p. 286).
- [Wik09b] Wikimedia Foundation, ed. *Knowledge management*. From *Wikipedia, the free encyclopedia*. Dec. 2, 2009. URL: [http://en.wikipedia.org/w/index.php?title=Knowledge\\_management&oldid=329227520](http://en.wikipedia.org/w/index.php?title=Knowledge_management&oldid=329227520) (visited on 2010-08-11) (cit. on p. 6).
- [Wik09c] Wikimedia Foundation, ed. *Portal: Mathematics*. From *Wikipedia, the free encyclopedia*. Dec. 2, 2009. URL: <http://en.wikipedia.org/w/index.php?title=Portal:Mathematics&oldid=329137789> (visited on 2010-08-11) (cit. on p. 14).
- [Wik09d] Wikimedia Foundation, ed. *Pythagorean theorem*. From *Wikipedia, the free encyclopedia*. Nov. 29, 2009. URL: [http://en.wikipedia.org/w/index.php?title=Pythagorean\\_theorem&oldid=328597679](http://en.wikipedia.org/w/index.php?title=Pythagorean_theorem&oldid=328597679) (visited on 2010-08-11) (cit. on p. 15).
- [Wik09e] Wikimedia Foundation, ed. *Wikipedia: Neutral point of view*. From *Wikipedia, the free encyclopedia*. Oct. 28, 2009. URL: [http://en.wikipedia.org/w/index.php?title=Wikipedia:Neutral\\_point\\_of\\_view&oldid=322591480](http://en.wikipedia.org/w/index.php?title=Wikipedia:Neutral_point_of_view&oldid=322591480) (visited on 2010-08-11) (cit. on p. 286).
- [Wik09f] Wikimedia Foundation, ed. *Wikipedia: No original research*. From *Wikipedia, the free encyclopedia*. Nov. 1, 2009. URL: [http://en.wikipedia.org/w/index.php?title=Wikipedia:No\\_original\\_research&oldid=323200797](http://en.wikipedia.org/w/index.php?title=Wikipedia:No_original_research&oldid=323200797) (visited on 2010-08-11) (cit. on p. 286).
- [Wik09g] Wikimedia Foundation, ed. *Wikipedia: Talk page*. From *Wikipedia, the free encyclopedia*. Nov. 2, 2009. URL: [http://en.wikipedia.org/w/index.php?title=Wikipedia:Talk\\_page&oldid=323514011](http://en.wikipedia.org/w/index.php?title=Wikipedia:Talk_page&oldid=323514011) (visited on 2010-08-11) (cit. on p. 286).

- [Wik10a] Wikimedia Foundation, ed. *Human Development Index*. From *Wikipedia, the free encyclopedia*. Aug. 22, 2010. URL: [http://en.wikipedia.org/w/index.php?title=Human\\_Development\\_Index&oldid=380378550](http://en.wikipedia.org/w/index.php?title=Human_Development_Index&oldid=380378550) (visited on 2010-08-11) (cit. on p. 207).
- [Wik10b] Wikimedia Foundation, ed. *Linked Data*. From *Wikipedia, the free encyclopedia*. June 18, 2010. URL: [http://en.wikipedia.org/w/index.php?title=Linked\\_Data&oldid=368472510](http://en.wikipedia.org/w/index.php?title=Linked_Data&oldid=368472510) (visited on 2010-08-11) (cit. on p. 53).
- [Wik10c] Wikimedia Foundation, ed. *Open Notebook Science*. From *Wikipedia, the free encyclopedia*. July 14, 2010. URL: [http://en.wikipedia.org/w/index.php?title=Open\\_Notebook\\_Science&oldid=372235042](http://en.wikipedia.org/w/index.php?title=Open_Notebook_Science&oldid=372235042) (visited on 2010-08-11) (cit. on p. 13).
- [Wik10d] Wikimedia Foundation, ed. *Wiki*. From *Wikipedia, the free encyclopedia*. Oct. 3, 2010. URL: <http://en.wikipedia.org/w/index.php?title=Wiki&oldid=388369282> (visited on 2010-08-11) (cit. on p. 282).
- [Wik10e] Wikimedia Foundation, ed. *Wikipedia: Assume good faith*. From *Wikipedia, the free encyclopedia*. Oct. 16, 2010. URL: [http://en.wikipedia.org/w/index.php?title=Wikipedia:Assume\\_good\\_faith&oldid=391099044](http://en.wikipedia.org/w/index.php?title=Wikipedia:Assume_good_faith&oldid=391099044) (visited on 2010-08-11) (cit. on p. 322).
- [Wik10f] Wikimedia Foundation, ed. *Wikipedia: Be bold*. From *Wikipedia, the free encyclopedia*. Oct. 21, 2010. URL: [http://en.wikipedia.org/w/index.php?title=Wikipedia:Be\\_bold&oldid=391966034](http://en.wikipedia.org/w/index.php?title=Wikipedia:Be_bold&oldid=391966034) (visited on 2010-08-11) (cit. on p. 322).
- [Wik10g] Wikimedia Foundation, ed. *Wikipedia: Template messages/Cleanup*. From *Wikipedia, the free encyclopedia*. Sept. 29, 2010. URL: [http://en.wikipedia.org/w/index.php?title=Wikipedia:Template\\_messages/Cleanup&oldid=387649325](http://en.wikipedia.org/w/index.php?title=Wikipedia:Template_messages/Cleanup&oldid=387649325) (visited on 2010-08-11) (cit. on p. 286).