PhD Thesis

Building Bridges in Abstraction-Based Controller Synthesis

Advancing, Combining, and Comparing Methods from Computer Science and Control

Anne-Kathrin Schmuck

Building Bridges in Abstraction-Based Controller Synthesis

Advancing, Combining, and Comparing Methods from Computer Science and Control

Vorgelegt von
Dipl.-Ing. Anne-Kathrin Schmuck (geb. Hess)
aus Erfurt

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktorin der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr.-Ing. Uwe Nestmann
Gutachter:	Prof. Dr.-Ing. Jörg Raisch
Gutachter:	Prof. Dr. Paulo Tabuada
Gutachter:	Prof. Dr.-Ing. Thomas Moor

Tag der wissenschaftliche Aussprache: 18. September 2015

Berlin 2015

PhD Thesis

Building Bridges in Abstraction-Based Controller Synthesis

Advancing, Combining, and Comparing Methods from Computer Science and Control

Anne-Kathrin Schmuck

Anne-Kathrin Schmuck: *Building Bridges in Abstraction-Based Controller Synthesis*
PhD Thesis, © Berlin 2015

Supervisor:
Prof. Dr.-Ing. Jörg Raisch

Location:
Fachgebiet Regelungssysteme, Fakultät IV - Elektrotechnik und Informatik, Technische Universität Berlin, Einsteinufer 17, 10587 Berlin, Germany

ISBN: 978-3-7375-7174-6
www.epubli.de

Abstract

Abstraction based controller synthesis is a well established two-step procedure to solve complex control problems involving discrete valued quantities. First, a symbolic abstraction of the system to be controlled is generated providing a discrete time model with a finite, discrete valued signal space and finitely many states. Second, a symbolic controller for a desired symbolic specification is constructed using the previously generated symbolic model. This controller synthesis approach is usually used in two different settings. Either (i) the specification is naturally given by a Linear Temporal Logic (LTL) or a Computation Tree Logic (CTL) formula over a finite set of symbols, which can only be encountered by symbolic controller synthesis techniques. Or (ii) the system to be controlled is naturally equipped with a finite set of external symbols through which it interacts with its environment, e.g., the controller.

For each of the two settings specific approaches to solve the synthesis problem have been proposed independently. In Part I of this thesis we will investigate the abstraction step of two different approaches, namely *quotient based abstractions* (QBA) and *strongest asynchronous l-complete approximations* (SAlCA), tailored to setting (i) and (ii), respectively. It will be shown that the resulting abstractions are generally incomparable. We will therefore derive necessary and sufficient conditions on the original system which allow for a detailed comparison. This builds our first bridge between the computer science community, which inspired the construction of QBA and the control systems community, where SAlCA were developed.

When the second setting is considered where the use of abstraction based controller synthesis is motivated by the symbolic input-output structure of the system, the desired specification might not naturally be symbolic. However, to apply supervisory control theory (SCT), a framework for symbolic controller synthesis commonly used in combination with SAlCA, the specification is required to be modelled by a deterministic finite automaton (DFA). This motivates the investigation of larger specification classes to enrich the applicability of abstraction based controller synthesis in setting (ii). In Part II of this thesis we show that SCT can be extended to handle specifications realized by deterministic pushdown automata (DPDA). This builds our second bridge between the computer science community, where different automata models, such as DFA and DPDA, are formalized and the control systems community, where SCT was developed.

Zusammenfassung

Der abstraktionsbasierte Reglerentwurf ist eine bekannte, zweistufige Entwurfsmethode, bei welcher zuerst eine Abstraktion des zu regelnden Systems konstruiert wird, welche diskretwertige externe Signale und eine endliche Zustandsmenge besitzt. Danach wird mit Hilfe dieser Abstraktion für eine gewünschte Spezifikation ein ereignisdiskreter Regler entworfen. Diese Reglersynthese wird üblicherweise in zwei verschiedenen Szenarien angewendet. Entweder ist die Spezifikation durch einen Logik-Ausdruck über einer endlichen Menge von Symbolen gegeben, welche nur durch symbolische (d.h. ereignisdiskrete) Reglersyntheseverfahren behandelt werden kann. Oder das zu regelnde System besitzt nur diskretwertige Mess- und Stellsignale, die nur die Interaktion mit einem ereignisdiskreten Regler erlauben.

Für jedes dieser beiden Szenarien wurden unabhängig voneinander verschiedene Ansätze für die abstraktionsbasierte Reglersynthese entwickelt. Im ersten Teil dieser Arbeit vergleichen wir den Abstraktionsschritt zweier Ansätze, nämlich die *quotientenbasierte Abstraktion* (QBA) und die *strengste asynchrone l-vollständige Approximation* (engl. strongest asynchronous l-complete approximations (SAlCA)). Es wird sich zeigen, dass die resultierenden Abstraktionen generell nicht vergleichbar sind. Wir geben deshalb notwendige und hinreichende Bedingungen für das Originalsystem an, welche einen direkten Vergleich ermöglichen. Damit bauen wir die erste Brücke zwischen dem Gebiet der Informatik, durch welche die Konstruktion der QBA beeinflusst wurde, und dem Gebiet der Regelungstechnik, in dem SAlCA entwickelt wurden.

Wenn im abstraktionsbasierten Reglerentwurf das zweite Szenario vorliegt, d.h., das zu regelnde System nur diskretwertige Mess- und Stellsignale besitzt, ist die Spezifikation meist nicht in einer einheitlichen Form gegeben. In Verbindung mit SAlCA wird der Reglerentwurf meist unter Verwendung der Supervisory Control Theory (SCT) durchgeführt. Allerdings kann die SCT nur angewendet werden, wenn die Spezifikation als deterministischer endlicher Automat (engl. deterministic finite automaton (DFA)) modelliert werden kann. Um die Anwendbarkeit dieser Entwurfsmethode zu verbessern, wird die SCT im zweiten Teil dieser Arbeit erweitert, um auch Spezifikationen in Form von deterministischen Kellerautomaten (engl. deterministic pushdown automata (DPDA)) behandeln zu können. Damit bauen wir die zweite Brücke zwischen dem Gebiet der Informatik, in dem verschiedene Automatenmodelle, wie DFA und DPDA, formalisiert wurden, und dem Gebiet der Regelungstechnik, in dem SCT entwickelt wurde.

Contents

Introduction

The increasing interconnection of physical components and digital hardware in today's engineering systems causes challenges that have been focused on both by the control and the computer science community. Although some efforts have been made to bring these parallel advances together there are still considerable gaps between concepts in both fields addressing very similar questions. Closing these gaps seems to be a promising way towards new solutions in this field.

In this thesis we build two different bridges between results obtained by the control and the computer science community in the broad field of abstraction-based controller synthesis. The common idea of the latter is to simplify a given complex (possibly not directly solvable) control problem by generating a symbolic abstraction of the system to be controlled. This allows to apply controller synthesis techniques developed for symbolic system models.

In Part I a bridge is built between two different techniques constructing symbolic abstractions, namely *quotient based abstractions* (QBA) and *strongest asynchronous l-complete approximations* (SAlCA). The construction of QBA is motivated by a problem setting where a system should obey a specification given in terms of a linear temporal logic (LTL) or computational tree logic (CTL) formula over a finite set of symbols, e.g., "always eventually visit region A". Inspired by techniques developed in the computer science community for verification and synthesis of software processes, QBA are generated by partitioning the state space of the original system into a finite number of cells. The obtained set of equivalence classes of this partition is then used as the set of states and the set of outputs of the QBA. This implies that the used output symbols are generated artificially usually based on the given specification and can be used as a degree of freedom in the abstraction process to adjust the abstraction accuracy.

Contrary to this viewpoint, SAlCA are tailored to handle systems where the available interface for control is symbolic. Hence, the construction of a symbolic abstraction is motivated by limited sensing (e.g., a sensor that can only detect threshold crossings) and/or limited actuation (e.g., a valve that can only be fully opened or closed). In this case, contrary to the setting for QBA, the set of external symbols that can be used to construct the abstraction is predefined by the system. To increase abstraction accuracy when constructing SAlCA the number l of past input and output symbols considered in the construction of the abstract state space can be increased.

While QBA and SAlCA are using a similar approach to tackle the abstraction based controller synthesis problem the constructions used are substantially different. To formally compare both approaches we investigate how the two outlined scenarios for QBA and SAlCA can be incorporated in a unified setup. For this unified setup we can show that the abstractions resulting from QBA and SAlCA are generally incomparable when assuming a predefined set of output symbols. We therefore derive necessary and sufficient conditions on the original system that allow to relate the resulting abstractions. Surprisingly, we can show that for the scenario considered by QBA those conditions are automatically satisfied and the construction of SAlCA and QBA results in bisimilar abstractions.

In the setting for SAlCA the use of abstraction based controller synthesis is motivated by the input/output structure of the plant. Hence the specification used for controller synthesis might not have a particular symbolic structure. However, to apply supervisory control theory (SCT), a popular symbolic controller synthesis technique, both the system and the specification need to be representable by a deterministic finite automaton (DFA). While both QBA and SAlCA can typically be realized by DFA requiring the same for the specification may be restrictive.

To extend the applicability of abstraction based controller synthesis in the presence of a symbolic controller interface Part II contributes to a generalization of SCT to specifications that can be realized by deterministic pushdown automata (DPDA). This builds a bridge between automata theoretic concepts from the computer science community and the controller synthesis technique of SCT.

As the classical algorithm to solve the supervisory control problem (SCP) for DFA does not readily carry over to the case where the specification is described by a DPDA we first introduce a new conceptual iterative algorithm to solve the SCP. This algorithm is composed of two subroutines, i.e., (i) enforcing controllability least restrictively for DPDA and (ii) ensuring nonblockingness of DPDA. While an implementable algorithm for (ii) was given in [60] we derive an implementable algorithm for (i) in Part II. The implementation of the overall algorithm solving the SCP for specifications representable by DPDA is available as a plug-in in `libFAUDES` [31].

The research which lead to this thesis was conducted in cooperation with my supervisor, Jörg Raisch, as well as Paulo Tabuada (Part I) and Sven Schneider (Part II). It was also inspired by discussions with Thomas Moor, Uwe Nestmann, Matthias Rungger and my colleagues Vladislav Nenchev and Behrang Monajemi Nejad. I am very thankful for their honest, valuable and critical feedback on my work and their willingness to cooperate and to share their view of abstraction based controller synthesis with me, which shaped my view of this field.

Part I.

A Bridge Between Strongest Asynchronous l-Complete Approximations and Quotient-Based Abstractions

1. Literature Review and Contributions I

In Part I two different techniques for constructing symbolic abstractions, namely *quotient based abstractions* (QBA) and *strongest asynchronous l-complete approximations* (SAlCA), are compared. To illustrate the conceptual differences of both approaches we discuss the historical background of both methods in Section 1.1 and Section 1.2. Thereafter, in Section 1.3 the steps of our subsequent bridging endeavor are outlined and the contributions of Part I are emphasized. Chapter 1 is concluded by a list of publications by the author where some results contained in Part I have appeared previously.

1.1. Quotient Based Abstractions

The construction of QBA originates from research activities in the computer science community on verification and synthesis of software processes. Given a property in linear temporal logic (LTL) or computational tree logic (CTL) and an automaton model of the environment, this work is focused on synthesizing a process which satisfies the specified property despite the behavior of the environment.

These ideas were first transfered to hybrid systems, i.e., systems involving continuous and discrete-valued signals, by Alur and Dill [1]. They showed that a very simple hybrid system, namely a timed automaton, allows for a symbolic abstraction. In particular, they derived a bisimulation relation, i.e., an equivalence relation ensuring that properties formulated in LTL or CTL are preserved [34] between the timed automata and its abstraction. Hence, verification and synthesis techniques can be applied to timed automata by using their bisimilar symbolic abstractions.

The work of Alur and Dill was extended to other classes of hybrid systems in [2] and inspired other researchers to extend the notion of bisimulation to other system classes, e.g. linear continuous systems [29, 41, 75], affine continuous systems [70], linear discrete-time systems [69], switched linear systems [44] , general flow systems [8] or behavioral systems [27]. While [41, 75, 70, 44, 8, 27] are investigating bisimilarity of system models of the same type for the purpose of complexity reduction, the goal of the work conducted in [1, 2, 29, 69], was to construct bisimilar *symbolic* models from other models to apply verification or synthesis techniques.

These advances led to the considered abstraction based controller synthesis, where based on a given LTL or CTL specification, (i) a (bisimilar) symbolic abstraction is calculated, (ii) a symbolic controller is synthesized, (iii) this controller is refined to be applicable to the original system. Thereby, the closed loop system is ensured to be correct by design, i.e., the original model connected to the refined controller fulfills the specification. As steps (i) and (iii) are highly dependent on the dynamics of the original model, the overall problem was only solved for special system classes, e.g., linear continuous systems [68, 63], multi-affine control systems [4] or piecewise-affine hybrid systems [21]. These results are nicely summarized in [67, Part II]. Additionally, [22, 64, 66] use category theory to unify this synthesis on a theoretical level.

In the contributions contained in [67, Part II] step (i) of the outlined synthesis is realized by partitioning the original signal space into a (finite) set of equivalence classes such that this partition allows for a bisimulation relation between the original model and its abstraction. The set of equivalence classes of this partition is then used as discrete states and output symbols of the constructed symbolic abstraction. Hence, the used output symbols are generated artificially, usually based on the given specification, and can be used as a degree of freedom in the abstraction process to adjust the abstraction accuracy.

However, the conditions for the existence of a partition allowing for a bisimilar abstraction are rather restrictive. Therefore, several researchers have also investigated the construction of approximately bisimilar symbolic abstractions [13] for controller synthesis, e.g., [65, 79, 15, 30, 14, 80], partially summarized in [67, Part III]. Here, instead of a partition, a cover of the state space is constructed and equivalence of systems is only investigated with respect to a deviation parameter ϵ. However, this research is also guided by the general idea of defining the set of output symbols artificially based on a given specification and required precision ϵ.

1.2. Strongest l-Complete Approximations

Mostly independently from the line of research outlined in the previous section abstraction based controller synthesis was also developed for a different problem setting first considered in [3] and [32]. Here, a system with possibly continuous dynamics is considered which interacts with the environment through discrete valued signals. This is for example the case if the input signal can only take a finite number of values due to actuator limitations (e.g. a valve which can only be "open" or "closed"), or if the measurements are quantized (e.g., the measurement device only detects if the temperature in a reactor is getting "too cold", "too warm", or returns to be "OK").

In this setting the construction of a symbolic abstraction for controller synthesis is motivated by the fact that the interface between the system and the controller is symbolic. This implies that, contrary to the setting for QBA, the set of external symbols is predefined by the abstracted system and cannot be used as a degree of freedom in the abstraction process.

One method explicitly addressing the problem of an adaptable abstraction accuracy in the presence of a predefined set of symbols is the so called strongest l-complete approximation (SlCA) [45, 46, 36]. This abstraction exactly mimics the strings of symbols generated by the system over finite time intervals of length $l+1$. Contrary to the work in [32, 3] this allows for increasing the approximation accuracy by increasing the length of the considered intervals.

The construction of SlCA is based on behavioral system theory [76, 77] and was originally motivated by the property of l-completeness denoting that the behavior of a system, i.e., the set of all infinite sequences of symbols compatible with its evolution, can be reconstructed by appending strings of symbols of length $(l+1)$ in a particular way. This idea is transferred to SlCA by taking the strings of length $(l+1)$ from the behavior of the original system and constructing the abstract behavior by appending them such that the latter becomes l-complete. Hence, the SlCA of a system is given as a behavior, i.e., as a set of infinite sequences of symbols. Generally this allows for many different automaton realizations of the latter, which is desirable for control purposes. However, almost all previous literature on SlCA makes only use of a particular realization naturally arising in the abstraction process. This realization uses the set of l-long strings of external symbols as its state space. Therefore, SlCA can be realized by a finite state symbolic model if the set of symbols is finite, which is usually assumed in the considered setting.

Using behavioral system theory to formalize the construction of SlCA has the advantage that the construction of an abstraction can be formalized without reference to a specific realization, comparable to the idea of using category theory to unify the quotient based approach in [22, 64, 66]. However, this also implies that the application of this approach to a real model is not straightforward. It was for example shown in [12, 37] and [50] how the construction of SlCA can be applied to continuous linear time invariant systems and nonlinear discrete-time systems, respectively. However, as for QBA, these concrete solutions do not translate to other system classes.

The idea of using l-long strings of symbols as abstract states was also used for constructing abstractions of incrementally stable switched systems in [30] and incrementally stable stochastic control systems in [80]. However, [30, 80] use l-long sequences of modes rather then input and output symbols as abstract states and no symbolic controller interface is assumed. Interestingly, the abstractions constructed in [30, 80] are based on (approximated versions of) QBA but employ ideas from SlCA in a quite different context. Contrary, the recent work by Tarraf [72, 71] uses the same setting as SlCA and also constructs abstractions using l-long strings of input and output symbols. However, the actual abstraction procedure is guided by a subsequently employed optimization based controller synthesis and is therefore substantially different from SlCA.

1.3. Comparison and Contributions

Using QBA and SlCA exactly as defined in existing work does not directly allow for a comparison. We therefore need to revisit both constructions before we can draw formal connections between the two.

In Chapter 2 we first consider SlCA which are only constructed for time invariant systems in [36] and subsequent papers. The term "time invariant" is thereby interpreted in a behavioral way, i.e., refers to systems that are invariant with respect to the backward time shift of signals. As this assumption is not made for QBA we extend the construction of SlCA to not necessarily time invariant systems based on our previous work in [53]. To obtain a symbolic abstraction which is still suitable for symbolic controller synthesis we ensure that this new abstraction, referred to as *strongest asynchronous l-complete approximation* (SAlCA), can still be realized by a finite state symbolic model.

During the derivation of SAlCA in Chapter 2 we additionally resolve an inconsistency on the definition of the l-completeness property in [36, 35] which is slightly stronger than the original definition by J.C.Willems in [76]. Furthermore, to formalize the term *asynchronous* in SAlCA, we introduce asynchronous versions of the well-known concepts of state property, memory span, and l-completeness. This extends the behavioral systems theory in a consistent way.

As SAlCA are formalized using behavioral system theory both the system to be abstracted and the resulting abstraction are given by a behavior, i.e., a set of infinite sequences of symbols. Contrary, in the construction of QBA the original and the abstract system are usually modelled by a so called transition system, which is formalized in Chapter 3. While these two setups are substantially different we will show in Chapter 4 that they can be incorporated under mild assumptions, by modeling the original system as a so called *asynchronous state space ϕ-dynamical system* (ASSϕDS), which we have introduced in [54]. This allows to define a common "original" model for the construction of QBA and SAlCA.

The behavior of an SAlCA is constructed by appending strings of length $l + 1$ from the behavior of the original system in a particular way such that the resulting abstract behavior is l-complete and a superset of the original behavior. Equality of both behaviors only holds if the original system is l-complete. Hence, the construction of SAlCA and its relation to the original system is based on behavioral inclusion and equivalence. Contrary, the construction of QBA was motivated by the need to obtain a bisimilar symbolic model of the original one to ensure that properties formulated in LTL or CTL are preserved. While the existence of a bisimulation relation implies behavioral equivalence, the inverse implication does usually not hold.

In the spirit of our bridging endeavor it is therefore interesting to investigate when the stronger property of bisimilarity also holds between the original system, modeled by an ASSϕDS, and its SAlCA, given by a behavioral system. To formally investigate this we derive a notion of (bi)similarity for ASSϕDS in Chapter 5, which is inspired by [27, 26, 8] and based on previous results in [52]. To also relate other system models we furthermore show in Chapter 5 that ASSϕDS can be used as a unifying modeling framework. Hence, the notion of (bi)simulation relations for ASSϕDS can be directly transferred to other system models, e.g., transition systems or behavioral systems. Besides the formal beauty of this result, it shows that (bi)simulation relations for ASSϕDS can be used to relate different system models (e.g., a transition system to a behavioral model).

Equipped with a unified model of the original system and a notion of similarity between different system models we now seem to be ready to compare SAlCA and QBA. However, there are two additional gaps resulting from the different problem setups considered by QBA and SAlCA that have to be closed first.

As discussed previously, the construction of QBA typically starts with partitioning the state space into a finite set of equivalence classes which is used as the set of discrete outputs and the set of abstract states of the resulting QBA. The calculation of a suitable partition is typically done iteratively. First, an initial partition of the state space is chosen, e.g., induced by the LTL or CTL specification. Then a refinement algorithm [11] is run which terminates if the partition allows to construct a QBA which is bisimilar to the original system. With every step the partition of the state space is refined until the algorithm terminates. Therefore, the construction of the set of output symbols (being the equivalence classes of the final partition) is already part of the abstraction process when using QBA.

Conceptually, there is a strong correspondence of this iterative calculation of the set of output symbols in the construction of QBA and increasing the parameter l when constructing realizations of SAlCA. However, there are two main differences. (i) While the state space of QBA is given by the set of output symbols, the state space of SAlCA is constructed from the whole set of external symbols, i.e., inputs and outputs. (ii) The set of output symbols is predefined in the setting for SAlCA. Hence, the cover of the state space is refined by changing l instead of changing the set of output symbols.

The key in understanding the formal and conceptual differences between QBA and SAlCA lies (i) in the incorporation of different choices of symbols used to construct the abstract state space of realizations of SAlCA and (ii) in the incorporation of the iterative calculation of the set of output symbols in the construction of QBA. These two points are addressed in Chapter 6 and Chapter 7, respectively, resulting in a set of new realizations for SAlCA and a a chain of QBA with increasing precision which previously appeared in [57, 58].

Using these new abstract models resulting from SAlCA and QBA we can finally formalize a comparison between the two in Chapter 8 which is based on [57, 58]. However, as outlined previously, the assumptions about the set of output symbols are fundamentally different in both settings. We therefore obtain a comparison for both viewpoints separately.

First, we take the viewpoint of SAlCA and assume a predefined set of output symbols. Surprisingly, this results in generally incomparable abstractions. We will therefore derive necessary and sufficient conditions in terms of properties on the original system which ensure that both abstractions become bisimilar. These conditions will turn out to be rather restrictive.

Thereafter, we take the viewpoint of QBA and assume that the output event set can be chosen arbitrarily. Hence, we assume that the refinement algorithm is run prior to the construction of SAlCA and QBA. In this special case we can show that both abstractions coincide. Hence, in this second set up the necessary and sufficient conditions derived previously are satisfied automatically.

1.4. Publications

In the following publications by the author some results contained in Part I have appeared previously or are currently under review.

[52] Anne-Kathrin Schmuck and Jörg Raisch. Constructing (bi)similar finite state abstractions using asynchronous l-complete approximations. In *Proc. 53rd Annual Conference on Decision and Control*, 2014.

[54] Anne-Kathrin Schmuck and Jörg Raisch. Simulation and bisimulation over multiple time scales in a behavioral setting. In *Proc. 22nd Mediterranean Conf. on Control and Automation, Palermo, Italy*, 2014.

[53] Anne-Kathrin Schmuck and Jörg Raisch. Asynchronous l-complete approximations. *Systems & Control Letters*, 73(0):67 – 75, 2014.

[57] Anne-Kathrin Schmuck, Paulo Tabuada, and Jörg Raisch. Comparing asynchronous l-complete approximations and quotient based abstractions. *Proc. 54rd Annual Conference on Decision and Control*, 2015. (to appear).

[58] Anne-Kathrin Schmuck, Paulo Tabuada, and Jörg Raisch. Comparing asynchronous l-complete approximations and quotient based abstractions. *IEEE Transactions on Automatic Control*, 2015. (under review, preprint available at http://arxiv.org/abs/1503.07139).

2. From SlCA to SAlCA

Behavioral systems theory was developed by J.C.Willems [76, 77] in the late 80ies as an alternative concept to model dynamical systems. The basic idea is to "provide a mathematical framework for discussing dynamics on a general level, that is, without reference to a specific class of (physical, economic, or engineering) examples" [76, p.171]. In behavioral systems theory a dynamical system is embedded in an "unexplained" environment with which it interacts through certain variables. These variables evolve along some timescale T and take their values in some set W. Obviously, a system can usually produce only a certain subset of trajectories from the set $W^T := \{\omega \mid \omega : T \to W\}$ of all signals over T taking values in W. This subset is called the behavior $\mathcal{B} \subseteq W^T$ of the dynamical system.

Definition 2.1. *A* dynamical system *is a tuple* $\Sigma = (T, W, \mathcal{B})$*, consisting of the time axis* T*, the signal space* W *and the behavior* $\mathcal{B} \subseteq W^T$*.*

Using behaviors to model systems abstracts from the detailed underlying process (e.g., differential equations) and focuses on the behavior itself. This motivated the use of this framework to derive strongest (asynchronous) l-complete approximations (SlCA). The construction of SlCA is based on the property of l-completeness introduced in [76, 77] for dynamical systems evolving over a right- and left-unbounded time scale, e.g., $T = \mathbb{Z}$ or $T = \mathbb{R}$. However, in [36] and subsequent papers, SlCA are only defined for time invariant systems evolving on $\mathbb{N}_0$. As pointed out in [35, p.44], for systems evolving on $\mathbb{N}_0$, the l-completeness property for time invariant systems used in [36, 35] is slightly stronger than the original definition by J.C.Willems in [76]. This implies that the SlCA suggested in [36] is also l-complete in the sense of [76] but not necessarily the *strongest* l-complete approximation in the sense of [76].

The use of the time axis $\mathbb{N}_0$ instead of $\mathbb{Z}$ when constructing SAlCA is motivated by the fact that technical systems are usually started at a particular time and might have a distinct start up phase before resuming a nominal behavior. This distinct initial behavior can be nicely modeled when considering a time axis starting at time $t = 0$, resulting in realizations with distinct initial states. We adapt this viewpoint in this thesis but want to consider systems which are not necessarily time invariant, i.e., systems that are not invariant with respect to the backward time shift of signals. We therefore extend the construction of SlCA to this system class while resolving the aforementioned inconsistency on the property of l-completeness in this chapter, which is based on [53].

After introducing required notation in Section 2.1 we first provide a straightforward extension of SlCA from [36] to the l-completeness definition from [76] in Section 2.2-Section 2.4. We will show in Section 2.5 that, unfortunately, those abstractions do generally not allow for a realization by a finite state machine. However, the latter is necessary to apply existing controller synthesis techniques such as supervisory control theory.

Intuitively, a system is realizable by a state machine if it allows for concatenation of state trajectories that reach the same state asynchronously (i.e., at different times), as used in the context of state maps by Julius and van der Schaft [27, 26]. To emphasize that this property does not imply and is not implied by the time invariance property of behavioral systems we call it *asynchronous state property* and formalize it in Section 2.6. Then we can introduce an *asynchronous l-completeness* property in Section 2.7 since the state and the l-completeness property are strongly related. This leads to a new approximation technique introduced in Section 2.8, which is referred to as *strongest asynchronous l-complete approximation* (SAlCA) and which ensures that the resulting abstraction can be realized by an FSM.

The previously outlined derivation of SAlCA from SlCA in this chapter is accompanied by a formal extension of the well-known concepts of state property, memory span and l-completeness from [76] to (i) a positive time axis $\mathbb{N}_0$ and (ii) an asynchronous interpretation of concatenation. Due to these extensions some well known results in behavioral system theory are proven again for this setting.

2.1. Notation

To simplify notation we derive most properties in this chapter for $T = \mathbb{N}_0$ only, if they are solely used for discrete time dynamical systems $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ in this thesis. However, it should be kept in mind that they analogously hold for other choices of $T \subseteq \mathbb{R}_0^+$. As we also discuss dynamical systems with time axis $T \neq \mathbb{N}_0$ in Chapter 4, we will occasionally use a general time axis $T \subseteq \mathbb{R}_0^+$ to apply the obtained properties in Chapter 4.

For any $l \in \mathbb{N}_0$, $(W)^l := \{\omega \mid \omega : [0, l-1] \to W\}$ denotes the set of strings with length $|\omega|_L = |[0, l-1]| = l$ and elements in W. Now let $\mathcal{I} = [t_1, t_2]$ be a bounded interval on $\mathbb{N}_0$ with length $|\mathcal{I}| = t_2 - t_1 + 1$. Then $\cdot|_{\mathcal{I}} : (W)^{\mathbb{N}_0} \to (W)^{|\mathcal{I}|}$ restricts a map $\omega \in W^{\mathbb{N}_0}$ to the domain $\mathcal{I}$ while disregarding absolute time information, i.e., for $\omega' = \omega|_{\mathcal{I}} = \omega(t_1) \ldots \omega(t_2)$ it holds that $\omega' \in (W)^{|\mathcal{I}|}$ instead of $\omega' \in (W)^{\mathcal{I}}$. Similarly, $\mathcal{B}|_{\mathcal{I}}$ restricts all trajectories in $\mathcal{B}$ to $\mathcal{I}$ while disregarding absolute time information. For $t_1 < t_2$ we define $\omega|_{[t_2,t_1]} := \lambda$, where λ denotes the *empty string* with $|\lambda|_L = 0$.

Now let W, V and $\tilde{V}$ be sets. Then the *projection* of the set W and the symbol $w \in W$

to V is defined by

$$\pi_V(W) := \begin{cases} V & , W{=}V{\times}\tilde{V} \\ W & , W{=}V \\ \emptyset & , \text{else} \end{cases} \quad \text{and} \quad \pi_V(w) := \begin{cases} v & , w{=}(v,\tilde{v}){\in}V{\times}\tilde{V} \\ w & , W{=}V \\ \lambda & , \text{else,} \end{cases}$$

respectively. With this, the projection of a signal $\omega \in W^T$ to V is given by[1]

$$\pi_V(\omega) := \{\nu \in V^T \mid \forall t \in T \,.\, \nu(t) = \pi_V(\omega(t))\}$$

and $\pi_V(\mathcal{B})$ denotes the projection of all signals in the behavior to V. Given two signals $\omega_1, \omega_2 \in W^T$ and two time instants $t_1, t_2 \in T$, the *concatenation* $\omega_3 = \omega_1 \wedge_{t_2}^{t_1} \omega_2$ is given by

$$\forall t \in T \,.\, \omega_3(t) = \begin{cases} \omega_1(t) & , \ t < t_1 \\ \omega_2(t - t_1 + t_2) & , \ t \geq t_1 \end{cases}, \tag{2.1}$$

where $\cdot \wedge_t^t \cdot$ is denoted by $\cdot \wedge_t \cdot$. Furthermore, if $T = \mathbb{N}_0$, the concatenation of their restrictions $\omega_1' = \omega_1|_{[0,t_1]}$ and $\omega_2' = \omega_2|_{[0,t_2]}$ is defined as

$$\omega_1' \cdot \omega_2' := \left(\omega_1 \wedge_0^{t_1+1} \omega_2\right)|_{[0,t_1+t_2+1]}. \tag{2.2}$$

This corresponds to the standard concatenation of finite strings.

2.2. Properties of Dynamical Systems with $T = \mathbb{N}_0$

Properties for dynamical systems where introduced in [76] for the left- and right-unbounded time axis $T = \mathbb{Z}$. As this thesis only considers systems evolving over a nonnegative time axis $T = \mathbb{N}_0$ (occasionally also $T \subseteq \mathbb{R}_0^+$), we restate all necessary properties for this case and derive complementary results to the setting in [76].

Time Invariance

Following [77, Def. II.3], a dynamical system $\Sigma = (T, W, \mathcal{B})$ with nonnegative time axis is said to be *time invariant* if $\sigma\mathcal{B} \subseteq \mathcal{B}$, where σ^t denotes the backward shift operator defined s.t. $\forall t, t' \in T \,.\, (\sigma^t f)(t') := f(t' + t)$. Furthermore, Σ is called *strictly time invariant* if $\sigma\mathcal{B} = \mathcal{B}$.

[1] Throughout this thesis we use the notation "$\forall \cdot \,.\, \cdot$", meaning that all statements after the dot hold for all variables before of the dot. "$\exists \cdot \,.\, \cdot$" is interpreted analogously.

Completeness

When reasoning about dynamical systems with right-unbounded time axis one has to distinguish between local and eventuality properties. Local properties can be evaluated on a finite time interval whereas eventuality properties can only be evaluated after infinite time. Systems whose behavior can be fully described by local properties are called complete.

Definition 2.2 ([77], Def. II.4)**.** *A dynamical system* $\Sigma = (T, W, \mathcal{B})$ *is* complete *if*

$$\big(\forall t_1, t_2 \in T, t_1 \leq t_2 \,.\, \omega|_{[t_1,t_2]} \in \mathcal{B}|_{[t_1,t_2]}\big) \Leftrightarrow \omega \in \mathcal{B}. \tag{2.3a}$$

It is easy to show that for $T = \mathbb{N}_0$ (2.3a) is equivalent to

$$\big(\forall k \in \mathbb{N}_0 \,.\, \omega|_{[0,k]} \in \mathcal{B}|_{[0,k]}\big) \Leftrightarrow \omega \in \mathcal{B}, \tag{2.3b}$$

which is also known as ω-closedness [39]. Using this simplified notation we can define the ω-closure of a dynamical system and state some interesting properties.

Definition 2.3. *Given a dynamical system* $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$*, its* ω*-closure is defined by* $\overline{\Sigma} = (\mathbb{N}_0, W, \overline{\mathcal{B}})$ *s.t.*

$$\overline{\mathcal{B}} = \{\omega \in W^{\mathbb{N}_0} \mid \forall k \in \mathbb{N}_0 \,.\, \omega|_{[0,k]} \in \mathcal{B}|_{[0,k]}\}. \tag{2.4}$$

Furthermore, $\overline{\mathcal{B}}$ *is called the* ω*-closure of* $\mathcal{B}$*.*

Lemma 2.4. *Given a dynamical system* $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ *and its* ω*-closure* $\overline{\Sigma} = (\mathbb{N}_0, W, \overline{\mathcal{B}})$ *it holds that*

(i) $\mathcal{B} \subseteq \overline{\mathcal{B}}$*,*

(ii) $\forall k \in \mathbb{N}_0 \,.\, \mathcal{B}|_{[0,k]} = \overline{\mathcal{B}}|_{[0,k]}$ *and*

(iii) $\mathcal{B} = \overline{\mathcal{B}} \Leftrightarrow \Sigma$ *is complete.*

Proof. We prove all statements separately.

(i) As $\omega \in \mathcal{B}$ implies $\forall k \in \mathbb{N}_0 \,.\, \omega|_{[0,k]} \in \mathcal{B}|_{[0,k]}$ it implies $\omega \in \overline{\mathcal{B}}$ (from (2.4)).

(ii) "$\subseteq$" follows directly from (i). To show "$\supseteq$" we pick $\tilde{\omega} \in \overline{\mathcal{B}}|_{[0,k]}$ implying the existence of $\omega \in \overline{\mathcal{B}}$ s.t. $\omega|_{[0,k]} = \tilde{\omega}$ and therefore $\tilde{\omega} \in \mathcal{B}|_{[0,k]}$ (from (2.4)).

(iii) "$\Rightarrow$": Observe that "$\Leftarrow$" in (2.3b) always holds. To show "$\Rightarrow$" we pick ω s.t. $\forall k \in \mathbb{N}_0 \,.\, \omega|_{[0,k]} \in \mathcal{B}|_{[0,k]}$. Using (2.4) this implies $\omega \in \overline{\mathcal{B}}$. As $\mathcal{B} = \overline{\mathcal{B}}$ we have $\omega \in \mathcal{B}$.
"$\Leftarrow$": $\mathcal{B} \subseteq \overline{\mathcal{B}}$ always holds from (i). To show $\overline{\mathcal{B}} \subseteq \mathcal{B}$ we pick $\omega \in \overline{\mathcal{B}}$. Using (2.4) this implies $\forall k \in \mathbb{N}_0 \,.\, \omega|_{[0,k]} \in \mathcal{B}|_{[0,k]}$. As Σ is complete, this implies $\omega \in \mathcal{B}$.

□

Example 2.1. Consider the dynamical system

$$\Sigma = (\mathbb{N}_0, W, \mathcal{B}) \text{ with } W = \{a, b\} \text{ and } \mathcal{B} = \{a^* b^\omega\}, \tag{2.5}$$

where $(\cdot)^*$ and $(\cdot)^w$ denote, respectively, the finite and the infinite repetition of the respective string. Using this behavior in (2.4) we obtain

$$\mathcal{B}|_{[0,k]} = \left\{ a^r b^{k+1-r} \middle| r \in \mathbb{N}_0, r \leq k+1 \right\}, \text{ hence } \overline{\mathcal{B}} = \{a^\omega, a^* b^\omega\} \supset \mathcal{B}.$$

This implies that $\mathcal{B}$ is not complete. ◁

l-Completeness

The behavior of a complete system can be fully described by local properties. In the special case where these properties can be evaluated on bounded time intervals of length $l+1$ with $l \in \mathbb{N}_0$, following [76, p.184], the system is called l-complete.

Definition 2.5. *A dynamical system* $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ *is* l-complete *for* $l \in \mathbb{N}_0$ *if*

$$\left(\forall k \in \mathbb{N}_0 \,.\, \omega|_{[k,k+l]} \in \mathcal{B}|_{[k,k+l]}\right) \Leftrightarrow \omega \in \mathcal{B}. \tag{2.6}$$

Intuitively, l-completeness is explained easiest by the following gedankenexperiment. Assume playing a sophisticated domino game where

$$\bigcup_{k \in \mathbb{N}_0} \mathcal{B}|_{[k,k+l]}$$

is the set of dominos, namely all finite strings representing the restriction of admissible signals to a time interval of length $l+1$. The domino game is played by picking the first domino from the set $\mathcal{B}|_{[0,l]}$ and appending one domino from the set $\mathcal{B}|_{[1,1+l]}$ if the last l symbols of the first domino are equivalent to the first l symbols of the second domino. Playing the domino game arbitrarily long and with all possible initial conditions and domino combinations results in the set $\mathcal{B}^l$ containing all signals that satisfy the left side of (2.6). If the system is l-complete we have $\mathcal{B} = \mathcal{B}^l$ emphasizing that all valid signals can be fully described by a local property.

Example 2.2. Consider the system

$$\begin{aligned} &\Sigma = (\mathbb{N}_0, \{a, b\}, \mathcal{B}) \quad s.t. \\ &\mathcal{B} = \{aaab(aab)^\omega, aab(aab)^\omega, ab(aab)^\omega, b(aab)^\omega\}, \end{aligned} \tag{2.7}$$

Observe that Σ is time invariant, but not strictly time invariant, since

$$aaab(aab)^w \notin \sigma\mathcal{B} = \{aab(aab)^\omega, ab(aab)^\omega, b(aab)^\omega, (aab)^\omega\}.$$

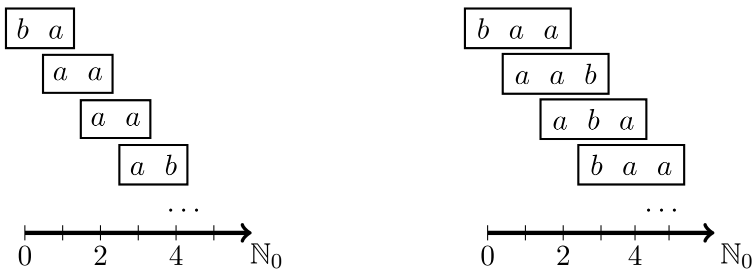

Figure 2.1.: Domino game for $l = 1$ (left) and $l = 2$ (right) in Example 2.2.

Using $l = 1$ we get the domino set

$$\forall k \in \mathbb{N}_0 \,.\, \mathcal{B}|_{[k,k+1]} = \mathcal{B}|_{[0,1]} = \{aa, ab, ba\}. \tag{2.8}$$

As depicted in Figure 2.1 (left), we can start the domino game with the piece ba and append a piece that starts with an a, e.g., aa. Observe that the signal constructed in Figure 2.1 (left), i.e., $\omega = baaab...$, is not allowed in (2.7) since not more than two sequential a's can occur for $k > 0$. However, we can of course construct all signals $\omega \in \mathcal{B}$ using the outlined domino game. This implies that (i) the system Σ in (2.7) is not 1-complete and (ii) the domino game constructs a behavior $\mathcal{B}^1$ that is larger than the one in (2.7), i.e., $\mathcal{B}^1 \supset \mathcal{B}$. Now, increasing l to $l = 2$ gives the following set of domino pieces

$$\begin{aligned} &\mathcal{B}|_{[0,2]} = \{aaa, aab, aba, baa\}, \\ &\forall k > 0 \,.\, \mathcal{B}|_{[k,k+2]} = \mathcal{B}|_{[1,3]} = \{aab, aba, baa\}. \end{aligned} \tag{2.9}$$

Playing the domino game with these sets results, for example, in the signal depicted in Figure 2.1 (right), where always two symbols are required to match. Observe that after the first piece we are only allowed to pick from the set $\mathcal{B}|_{[1,3]}$. This prevents the occurrence of more than two sequential a's since the domino aaa cannot be attached. We get $\mathcal{B}^2 = \mathcal{B}$, i.e., the system Σ in (2.7) is 2-complete. $\triangleleft$

As a special case it can be shown that the behavior of an l-complete system Σ can by fully described by the initial signal pieces $\mathcal{B}|_{[0,l]}$ if Σ is strictly time invariant.

Lemma 2.6. *Let $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ be a strictly time invariant dynamical system and $l \in \mathbb{N}_0$. Then Σ is l-complete if*

$$\left(\forall k \in \mathbb{N}_0 \,.\, \omega|_{[k,k+l]} \in \mathcal{B}|_{[0,l]}\right) \Leftrightarrow \omega \in \mathcal{B}. \tag{2.10}$$

Proof. If Σ is strictly time invariant, i.e., $\sigma\mathcal{B} = \mathcal{B}$, then $\forall k \in \mathbb{N}_0 \,.\, \sigma^k\mathcal{B} = \mathcal{B}$, hence $\forall k \in \mathbb{N}_0 \,.\, \mathcal{B}|_{[k,k+l]} = \mathcal{B}|_{[0,l]}$. Therefore, (2.10) and (2.6) are equivalent. $\square$

Remark 2.1. For systems that are time invariant but not strictly time invariant there exist $k \in \mathbb{N}_0$ s.t. $\mathcal{B}|_{[k,k+l]} \subset \mathcal{B}|_{[0,l]}$, implying that in this case (2.6) and (2.10) are not equivalent. Therefore, as already pointed out in [35, p.44], the definition of l-completeness via (2.10), as used in [36, Def.8] and subsequent papers, does not coincide with the definition of l-completeness via (2.6), as originally used by J.C.Willems [76, Sec.1.4.1]. In particular, an l-complete system in the sense of [36] (i.e., a system satisfying (2.10)) is also l-complete in the sense of [76] (i.e., it also satisfies (2.6)) but the inverse implication does not hold in general. To see this, suppose that $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ is l-complete in the sense of [36]. Then, obviously,

$$\left(\forall k \in \mathbb{N}_0 \;.\; \omega|_{[k,k+l]} \in \mathcal{B}|_{[0,l]}\right) \Rightarrow \omega \in \mathcal{B}$$

and $\forall k \in \mathbb{N}_0 \;.\; \mathcal{B}|_{[k,k+l]} \subseteq \mathcal{B}|_{[0,l]}$. It follows that

$$\left(\forall k \in \mathbb{N}_0 \;.\; \omega|_{[k,k+l]} \in \mathcal{B}|_{[k,k+l]}\right) \Rightarrow \omega \in \mathcal{B},$$

which implies l-completeness in the sense of [76]. Therefore, l-completeness in the sense of [36] is a stronger property than l-completeness in the sense of [76]. $\triangleleft$

Memory Span

Recalling the gedankenexperiment of Section 2.2, at time k all necessary information to determine the future evolution (i.e., the next feasible domino) of an l-complete system is captured in the last l symbols of the observed signal ω. Following [76, p.184], systems which exhibit this property are said to have memory span l.

Definition 2.7. *A dynamical system $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ has* memory span $l \in \mathbb{N}_0$ *if*

$$\forall \omega, \omega' \in \mathcal{B}, k \in \mathbb{N}_0 \;.\; \left(\omega|_{[k,k+l-1]} = \omega'|_{[k,k+l-1]} \Rightarrow \omega \wedge_k \omega' \in \mathcal{B}\right). \tag{2.11}$$

If Σ has memory span $l = 0$ it is called memoryless*.*

It obviously holds that any l-complete system has memory span l. However, the reverse implication is only true if the system is complete to ensure that its behavior can be fully described by a local property such as a finite memory span. This statement was proven in [76, Prop.1.1] for $T = \mathbb{Z}$ and trivially specializes to $T = \mathbb{N}_0$.

Proposition 2.8. *Let $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ be a dynamical system and $l \in \mathbb{N}_0$. Then*

$$\Sigma \textit{ is } l\textit{-complete} \Leftrightarrow \begin{pmatrix} \Sigma \textit{ is complete} \\ \wedge \Sigma \textit{ has memory span } l \end{pmatrix} \tag{2.12}$$

Proof. "$\Rightarrow$": Obviously l-completeness implies completeness (from (2.3b) and (2.6)). To show that l-completeness of Σ implies (2.11) we fix $\omega_1, \omega_2 \in \mathcal{B}, k \in \mathbb{N}_0$ s.t. $\omega_1|_{[k,k+l-1]} = \omega_2|_{[k,k+l-1]}$ and show $\omega = \omega_1 \wedge_k \omega_2 \in \mathcal{B}$. As $\omega_1|_{[k,k+l-1]} = \omega_2|_{[k,k+l-1]}$ it holds that

$$\omega|_{[k',k'+l]} = \begin{cases} \omega_1|_{[k',k'+l]} & , \ k' < k \\ \omega_2|_{[k',k'+l]} & , \ k' \geq k \end{cases}, \quad \text{hence} \quad \forall k' \in \mathbb{N}_0 \ . \ \omega|_{[k',k'+l]} \in \mathcal{B}|_{[k',k'+l]}.$$

Now $\omega \in \mathcal{B}$ follows from l-completeness of Σ with (2.6).
"$\Leftarrow$": To show that (2.6) holds for a complete system Σ with memory span l we fix $\omega \in W^{\mathbb{N}_0}$ s.t. the left hand side of (2.6) holds and show that $\omega \in \mathcal{B}$ follows if Σ is complete and has memory span l (as the reverse direction in (2.6) always holds). Observe that the left hand side of (2.6) implies

$$\exists \omega_0, \omega_1, \omega_2, \ldots \in \mathcal{B} \ . \begin{pmatrix} \omega_0|_{[0,l]} = \omega|_{[0,l]} \\ \wedge \omega_1|_{[1,l+1]} = \omega|_{[1,l+1]} \\ \wedge \omega_2|_{[2,l+2]} = \omega|_{[2,l+2]} \\ \wedge \quad \ldots \end{pmatrix}, \tag{2.13}$$

hence $\omega_0|_{[1,l]} = \omega_1|_{[1,l]}$ and $\omega_1|_{[2,l+1]} = \omega_2|_{[2,l+1]}$. As Σ has memory span l this implies $\omega_0 \wedge_1 \omega_1 \in \mathcal{B}$ and $\omega_1 \wedge_2 \omega_2 \in \mathcal{B}$ and therefore

$$(\omega_0 \wedge_1 \omega_1 \wedge_2 \omega_2)\,|_{[0,l+2]} = \omega|_{[0,l+2]} \in \mathcal{B}|_{[0,l+2]}.$$

Iteratively applying this procedure therefore yields $\forall \tau \in \mathbb{N}_0$. $\omega|_{[0,\tau]} \in \mathcal{B}|_{[0,\tau]}$ implying $\omega \in \mathcal{B}$ from (2.3b) as Σ is complete. □

2.3. Strongest l-Complete Approximations (SlCA)

The set $\mathcal{B}^l$ generated in the domino game outlined in Section 2.2 also matches the behavior of the system Σ if Σ is r-complete with $r \leq l$ since using larger dominos cannot lead to a richer behavior. Furthermore, as already shown in Example 2.2, we will always get $\mathcal{B}^l \supseteq \mathcal{B}$ even if the system is not complete at all, since using less information in the domino game generates more freedom in constructing signals. Following [36, Def.9], this leads to the definition of l-complete approximations.

Definition 2.9. *Let $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ be a dynamical system. Then $\Sigma' = (\mathbb{N}_0, W, \mathcal{B}')$ is an* l-complete approximation *of $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ if (i) Σ' is l-complete (defined via (2.6)) and (ii) $\mathcal{B}' \supseteq \mathcal{B}$. Furthermore, Σ' is the* strongest l-complete approximation *of Σ if (i) Σ' is an l-complete approximation of Σ and (ii) for any l-complete approximation $\Sigma'' = (\mathbb{N}_0, W, \mathcal{B}'')$ of Σ it holds that $\mathcal{B}' \subseteq \mathcal{B}''$.*

Remark 2.2. As an immediate consequence of Remark 2.1, l-complete approximations introduced in [36] (where l-completeness is defined via (2.10) instead of (2.6)) are always l-complete approximations as defined above. The reverse implication does generally not hold. $\triangleleft$

Generalizing the results in [36, Prop.10] to the l-completeness definition in (2.6) shows that the behavior $\mathcal{B}^l$ constructed in the domino game of the previous section is the behavior of the (unique) strongest l-complete approximation.

Theorem 2.10. *Let $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ be a dynamical system. Then the unique strongest l-complete approximation (SlCA) of Σ is given by $\widehat{\Sigma}^{l\Uparrow} = (\mathbb{N}_0, W, \widehat{\mathcal{B}}^{l\Uparrow})$, with*

$$\widehat{\mathcal{B}}^{l\Uparrow} := \left\{\omega \in W^{\mathbb{N}_0} \middle| \forall k \in \mathbb{N}_0 \;.\; \omega|_{[k,k+l]} \in \mathcal{B}|_{[k,k+l]}\right\}. \tag{2.14}$$

Furthermore, if Σ is strictly time invariant then

$$\widehat{\mathcal{B}}^{l\Uparrow} = \left\{\omega \in W^{\mathbb{N}_0} \middle| \forall k \in \mathbb{N}_0 \;.\; \omega|_{[k,k+l]} \in \mathcal{B}|_{[0,l]}\right\}. \tag{2.15}$$

Proof. To show that $\widehat{\Sigma}^{l\Uparrow}$ is a strongest l-complete approximation we prove all three conditions in Definition 2.9 separately.

(i) $\widehat{\Sigma}^{l\Uparrow}$ is l-complete as (2.14) implies $\widehat{\mathcal{B}}^{l\Uparrow}|_{[k,k+l]} = \mathcal{B}|_{[k,k+l]}$, hence $\omega \in \widehat{\mathcal{B}}^{l\Uparrow} \Leftrightarrow \left(\forall k \in \mathbb{N}_0 \;.\; \omega|_{[k,k+l]} \in \widehat{\mathcal{B}}^{l\Uparrow}|_{[k,k+l]}\right)$.

(ii) $\mathcal{B} \subseteq \widehat{\mathcal{B}}^{l\Uparrow}$ holds as $\omega \in \mathcal{B}$ implies $\forall k \in \mathbb{N}_0 \;.\; \omega|_{[k,k+l]} \in \mathcal{B}|_{[k,k+l]}$, hence $\omega \in \widehat{\mathcal{B}}^{l\Uparrow}$ from (2.14).

(iii) For any l-complete approximation $\Sigma' = (\mathbb{N}_0, W, \mathcal{B}')$ of Σ the inclusion $\mathcal{B} \subseteq \mathcal{B}'$ and therefore $\mathcal{B}|_{[k,k+l]} \subseteq \mathcal{B}'|_{[k,k+l]}$ holds. Hence, using (2.14), $\omega \in \widehat{\mathcal{B}}^{l\Uparrow}$ implies $\forall k \in \mathbb{N}_0 \;.\; \omega|_{[k,k+l]} \in \mathcal{B}'|_{[k,k+l]}$ and therefore $\omega \in \mathcal{B}'$ since Σ' is l-complete.

Observe that $\widehat{\Sigma}^{l\Uparrow}$ is unique as (iii) implies that $\widehat{\mathcal{B}}^{l\Uparrow}$ is the unique smallest element of the set $\{\mathcal{B}'\}$ containing the behaviors of all l-complete approximations $\Sigma' = (\mathbb{N}_0, W, \mathcal{B}')$ of Σ. The second part of the theorem follows directly from (2.10) in Lemma 2.6. $\square$

Example 2.3. As a consequence of Theorem 2.10, the behaviors $\mathcal{B}^1$ and $\mathcal{B}^2$ constructed in Example 2.2 characterize the strongest 1-complete and the strongest 2-complete approximation of the system in (2.7), respectively. $\triangleleft$

Remark 2.3. The SlCA for a time invariant system Σ using the l-completeness property in the sense of [36] was shown to be characterized by the behavior

$$\widetilde{\mathcal{B}} = \left\{\omega \in W^{\mathbb{N}_0} \middle| \forall k \in \mathbb{N}_0 \;.\; \omega|_{[k,k+l]} \in \mathcal{B}|_{[0,l]}\right\}.$$

Then, since time invariance implies $\forall k \in \mathbb{N}_0 \;.\; \mathcal{B}|_{[k,k+l]} \subseteq \mathcal{B}|_{[0,l]}$, we have $\widehat{\mathcal{B}}^{l\Uparrow} \subseteq \widetilde{\mathcal{B}}$. Therefore, the S$l$CA characterized by Theorem 2.10 is a tighter approximation than the SlCA defined in [36]. $\triangleleft$

2.4. State Space Dynamical Systems (SSDS)

As behaviors are sets of infinite sequences which can technically not be recorded one usually derives the behavior of a system from physical principals which results in equations including more than the externally visible variables. These additional variables are called *latent variables* [76]. Latent variables for which the axiom of state holds are called *states*. The latter is true if at any time all relevant information necessary to decide on the possible future evolution of a dynamical system is captured by the current value of the latent variables. Additionally using state variables to model the dynamics of a system results in a so called state space dynamical system.

Definition 2.11 ([76], p.185)**.** *The dynamical system* $\Sigma_S = (T, W \times X, \mathcal{B}_S)$ *is a* state space dynamical system *(SSDS) if*

$$\forall (\omega, \xi), (\omega', \xi') \in \mathcal{B}_S, t \in T \ . \ \big(\xi(t) = \xi'(t) \Rightarrow (\omega, \xi) \wedge_t (\omega', \xi') \in \mathcal{B}_S\big) \, . \tag{2.16}$$

Σ_S *is a* state space representation *of* $\Sigma = (T, W, \mathcal{B})$ *if* $\pi_W(\mathcal{B}_S) = \mathcal{B}$.

By investigating Definition 2.5, 2.7 and 2.11 we can conclude that (i) every l-complete system has memory span l, (ii) the state property implies that $\Sigma_x = (\mathbb{N}_0, X, \pi_X(\mathcal{B}_S))$ has memory span one and (iii) a straightforward choice for the state space of systems with memory span l is given by the set of admissible strings[2] up to length l. This takes into account that for the first l time steps we can only memorize the symbols already seen. Using this choice, we can generalize the construction of a state space representation given in [36, p.6] to systems with memory span l in the sense of Definition 2.7.

Proposition 2.12. *Let* $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ *be a system with memory span* l*. Furthermore, let*

$$X := \Big(\bigcup_{r \in [0,l-1]} \mathcal{B}|_{[0,r-1]}\Big) \cup \big(\bigcup_{k \in \mathbb{N}_0} \mathcal{B}|_{[k,k+l-1]}\big) \tag{2.17}$$

and let $\mathcal{B}_S \subseteq (W \times X)^{\mathbb{N}_0}$ *s.t.* $(\omega, \xi) \in \mathcal{B}_S$ *iff*

$$\xi(k) = \begin{cases} \omega|_{[0,k-1]} & 0 \leq k < l \\ \omega|_{[k-l,k-1]} & k \geq l \end{cases} \quad \text{and} \quad \omega \in \mathcal{B}. \tag{2.18}$$

Then $\Sigma_S = (\mathbb{N}_0, W \times X, \mathcal{B}_S)$ *is a state space representation of* Σ.

Proof. $\pi_W(\mathcal{B}_S) = \mathcal{B}$ holds by construction. To show (2.16), pick $(\omega_1, \xi_1), (\omega_2, \xi_2) \in \mathcal{B}_S$ and $k' \in \mathbb{N}_0$ s.t. $\xi_1(k') = \xi_2(k')$ and show $(\omega, \xi) = (\omega_1, \xi_1) \wedge_{k'} (\omega_2, \xi_2) \in \mathcal{B}_S$. First observe that

$$\xi_1(k') = \omega_1|_{[\max\{0,k'-l\},k'-1]} = \omega_2|_{[\max\{0,k'-l\},k'-1]} = \xi_2(k'). \tag{2.19}$$

[2] In contrast to [36] this choice of the state space represents only the reachable part of $\bigcup_{r \leq l} (W)^r$.

This implies for $k' < l$ that $\omega = \omega_1 \wedge_{k'} \omega_2 = \omega_2 \in \mathcal{B}$. From Σ having memory span l (2.19) implies[3] $\omega = \omega_1 \wedge_{k'} \omega_2 \in \mathcal{B}$ for $k' \geq l$. Now remember that (2.18) holds for $(\omega_1, \xi_1), (\omega_2, \xi_2) \in \mathcal{B}_S$. Therefore, $\xi = \xi_1 \wedge_{k'} \xi_2$ implies that for all $k \in \mathbb{N}_0$

$$\xi(k) = \begin{cases} \omega_1|_{[0,k-1]} & (k \leq k') \wedge (k < l) \\ \omega_1|_{[k-l,k-1]} & (k \leq k') \wedge (k \geq l) \\ \omega_1|_{[0,k'-1]} \cdot \omega_2|_{[k',k-1]} & (k' < k < k'+l) \wedge (k < l) \\ \omega_1|_{[k-l,k'-1]} \cdot \omega_2|_{[k',k-1]} & (k' < k < k'+l) \wedge (k \geq l) \\ \omega_2|_{[k-l,k-1]} & (k \geq k'+l)\,. \end{cases}$$

Hence, with $\omega = \omega_1 \wedge_{k'} \omega_2 \in \mathcal{B}$, $\xi = \xi_1 \wedge_{k'} \xi_2$ satisfies (2.18), proving $(\omega, \xi) \in \mathcal{B}_S$. □

Since the SlCA $\widehat{\Sigma}^{l\Uparrow} = (\mathbb{N}_0, W, \widehat{\mathcal{B}}^{l\Uparrow})$ of any dynamical system $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ is l-complete it follows from Proposition 2.8 that $\widehat{\Sigma}^{l\Uparrow}$ has memory span l and we can use Proposition 2.12 to construct a state space representation of $\widehat{\Sigma}^{l\Uparrow}$, which is denoted by $\widehat{\Sigma}_S^{l\Uparrow} = (\mathbb{N}_0, W \times \widehat{X}^{l\Uparrow}, \widehat{\mathcal{B}}_S^{l\Uparrow})$. Note that the state space constructed in Proposition 2.12 has finitely many elements if W is finite.

Example 2.4. Recall that the dynamical system Σ in Example 2.2 is 2-complete and we have

$$\bigcup_{r\in[0,1]} \mathcal{B}|_{[0,r-1]} = \{\lambda, a, b\} \quad \text{and} \quad \bigcup_{k\in\mathbb{N}_0} \mathcal{B}|_{[k,k+1]} = \{aa, ab, ba\}.$$

Using Proposition 2.12, this implies that Σ in (2.7) can be realized by the state space dynamical system $\Sigma_S = (\mathbb{N}_0, W \times X, \mathcal{B}_S)$ with $X = \{\lambda, a, b, aa, ab, ba\}$. Analogously, the state space representation of the strongest 1-complete approximation of Σ has state space $X^{1\Uparrow} = \{\lambda, a, b\}$. ◁

2.5. Evolution Laws and State Machines

Evolution laws where introduced in [76] to model the step-by-step evolution of a behavioral system. In comparison to their construction in [76, Sec.1.5], where only time invariant systems with time axis $\mathbb{Z}$ are considered, a slightly more general definition of *time dependent evolution laws* is used in this section.

Definition 2.13. *A* time dependent discrete time evolution law *is a tuple* $\Sigma_\psi = (\mathbb{N}_0, W, X, \psi, X_0)$*, where* X *is the set of states,* W *is the set of external symbols,* $X_0 \subseteq X$ *is the*

[3] Observe that, under the premises of (2.11), $\omega_1 \wedge_k \omega_2 = \omega_1 \wedge_{k+l} \omega_2$ in the right side of the implication in (2.11).

set of initial states and $\psi : \mathbb{N}_0 \to 2^{X\times W\times X}$ *is the time dependent next state relation. Furthermore, the* behavior induced by ψ *is defined by*

$$\mathcal{B}_\psi := \left\{ (\omega,\xi) \middle| \begin{pmatrix} \xi(0) \in X_0 \\ \wedge \forall k \in \mathbb{N}_0 \,.\, (\xi(k), \omega(k), \xi(k+1)) \in \psi(k) \end{pmatrix} \right\}. \tag{2.20a}$$

Let $\delta := \bigcup_{k\in\mathbb{N}_0} \psi(k) \subseteq X \times W \times X$ *be a time independent next state relation and*

$$\mathcal{B}_\delta := \left\{ (\omega,\xi) \middle| \begin{pmatrix} \xi(0) \in X_0 \\ \wedge \forall k \in \mathbb{N}_0 \,.\, (\xi(k), \omega(k), \xi(k+1)) \in \delta \end{pmatrix} \right\} \tag{2.20b}$$

the behavior induced by it. Then Σ_ψ *is called* time independent *if* $\mathcal{B}_\psi = \mathcal{B}_\delta$*.*

When Σ_ψ is time independent the definition of Σ_ψ simplifies to the time independent discrete time evolution law $\Sigma_\delta = (\mathbb{N}_0, W, X, \delta, X_0)$ with induced behavior $\mathcal{B}_\delta$. Note that $\mathcal{B}_\delta$ is not necessarily time invariant in the (behavioral) sense of Section 2.2. Observe that for unrestricted initial states, i.e., $X_0 = X$, Σ_δ coincides with [76, Def.1.4] and the behavior $\mathcal{B}_\delta$ becomes time invariant and coincides with the one in [76, p.189].

Based on the behavior of an SSDS Σ_S an evolution law reproducing its step-by-step evolution can be constructed as follows.

Definition 2.14. *Let* $\Sigma_S = (\mathbb{N}_0, W \times X, \mathcal{B}_S)$ *be an SSDS and* $\Sigma_\psi = (\mathbb{N}_0, W, X, \psi, X_0)$ *an evolution law s.t.* $X_0 = \pi_X(\mathcal{B}_S)|_{[0,0]}$ *and*

$$\psi(k) := \left\{ (x, w, x') \middle| \exists (\omega,\xi) \in \mathcal{B}_S \,.\, \begin{pmatrix} \xi(k) = x \\ \wedge \omega(k) = w \\ \wedge \xi(k+1) = x' \end{pmatrix} \right\}. \tag{2.21}$$

Then Σ_ψ *is the* evolution law induced by Σ_S*.*

State Machines

Time independent discrete time evolution laws Σ_δ are, up to some notational differences, identical to *state machines* $\mathcal{Q}$ as, e.g., used in [36].

Definition 2.15 ([36], Def. 3)**.** *A* state machine *(SM) is a tuple* $\mathcal{Q} = (X, W, \delta, X_0)$ *where* X *is the set of states,* W *is the set of external symbols,* $X_0 \subseteq X$ *is the set of initial states, and* $\delta \subseteq X \times W \times X$ *is a next state relation. The* full behavior *of* $\mathcal{Q}$ *is defined by*

$$\mathcal{B}_f(\mathcal{Q}) := \mathcal{B}_\delta. \tag{2.22}$$

$\mathcal{Q}$ *is called* live and reachable *if*

$$\forall x \in X \,.\, \exists (\omega,\xi) \in \mathcal{B}_f(\mathcal{Q}), k \in \mathbb{N}_0 \,.\, \xi(k) = x \tag{2.23}$$

Furthermore, $\mathcal{Q}$ *is called a* finite state machine *(FSM) if* $|X| < \infty$ *and* $|W| < \infty$*.*

As the realization of an abstraction by an FSM is a necessary prerequisite for the application of symbolic controller synthesis methods, we are interested in realizing SlCA by an FSM whenever W is finite.

Evolution Laws for l-Complete Dynamical Systems

Thinking back to the presented domino game in Section 2.2 and the construction of a state space representation $\Sigma_S = (\mathbb{N}_0, W \times X, \mathcal{B}_S)$ of an l-complete dynamical system Σ in Proposition 2.12, a transition constructed in (2.21) from $\mathcal{B}_S$ represents adding an allowed domino in the domino game. However, the set of allowed dominos is time dependent since we have to pick from the subset $\mathcal{B}|_{[k,k+l]}$ of all dominos D_{l+1} at time k. This suggests that the evolution law induced by Σ_S is *time dependent*. Therefore, extending the usual construction from [36, Thm.12] to the l-completeness definition used in this thesis results in a time dependent evolution law rather than an SM.

Proposition 2.16. *Let $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ be an l-complete dynamical system and $\Sigma_S = (\mathbb{N}_0, W \times X, \mathcal{B}_S)$ its state space representation constructed in Proposition 2.12. Then Σ_S induces the evolution law $\Sigma_\psi = (\mathbb{N}_0, W, X, \psi, X_0)$ with $X_0 = \{\lambda\}$ and*

$$\begin{aligned}\psi(k) = &\left\{(x, w, x \cdot w) \middle| k < l \wedge x \cdot w \in \mathcal{B}|_{[0,k]}\right\} \\ &\cup \left\{(x, w, (x \cdot w)|_{[1,l-1]}) \middle| k \geq l \wedge x \cdot w \in \mathcal{B}|_{[k-l,k]}\right\}.\end{aligned} \tag{2.24}$$

Proof. The next state relation (2.24) satisfies (2.21). Moreover, $\pi_X(\mathcal{B}_S)|_{[0,0]} = \{\lambda\}$ as (2.18) implies $\xi(0) = \omega|_{[0,-1]} = \lambda$. Hence Σ_ψ is the evolution law induced by Σ_S. $\square$

Remark 2.4. Recall the gedankenexperiment in Section 2.2 and observe that in the construction of Proposition 2.12 the state represents the “recent past” of the signal w, i.e., a finite string of length l if $k \geq l$ (see Figure 2.2 (middle)). However, at start up, i.e., for $k < l$, no “past” of this length exists. In this case the state describes the available past information, i.e., a finite string of length $r \in [0, l-1]$ contained in the set $\mathcal{B}|_{[0,r-1]}$ (see Figure 2.2 (left)). Therefore, assuming $|x|_L = r < l$ implies that $(x, w, x') \in \psi(k)$ if x' is the extension of x by w and a valid initial behavior, i.e., $x' \in \mathcal{B}|_{[0,r]}$.

Recall that the domino game describes the admissible behavior by appending domino pieces of length $l+1$ such that the last l symbols match. Therefore, assuming $|x|_L = l$ implies $(x, w, x') \in \psi(k)$ if $x \cdot w$ is a domino that is currently allowed to be attached, hence $x \cdot w \in \mathcal{B}|_{[k-l,k]}$ and $x' = x|_{[1,l-1]} \cdot w$ (see Figure 2.2 (middle)). $\triangleleft$

As every l-complete approximation is l-complete, we can use Proposition 2.16 to construct an evolution law induced by the SlCA of our running example.

Example 2.5. Using the state space derived in Example 2.4 and the construction of the next state relation in (2.24), we can construct the evolution law Σ_ψ realizing the state space representation Σ_S of the dynamical system Σ in Example 2.2. The evolution law

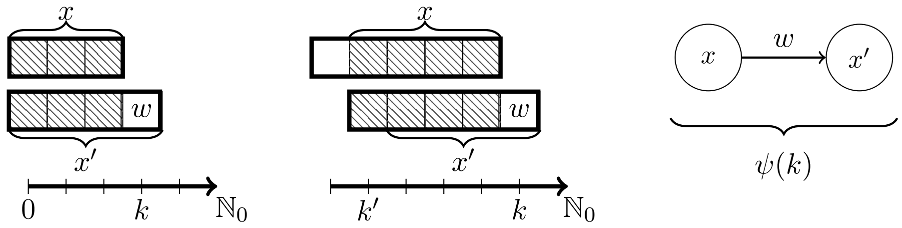

Figure 2.2.: Correspondence of one step in the domino game for $k = 3 < l = 4$ (left) and $k > l = 4$ with $k' = k - l$ (middle) to one transition in Σ_ψ (right).

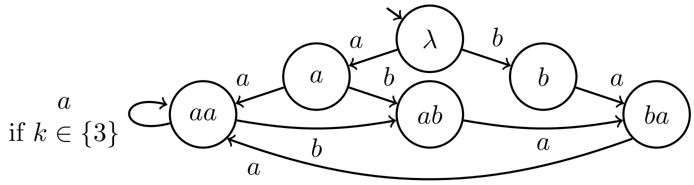

Figure 2.3.: Evolution law Σ_ψ constructed in Example 2.5, realizing the state space representation Σ_S of the dynamical system Σ in Example 2.2.

Σ_ψ is depicted in Figure 2.3 where a transition $(x, w, x') \in \psi(k)$ is depicted by an arrow from x to x' labeled by "w if $k \in V$" where $V := \{k \in \mathbb{N}_0 | (x, w, x') \in \psi(k)\}$. Whenever a transition can always occur if its source state x is reached the label reduces to "w". The initial state is indicated by an arrow pointing to it from "outside". Note, that in Σ_{ψ_2} the transition from state aa to itself is time dependent as three sequential a's are only allowed at start up. $\triangleleft$

2.6. Asynchronous State Space Dynamical Systems (ASSDS)

It was shown in the previous section that a straightforward extension of [36, Thm.12] to the l-completeness definition used in this thesis results in time dependent evolution laws rather than state machine realizations of SlCA. Obviously, one could render the evolution law Σ_ψ in Definition 2.14 time independent by using time as an additional state variable. However, this would always lead to an infinite state set.

To ensure that constructed abstractions are always realizable by state machines we want to derive a new property of dynamical systems in this section allowing for realizability by a *time independent* evolution law, in particular by a (finite) SM. Observe that this property must allow for concatenation of state trajectories that reach the same state asynchronously (i.e., at different times). This is formalized in the following definition inspired by [26, p.59].

Definition 2.17. *The dynamical system $\Sigma_S = (\mathbb{N}_0, W \times X, \mathcal{B}_S)$ is an* asynchronous state space dynamical system *(ASSDS) if*

$$\forall(\omega,\xi),(\omega',\xi')\in\mathcal{B}_S, k, k'\in\mathbb{N}_0 \,.\, \Big(\xi(k){=}\xi'(k') \Rightarrow (\omega,\xi) \wedge_{k'}^{k} (\omega',\xi')\in\mathcal{B}_S\Big)\,. \tag{2.25}$$

Σ_S *is an* asynchronous state space representation *of* $\Sigma = (T, W, \mathcal{B})$ *if* $\pi_W(\mathcal{B}_S) = \mathcal{B}$.

It can be easily observed that every asynchronous state space dynamical system is also a synchronous[4] state space dynamical system since we can always pick $k_1 = k_2 = t$ in (2.25) and get (2.16).

Using Definition 2.17 we can verify our intuition and show that the asynchronous concatenation in the definition of ASSDS implies that its induced evolution law is time independent and can therefore be equivalently given by a state machine.

Proposition 2.18. *Let $\Sigma_S = (\mathbb{N}_0, W, X, \mathcal{B}_S)$ be an ASSDS. Then the evolution law induced by Σ_S is time independent and can therefore be equivalently represented by an SM $\mathcal{Q} = (X, W, \delta, X_0)$ s.t. $X_0 = \pi_X(\mathcal{B}_S)|_{[0,0]}$ and*

$$\delta := \left\{ (x,w,x') \,\middle|\, \exists(\omega,\xi)\in\mathcal{B}_S, k\in\mathbb{N}_0 \,.\, \begin{pmatrix} \xi(k) = x \\ \wedge\omega(k) = w \\ \wedge\xi(k+1) = x' \end{pmatrix} \right\}. \tag{2.26}$$

Furthermore, $\mathcal{Q}$ is live and reachable. $\mathcal{Q}$ is called the state machine induced by Σ_S.

Proof. First, observe that (2.23) holds for $\mathcal{Q}$ by construction. To prove the first part of the statement, let $\Sigma_\psi = (\mathbb{N}_0, W, X, \psi, X_0)$ be the evolution law induced by Σ_S and $\delta = \bigcup_{k\in\mathbb{N}_0} \psi(k)$. Now observe that (2.20a) and (2.20b) coincide (i.e., $\mathcal{B}_\psi = \mathcal{B}_\delta$) if

$$\forall k\in\mathbb{N}_0 \,.\, \begin{pmatrix} (x,w,x')\in\psi(k) \\ \wedge(x,w'',x'')\in\delta \end{pmatrix} \Rightarrow (x,w'',x'')\in\psi(k), \tag{2.27}$$

which remains to be shown. Using (2.21) and $\delta = \bigcup_{k\in\mathbb{N}_0} \psi(k)$, the left hand side of (2.27) implies for all $k\in\mathbb{N}_0$

$$\exists(\omega',\xi')\in\mathcal{B}_S. \begin{pmatrix} \xi'(k){=}x \\ \wedge\omega'(k){=}w \\ \wedge\xi'(k{+}1){=}x' \end{pmatrix} \text{ and } \exists(\omega'',\xi'')\in\mathcal{B}_S, k''\in\mathbb{N}_0. \begin{pmatrix} \xi''(k''){=}x \\ \wedge\omega''(k''){=}w'' \\ \wedge\xi''(k''{+}1){=}x'' \end{pmatrix}.$$

[4] To clearly distinguish the asynchronous state property and the (standard) state property from Section 2.4 we will refer to the latter one as **synchronous** state property in the remainder of this thesis. The same convention is applied to other properties as memory span and l-completeness.

Now (2.25) implies $(\omega, \xi) = (\omega', \xi') \wedge_{k''}^{k} (\omega'', \xi'') \in \mathcal{B}_S$ and therefore $\xi(k) = x$, $\omega(k) = w''$ and $\xi(k+1) = x''$ implying $(x, w'', x'') \in \psi(k)$ (from (2.21)). $\square$

Remark 2.5. It is interesting to note that the asynchronous state property from Definition 2.17 and the resulting time independence of the induced evolution law coincide precisely with the "standard" notion of time invariant state space systems in continuous or discrete time (see, e.g., [28, Ch.2]). It should be kept in mind though that this notion does not imply and is not implied by the (behavioral) notion of time invariance introduced in Section 2.1. $\triangleleft$

ASSDS vs. SM

As we will use the connection between ASSDS and SM quite extensively in Part I we investigate Proposition 2.18 more closely.

First, observe that we can also invert Proposition 2.18 to obtain an ASSDS from a given SM. However, it follows from (2.22) and (2.20b) that the full behavior $\mathcal{B}_f(\mathcal{Q})$ of a state machine $\mathcal{Q}$ is fully determined by a local property, i.e., the next state relation δ. Hence, the resulting ASSDS is always complete.

Proposition 2.19. *Let $\mathcal{Q} = (X, W, \delta, X_0)$ be an SM. Then $\Sigma_S = (\mathbb{N}_0, W \times X, \mathcal{B}_f(\mathcal{Q}))$ is an ASSDS and Σ_S is complete. Σ_S is called the ASSDS induced by $\mathcal{Q}$.*

Proof. To see that Σ_S is an ASSDS pick $(\omega, \xi), (\omega', \xi') \in \mathcal{B}_f(\mathcal{Q})$ s.t. $\xi(k) = \xi'(k')$. Then it follows immediately from (2.20b) that $(\tilde{\omega}, \tilde{\xi}) = (\omega, \xi) \wedge_k (\omega', \xi') \in \mathcal{B}_f(\mathcal{Q})$ as for all k'' holds that $(\tilde{\xi}(k''), \tilde{\omega}(k''), \tilde{\xi}(k''+1)) \in \delta$.

For completeness of Σ_S, observe that "$\Leftarrow$" always holds in (2.3b). To show "$\Rightarrow$", pick (ω, ξ) s.t. $\forall k \in \mathbb{N}_0 \,.\, (\omega|_{[0,k]}, \xi|_{[0,k]}) \in \mathcal{B}_f(\mathcal{Q})|_{[0,k]}$. This implies that $\xi(0) \in X_0$ and $\forall k \in \mathbb{N}_0 \,.\, (\xi(k), \omega(k), \xi(k+1)) \in \delta$ and therefore $(\omega, \xi) \in \mathcal{B}_f(\mathcal{Q})$ from (2.20a). Hence, $\mathcal{B}_f(\mathcal{Q})$ is complete. $\square$

Investigating Proposition 2.18 and 2.19 it is immediately obvious that the two constructions do not always commute. Hence, starting with an ASSDS Σ_S and constructing its induced SM $\mathcal{Q}$, the latter only reproduces Σ_S as its induced ASSDS if Σ_S is complete. This is formalized in the following lemma.

Lemma 2.20. *Let $\Sigma_S = (\mathbb{N}_0, W \times X, \mathcal{B}_S)$ be an ASSDS, $\mathcal{Q} = (X, W, \delta, X_0)$ the SM induced by Σ_S and $\Sigma'_S = (\mathbb{N}_0, W \times X, \mathcal{B}'_S)$ the ASSDS induced by $\mathcal{Q}$. Then*
(i) $\Sigma'_S = \overline{\Sigma_S}$ and
(ii) $\Sigma_S = \Sigma'_S$ iff Σ_S is complete.

Proof. Note that (ii) is an immediate consequence of (i) and Lemma 2.4 (iii). We therefore only prove (i), hence we show that $\mathcal{B}_f(\mathcal{Q}) = \overline{\mathcal{B}_S}$. Fix any $(\omega, \xi) \in \mathcal{B}_f(\mathcal{Q})$ and observe that using (2.21) in (2.20a) implies

$$\exists(\omega_0,\xi_0),(\omega_1,\xi_1)\in\mathcal{B}_S\ .\ \begin{pmatrix} \xi_0|_{[0,1]}=\xi|_{[0,1]}\wedge\omega_0(0)=\omega(0)\\ \wedge\xi_1|_{[1,2]}=\xi|_{[1,2]}\wedge\omega_1(1)=\omega(1)\\ \wedge\xi_0(1)=\xi_1(1) \end{pmatrix}. \tag{2.28}$$

As Σ_S is an ASSDS the last line in (2.28) implies $(\omega_0,\xi_0)\wedge_1(\omega_1,\xi_1)\in\mathcal{B}_S$ (from (2.16)) hence $(\omega,\xi)|_{[0,1]}\in\mathcal{B}_S|_{[0,1]}$. Applying this argument iteratively gives

$$\mathcal{B}_f(\mathcal{Q})=\left\{(\omega,\xi)\middle|\forall k\in\mathbb{N}_0\ .\ (\omega,\xi)|_{[0,k]}\in\mathcal{B}_S|_{[0,k]}\right\}.$$

Therefore, using (2.4), obviously $\mathcal{B}_f(\mathcal{Q})=\overline{\mathcal{B}_S}$. □

For the complementary result to Lemma 2.20 recall from Proposition 2.18 that the SM induced by an ASSDS is always live and reachable. Hence, starting with an SM $\mathcal{Q}$ and constructing its induced ASSDS Σ_S, the latter only reproduces $\mathcal{Q}$ as its induced SM if $\mathcal{Q}$ is live and reachable.

Lemma 2.21. *Let $\mathcal{Q}=(X,W,\delta,X_0)$ be an SM, $\Sigma_S=(\mathbb{N}_0,W\times X,\mathcal{B}_S)$ the ASSDS induced by $\mathcal{Q}$ and $\mathcal{Q}'=(X,W,\delta',X_0')$ the SM induced by Σ_S. Then $\mathcal{Q}=\mathcal{Q}'$ iff $\mathcal{Q}$ is live and reachable.*

Proof. Recall from Proposition 2.19 that $\mathcal{B}_S=\mathcal{B}_f(\mathcal{Q})$. Therefore, using (2.23) in Proposition 2.18 immediately implies $X_0=X_0'$ and $\delta=\delta'$. It is furthermore easy to see that $X_0=X_0'$ and $\delta=\delta'$ with X_0' and δ' as in Proposition 2.18 implies (2.23). □

Example 2.6. We revisit Example 2.1 and assume that the system Σ in (2.5) is additionally equipped with a state space X s.t.

$$\Sigma_S=(\mathbb{N}_0,W\times X,\mathcal{B}_S)\text{ with }X=\{x_a,x_b\}\text{ and }\mathcal{B}_S=\{(a,x_a))^*(b,x_b)^\omega\}. \tag{2.29}$$

It can easily be verified that Σ_S is an ASSDS. Using Proposition 2.18 therefore results in the SM $\mathcal{Q}_1$ induced by Σ_S which is depicted in Figure 2.4 (left). It is obvious that $\mathcal{Q}_1$ is live and reachable. Now applying (2.22) to $\mathcal{Q}_1$ results in its full behavior

$$\mathcal{B}_f(\mathcal{Q}_1)=\{(a,x_a))^\omega,(a,x_a))^*(b,x_b)^\omega\}.$$

Recalling the arguments from Example 2.1 it can be easily observed that

$$\mathcal{B}_f(\mathcal{Q}_1)=\overline{\mathcal{B}_f(\mathcal{Q}_1)}=\overline{\mathcal{B}_S}\supset\mathcal{B}_S.$$

Hence the ASSDS $\Sigma_{S,1}=(\mathbb{N}_0,W\times X,\mathcal{B}_f(\mathcal{Q}_1))$ induced by $\mathcal{Q}_1$ via Proposition 2.19 is complete while Σ_S is not complete implying $\Sigma_S\neq\Sigma_{S,1}$.

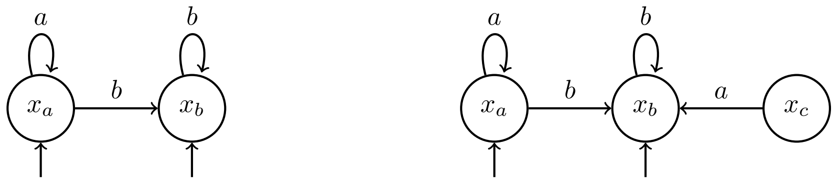

Figure 2.4.: SM $\mathcal{Q}_1$ (left) and $\mathcal{Q}_2$ (right) used in Example 2.6. The former is live and reachable while the latter is not reachable.

Now we consider the SM $\mathcal{Q}_2$ depicted in Figure 2.4 (right). The additional state x_c is obviously not reachable in $\mathcal{Q}_2$ (i.e., $\mathcal{Q}_2$ is not reachable). Constructing the ASSDS induced by $\mathcal{Q}_2$ via Proposition 2.19 results again in $\Sigma_{S,1}$ as $\mathcal{B}_f(\mathcal{Q}_1) = \mathcal{B}_f(\mathcal{Q}_2)$. However, the SM induced by $\Sigma_{S,1}$ is given by $\mathcal{Q}_1$ depicted in Figure 2.4 (left) which is obviously not equivalent to $\mathcal{Q}_2$. ◁

Summarizing the results from Lemma 2.20 and Lemma 2.21 ASSDS and SM can be used interchangeably to model a system if the former is complete and the latter is live and reachable. In this case we call them a realization of one another. As we will occasionally use different external signal spaces in the following chapters we define realizations slightly more general as suggested by the preceding discussion.

Definition 2.22. *Given a dynamical system $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ and an ASSDS $\Sigma_S = (\mathbb{N}_0, W \times X, \mathcal{B}_S)$, let V be a set s.t. $\pi_V(W) \neq \emptyset$ (resp. $\pi_V(W \times X) \neq \emptyset$). Then Σ (resp. Σ_S) is said to be realized by the SM $\mathcal{Q}$ w.r.t. V, if $\pi_V(\mathcal{B}_f(\mathcal{Q})) = \pi_V(\mathcal{B})$ (resp. $\pi_V(\mathcal{B}_f(\mathcal{Q})) = \pi_V(\mathcal{B}_S)$) and $\mathcal{Q}$ is live and reachable.*

2.7. Asynchronous Properties of Dynamical Systems

We have shown in the previous section that the asynchronous state property renders the step-by-step evolution of a dynamical system realizable by a state machine. It now remains to introduce an abstraction method generating an ASSDS rather then an SSDS. Following the spirit of the construction of SlCA and recalling from Section 2.2 that the notions of state, memory span and l-completeness are strongly related, we first derive asynchronous versions of the latter in this section. We furthermore prove their connection in direct analogy to the synchronous case in Section 2.2.

Asynchronous Memory Span

Recall that the concepts of synchronous state property and synchronous memory span are strongly related, since the synchronous state property implies that

$\Sigma_x = (\mathbb{N}_0, X, \pi_X(\mathcal{B}_S))$ has memory span one. To get the same relation for the asynchronous case we define an asynchronous memory span.

Definition 2.23. *The dynamical system* $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ *has* asynchronous memory span l *if*

$$\forall \omega, \omega' \in \mathcal{B}, k, k' \in \mathbb{N}_0 \, . \, \left(\omega|_{[k,k+l-1]} = \omega'|_{[k',k'+l-1]} \Rightarrow \omega \wedge_{k'}^{k} \omega' \in \mathcal{B} \right). \tag{2.30}$$

As expected, it can be easily seen that every system with asynchronous memory span l also has synchronous memory span l.

For systems with an asynchronous memory span the domino game presented in Section 2.2 is significantly simplified. At any point in time k we can attach any domino from the whole domino set $D_{l+1} = \bigcup_{k \in \mathbb{N}_0} \mathcal{B}|_{[k,k+l]}$, as long as the first l symbols of the newly attached domino match the last l symbols of the previous domino. Recall that this implies time independent transitions in the induced evolution law which is what we are aiming at.

Asynchronous l-Completeness

Having in mind the aforementioned simplification of the domino game, the definition of asynchronous l-completeness comes as no surprise.

Definition 2.24. *The dynamical system* $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ *is* asynchronously l-complete *if*

$$\begin{pmatrix} \omega|_{[0,l-1]} \in \mathcal{B}|_{[0,l-1]} \\ \wedge \forall k \in \mathbb{N}_0 \, . \, \omega|_{[k,k+l]} \in \bigcup_{k' \in \mathbb{N}_0} \mathcal{B}|_{[k',k'+l]} \end{pmatrix} \Leftrightarrow \omega \in \mathcal{B}. \tag{2.31}$$

Again, it is easily verified that a system is synchronously l-complete if it is asynchronously l-complete.

Remark 2.6. The second line in (2.31) describes that the possible future evolution of the system depends on the l past values of a signal if $k \geq l$. However, at start up this "past" is not yet fully available. Therefore, the first line in (2.31) is needed to ensure that all signals start with an allowed initial pattern. However, observe that if Σ is time invariant the condition $\sigma\mathcal{B} \subseteq \mathcal{B}$ implies $\forall k \in \mathbb{N}_0 \, . \, \sigma^t \mathcal{B}|_{[k,k+l]} \subseteq \mathcal{B}|_{[0,l]}$ giving $\bigcup_{k' \in \mathbb{N}_0} \mathcal{B}|_{[k',k'+l]} = \mathcal{B}|_{[0,l]}$. Then the first line in (2.31) is implied by the second and is therefore unnecessary. This is stated in the following lemma. ◁

Lemma 2.25. *A* time invariant *dynamical system* $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ *is asynchronously l-complete if*

$$\left(\forall k \in \mathbb{N}_0 \, . \, \omega|_{[k,k+l]} \in \mathcal{B}|_{[0,l]} \right) \Leftrightarrow \omega \in \mathcal{B}. \tag{2.32}$$

Proof. As pointed out in Remark 2.6, time invariance of Σ implies $\bigcup_{k' \in \mathbb{N}_0} \mathcal{B}|_{[k',k'+l]} = \mathcal{B}|_{[0,l]}$. Hence (2.32) and (2.31) are identical. □

Remark 2.7. Recall from Remark 2.1 that in [36] l-completeness for time invariant systems is defined by (2.32) (instead of (2.6)). Therefore, Lemma 2.25 implies that this stronger version of l-completeness from [36] coincides with the property of asynchronous l-completeness for time-invariant systems. ◁

Example 2.7. We now investigate the asynchronous l-completeness properties of the system Σ in (2.7). Since Σ is time invariant, it follows from Remark 2.6 that

$$\bigcup_{k\in\mathbb{N}_0} \mathcal{B}|_{[k,k+l]} = \mathcal{B}|_{[0,l]}.$$

Therefore, the simplified domino game for $l = 1$ is identical to the one played in Example 2.2 implying that the system (2.7) is not asynchronously 1-complete. For $l = 2$, observe that in the simplified domino game we are still allowed to use the piece aaa from the set $\mathcal{B}|_{[0,2]}$ at any time $k > 0$. Therefore, more than two sequential a's can be produced by this game implying that the system (2.7) is *not* asynchronously 2-complete. Extending l to $l = 3$ gives the set

$$\bigcup_{k'\in\mathbb{N}_0} \mathcal{B}|_{[k',k'+3]} = \mathcal{B}|_{[0,3]} = \{aaab, aaba, abaa, baab\}.$$

Now, playing the simplified domino game ensures that always three symbols have to match, preventing the piece $aaab$ to be attachable for $k > 0$. Hence, the resulting behavior is identical to $\mathcal{B}$. This implies that the system (2.7) is asynchronously 3-complete. ◁

Asynchronous l-Completeness vs. Asynchronous Memory Span

Recall from Proposition 2.8 that a synchronously l-complete system always has synchronous memory span l whereas the reverse implication only holds if the system is complete. To emphasize that the asynchronous properties extend the behavioral systems theory in a consistent way, we prove the same correspondence for the asynchronous case.

Proposition 2.26. *Let $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ be a dynamical system and $l \in \mathbb{N}_0$. Then*

$$\Sigma \text{ is asynchronously } l\text{-complete} \Leftrightarrow \begin{pmatrix} \Sigma \text{ is complete} \\ \wedge \Sigma \text{ has asynchronous memory span } l \end{pmatrix}. \tag{2.33}$$

Proof. "⇒" As asynchronous l-completeness implies synchronized l-completeness, it also implies completeness. To show that asynchronous l-completeness of Σ implies (2.30), we fix $\omega_1, \omega_2 \in \mathcal{B}$, $k_1, k_2 \in \mathbb{N}_0$ s.t. $\omega_1|_{[k_1,k_1+l-1]} = \omega_2|_{[k_2,k_2+l-1]}$ and show that $\omega = \omega_1 \wedge^{k_1}_{k_2} \omega_2 \in \mathcal{B}$ follows. We distinguish two cases:

- $k_1 = 0$: Observe that $\omega = \omega_1 \wedge^0_{k_2} \omega_2 = \omega_2|_{[k_2,\infty)}$, hence

$$\omega|_{[0,l-1]} = \omega_2|_{[k_2,k_2+l-1]} = \omega_1|_{[0,l-1]} \in \mathcal{B}|_{[0,l-1]} \text{ and}$$
$$\forall k \in \mathbb{N}_0 \,.\, \omega|_{[k,k+l]} \in \mathcal{B}|_{[k+k_2,k+k_2+l]} \subseteq \bigcup_{k'\in\mathbb{N}_0} \mathcal{B}|_{[k',k'+l]}.$$

  With asynchronous l-completeness of Σ it follows that $\omega \in \mathcal{B}$ (from (2.31)).
- $k_1 \neq 0$: Observe that for all $k \in \mathbb{N}_0$

$$\omega|_{[k,k+l]} = \begin{cases} \omega_1|_{[k,k+l]} \in \mathcal{B}|_{[k,k+l]} \subseteq \bigcup_{k'\in\mathbb{N}_0} \mathcal{B}|_{[k',k'+l]} & , \ k < k_1 \\ \omega_2|_{[k',k'+l]} \in \mathcal{B}|_{[k',k'+l]} \subseteq \bigcup_{k'\in\mathbb{N}_0} \mathcal{B}|_{[k',k'+l]} & , \ k \geq k_1 \end{cases},$$

  with $k' = k - k_1 + k_2$. Hence $\omega|_{[0,l-1]} = \omega_1|_{[0,l-1]} \in \mathcal{B}|_{[0,l-1]}$ and $\forall k \in \mathbb{N}_0 \,.\, \omega|_{[k,k+l]} \in \bigcup_{k'\in\mathbb{N}_0} \mathcal{B}|_{[k',k'+l]}$. With asynchronous l-completeness of Σ it follows that $\omega \in \mathcal{B}$ (from (2.31)).

"$\Leftarrow$" To show that (2.31) holds for a complete system Σ with asynchronous memory span l, we fix $\omega \in W^{\mathbb{N}_0}$ s.t. the left hand side of (2.31) holds and show that $\omega \in \mathcal{B}$ follows if Σ is complete and has asynchronous memory span l (as the inverse direction in (2.31) always holds). Observe that the left hand side of (2.31) implies

$$\exists \omega_0, \omega_1, \omega_2, \ldots \in \mathcal{B}, k', k'', \ldots \in \mathbb{N}_0 \,.\, \begin{pmatrix} \omega_0|_{[0,l]} = \omega|_{[0,l]} \\ \wedge \omega_1|_{[k',k'+l]} = \omega|_{[1,1+l]} \\ \wedge \omega_2|_{[k'',k''+l]} = \omega|_{[2,2+l]} \\ \wedge \quad \ldots \end{pmatrix}. \tag{2.34}$$

hence $\omega_0|_{[1,l]} = \omega_1|_{[k',k'+l-1]}$ and $\omega_1|_{[k'+1,k'+l]} = \omega_2|_{[k'',k''+l-1]}$. As Σ has asynchronous memory span l, this implies $\omega_0 \wedge^1_{k'} \omega_1 \in \mathcal{B}$ and $\omega_1 \wedge^{k'+1}_{k''} \omega_2 \in \mathcal{B}$ and therefore

$$\left(\omega_0 \wedge^1_{k'} \omega_1 \wedge^{k'+1}_{k''} \omega_2\right)|_{[0,l+2]} = \omega|_{[0,l+2]} \in \mathcal{B}|_{[0,l+2]}.$$

Iteratively applying this procedure therefore yields $\forall \tau \in \mathbb{N}_0 \,.\, \omega|_{[0,\tau]} \in \mathcal{B}|_{[0,\tau]}$ implying $\omega \in \mathcal{B}$ as Σ is complete. □

ASSDS and SM of Asynchronously l-Complete Systems

Recall from the simplified domino game that the memory of an asynchronously l-complete system is still given by the last l symbols of the last domino. Hence, we can construct a state space representation for the asynchronous case equivalently to Proposition 2.12.

Proposition 2.27. *Let $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ be a dynamical system with asynchronous memory span l. Then $\Sigma_S = (\mathbb{N}_0, W, X, \mathcal{B}_S)$ from Proposition 2.12 is an asynchronous state space representation of Σ.*

Proof. $\pi_W(\mathcal{B}_S) = \mathcal{B}$ holds by construction from Proposition 2.12. To show (2.25) we pick $(\omega_1, \xi_1), (\omega_2, \xi_2) \in \mathcal{B}_S$ and $k', k'' \in \mathbb{N}_0$ s.t. $\xi_1(k') = \xi_2(k'')$ and show $(\omega, \xi) = (\omega_1, \xi_1) \wedge_{k''}^{k'} (\omega_2, \xi_2) \in \mathcal{B}_S$.

We first observe that

$$\xi_1(k') = \omega_1|_{[\max\{0,k'-l\},k'-1]} = \omega_2|_{[\max\{0,k''-l\},k''-1]} = \xi_2(k'') \tag{2.35}$$

and show $\omega \in \mathcal{B}$ using two different cases:
- If $k', k'' \leq l$, (2.35) implies $k' = k''$ and $\omega = \omega_1 \wedge_{k''}^{k'} \omega_2 = \omega_2 \in \mathcal{B}$.
- If $k' \leq l$, $k'' > l$ or $k' > l$, $k'' \leq l$ (2.35) implies $k' = l$ or $k'' = l$, respectively. As $k', k'' > l$, this implies that $\omega \in \mathcal{B}$ follows from Σ having asynchronous memory span[5] l.

It remains to show that ξ satisfies (2.18). Remember that (2.18) holds for (ω_1, ξ_1), $(\omega_2, \xi_2) \in \mathcal{B}_S$. Therefore, $\xi = \xi_1 \wedge_{k''}^{k'} \xi_2$ implies that for all $k \in \mathbb{N}_0$

$$\xi(k) = \begin{cases} \omega_1|_{[0,k-1]} & (k \leq k') \wedge (k < l) \\ \omega_1|_{[k-l,k-1]} & (k \leq k') \wedge (k \geq l) \\ \omega_1|_{[0,k'-1]} \cdot \omega_2|_{[k'',k-k'+k''-1]} & (k' < k < k'+l) \wedge (k < l) \\ \omega_1|_{[k-l,k'-1]} \cdot \omega_2|_{[k'',k-k'+k''-1]} & (k' < k < k'+l) \wedge (k \geq l) \\ \omega_2|_{[k-k'+k''-l,k-k'+k''-1]} & (k \geq k'+l)\,. \end{cases}$$

Hence, with $\omega = \omega_1 \wedge_{k''}^{k'} \omega_2 \in \mathcal{B}$ we know that $\xi = \xi_1 \wedge_{k''}^{k'} \xi_2$ satisfies (2.18). $\square$

As we already know from Proposition 2.18 that every ASSDS allows for a SM realization, we can construct the latter in direct analogy to Proposition 2.16.

Proposition 2.28. *Let $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ be an asynchronously l-complete dynamical system and $\Sigma_S = (\mathbb{N}_0, W, X, \mathcal{B}_S)$ its asynchronous state space representation from Proposition 2.27. Then $\mathcal{Q} = (X, W, \delta, X_0)$ with $X_0 = \{\lambda\}$ and*

$$\begin{aligned} \delta = &\left\{(x, w, x \cdot w) \middle| |x|_L < l \wedge x \cdot w \in \mathcal{B}|_{[0,|x|_L]}\right\} \\ &\cup \left\{(x, w, (x \cdot w)|_{[1,l-1]}) \middle| |x|_L = l \wedge x \cdot w \in \textstyle\bigcup_{k' \in \mathbb{N}_0} \mathcal{B}|_{[k',k'+l]}\right\} \end{aligned} \tag{2.36}$$

realizes Σ_S w.r.t. $W \times X$.

[5]Observe that, under the premises of Definition 2.23, $\omega_1 \wedge_{k''}^{k'} \omega_2 = \omega_1 \wedge_{k''+l}^{k'+l} \omega_2$ in the right side of the implication in (2.30).

Proof. Observe that $\delta = \bigcup_{k\in\mathbb{N}_0} \psi(k)$ with $\psi(k)$ from (2.24) and recall from Proposition 2.16 that Σ_ψ is the induced evolution law of Σ_S. Furthermore, Proposition 2.18 implies that $\mathcal{Q}$ is the induced evolution law of Σ_S and $\mathcal{Q}$ is live and reachable. Now l-completeness of Σ implies completeness of Σ_S.Therefore it follows from Lemma 2.20 (ii) and Definition 2.22 that Σ_S is realized by the SM $\mathcal{Q} = (X, W, \delta, X_0)$ w.r.t. $W \times X$. □

Remark 2.8. The next state relation δ in (2.36) can be interpreted analogously to ψ in (2.24), see the discussion in Remark 2.4. Observe that now the condition in the last line of (2.36) is weakened w.r.t. (2.24) in the sense that $x \cdot w$ can be any domino in the gedankenexperiment in Section 2.2. ◁

Example 2.8. Recall from Example 2.7 that the system (2.7) in Example 2.2 is asynchronously 3-complete and that (2.9) implies $\bigcup_{k\in\mathbb{N}_0} \mathcal{B}|_{[t,k+2]} = \{aaa, aab, aba, baa\}$. Adding the set $\bigcup_{r\in[0,2]} \mathcal{B}|_{[0,r-1]} = \{\lambda, a, b, aa, ab, ba\}$, the state space defined in Proposition 2.12 with $l = 3$ for the system in (2.7) is given by

$$X = \{\lambda, a, b, aa, ab, ba, aaa, aab, aba, baa\}.$$

Using this state space and the construction of the next state relation in (2.36), we can construct an FSM $\mathcal{Q}$ realizing the system (2.7) in Example 2.2. The result is depicted in Figure 2.5. ◁

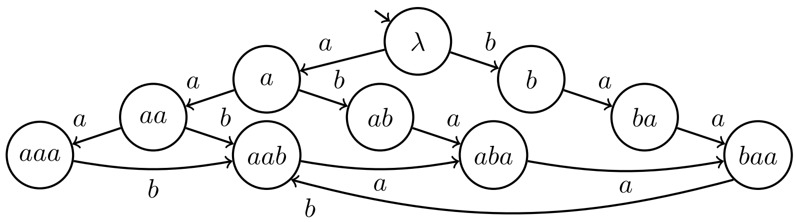

Figure 2.5.: FSM $\mathcal{Q}$ realizing (2.7).

Remark 2.9. Observe that (2.10) in Lemma 2.6 and (2.32) in Lemma 2.25 are identical. Therefore, Lemma 2.6 and Lemma 2.25 imply that the asynchronous and the synchronous l-completeness property coincide for strictly time invariant systems. As a direct consequence, the state space representation of a strictly time invariant (synchronously) l-complete system can be realized by the FSM $\mathcal{Q}$ constructed in Proposition 2.28. ◁

2.8. Strongest Asynchronous l-Complete Approximations (SAlCA)

It was shown in Section 2.7 that asynchronously l-complete dynamical systems can be represented by an ASSDS which allows for realizability by an SM. Hence, abstracting a dynamical system by an asynchronously l-complete one allows to realize the latter by an SM. Motivated by this we construct asynchronous l-complete approximations analogously to the synchronous versions in Definition 2.9.

Definition 2.29. *Let $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ be a dynamical system. Then $\Sigma' = (\mathbb{N}_0, W, \mathcal{B}')$ is an* asynchronous l-complete approximation *of Σ if (i) Σ' is asynchronously l-complete and (ii) $\mathcal{B}' \supseteq \mathcal{B}$. Furthermore, $\Sigma' = (\mathbb{N}_0, W, \mathcal{B}')$ is the* strongest asynchronous l-complete approximation *of Σ if (i) Σ' is an asynchronous l-complete approximation of Σ and (ii) for any asynchronous l-complete approximation $\Sigma'' = (\mathbb{N}_0, W, \mathcal{B}'')$ of Σ it holds that $\mathcal{B}' \subseteq \mathcal{B}''$.*

Recall that for an asynchronously l-complete system the domino game gedankenexperiment can be simplified such that at any time k we can attach any domino from the whole domino set

$$D_{l+1} = \bigcup_{k\in\mathbb{N}_0} \mathcal{B}|_{[k,k+l]}. \tag{2.37}$$

This simplified domino game now constructs the unique strongest asynchronous l-complete approximation $\widehat{\mathcal{B}}^{l\uparrow}$.

Theorem 2.30. *Let $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ be a dynamical system. Then the unique strongest asynchronous l-complete approximation (SAlCA) of Σ is given by $\widehat{\Sigma}^{l\uparrow} = (\mathbb{N}_0, W, \widehat{\mathcal{B}}^{l\uparrow})$, with*

$$\widehat{\mathcal{B}}^{l\uparrow} := \left\{ \omega \in W^{\mathbb{N}_0} \middle| \begin{pmatrix} \omega|_{[0,l-1]} \in \mathcal{B}|_{[0,l-1]} \\ \wedge \forall k \in \mathbb{N}_0 \;.\; \omega|_{[k,k+l]} \in \bigcup_{k'\in\mathbb{N}_0} \mathcal{B}|_{[k',k'+l]} \end{pmatrix} \right\}. \tag{2.38}$$

Furthermore, if Σ is time invariant then

$$\widehat{\mathcal{B}}^{l\uparrow} = \left\{ w \in W^{\mathbb{N}_0} \middle| \forall k \in \mathbb{N}_0 \;.\; w|_{[k,k+l]} \in \mathcal{B}|_{[0,l]} \right\}. \tag{2.39}$$

Proof. The proof of the first part follows the same lines as the proof of Theorem 2.10, and the second part follows directly from Lemma 2.25. □

Remark 2.10. As a direct consequence of Remark 2.9, the strongest synchronous and the strongest asynchronous l-complete approximation coincide for strictly time invariant systems. This can be easily seen by noting that (2.15) in Theorem 2.10 and (2.39) in Theorem 2.30 coincide.

Furthermore, recall from Remark 2.7 that the stronger notion of l-completeness from [36] coincides with the property of asynchronous l-completeness for time-invariant systems. Therefore, the strongest l-complete approximation of a time invariant system Σ suggested in [36] is identical to its strongest asynchronous l-complete approximation $\widehat{\Sigma}^{l\uparrow}$ introduced in Definition 2.29. The latter is, by definition, also a synchronous l-complete approximation, but not necessarily the strongest one. ◁

Remark 2.11. The construction of $\widehat{\Sigma}^{l\uparrow}$ relies on the computation of the set $\bigcup_{k\in\mathbb{N}_0} \mathcal{B}|_{[k,k+l]}$, which is in general not a trivial task. However, a finite external signal space W implies the existence of a finite time $k_{ti} \in \mathbb{N}_0$ s.t. Σ becomes time invariant for $k > k_{ti}$. For this case, [35, p.51] proposes a pragmatically motivated extension of the strongest l-complete approximation in the sense of [36]. It can be easily observed that the behavior $\mathcal{B}_l^\star$ constructed in [35, p.51] coincides with $\widehat{\mathcal{B}}^{l\uparrow}$ in (2.38). As the approach in [35, p.51] is pragmatically motivated this coincidence underlines the relevance of our theoretically developed notion. ◁

Properties of SAlCA

As we will extensively use SAlCA in Part I of this thesis we investigate this abstraction more closely than SlCA.

Lemma 2.31. *Given a dynamical system $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ and its SAlCA $\widehat{\Sigma}^{l\uparrow} = (\mathbb{N}_0, W, \widehat{\mathcal{B}}^{l\uparrow})$ it holds that*

(i) $\mathcal{B} \subseteq \widehat{\mathcal{B}}^{l+1\uparrow} \subseteq \widehat{\mathcal{B}}^{l\uparrow}$,

(ii) $\mathcal{B}|_{[0,l-1]} = \widehat{\mathcal{B}}^{l\uparrow}|_{[0,l-1]}$,

(iii) $\bigcup_{k\in\mathbb{N}_0} \mathcal{B}|_{[k,k+l]} = \bigcup_{k\in\mathbb{N}_0} \widehat{\mathcal{B}}^{l\uparrow}|_{[k,k+l]}$, *and*

(iv) $\mathcal{B} = \widehat{\mathcal{B}}^{l\uparrow} \Leftrightarrow \Sigma$ *is asynchronously l-complete.*

Proof. We show all statements separately.

(i) Pick $\omega \in \mathcal{B}$ and observe that $\omega|_{[0,l]} \in \mathcal{B}|_{[0,l]}$ and $\forall k \in \mathbb{N}_0 \,.\, \omega|_{[k,k+l+1]} \in \mathcal{B}|_{[k,k+l+1]} \subseteq \bigcup_{k'\in\mathbb{N}_0} \mathcal{B}|_{[k',k'+l+1]}$. Hence, (2.38) implies $\omega \in \widehat{\mathcal{B}}^{l+1\uparrow}$. Similarly, pick $\omega \in \widehat{\mathcal{B}}^{l+1\uparrow}$, implying $\omega|_{[0,l-1]} \in \mathcal{B}|_{[0,l-1]}$ (as $\omega|_{[0,l]} \in \mathcal{B}|_{[0,l]}$) and $\forall k \in \mathbb{N}_0 \,.\, \omega|_{[k,k+l+1]} \in \mathcal{B}|_{[k,k+l+1]}$ and therefore $\forall k \in \mathbb{N}_0 \,.\, \omega|_{[k,k+l]} \in \mathcal{B}|_{[k,k+l]} \subseteq \bigcup_{k'\in\mathbb{N}_0} \widehat{\mathcal{B}}^{l\uparrow}|_{[k',k'+l]}$. With this (2.38) implies $\omega \in \widehat{\mathcal{B}}^{l\uparrow}$.

(ii) "$\subseteq$" follows immediately from (i). To show "$\supseteq$" we pick $\tilde{\omega} \in \widehat{\mathcal{B}}^{l\uparrow}|_{[0,l-1]}$. This implies that there exists $\omega \in \widehat{\mathcal{B}}^{l\uparrow}$ s.t. $\omega|_{[0,l-1]} = \tilde{\omega}$. Using (2.38) therefore yields $\tilde{\omega} \in \mathcal{B}|_{[0,l-1]}$.

(iii) "$\subseteq$" follows immediately from (i). To show "$\supseteq$" we pick $\tilde{\omega} \in \bigcup_{k\in\mathbb{N}_0} \widehat{\mathcal{B}}^{l\uparrow}|_{[k,k+l]}$. This implies that there exists $\omega \in \widehat{\mathcal{B}}^{l\uparrow}$ and $k \in \mathbb{N}_0$ s.t. $\omega|_{[k,k+l]} = \tilde{\omega}$. Using (2.38) therefore yields $\tilde{\omega} \in \bigcup_{k\in\mathbb{N}_0} \mathcal{B}|_{[k,k+l]}$.

(iv) "$\Rightarrow$" Observe that "$\Leftarrow$" in (2.31) always holds. We therefore pick ω s.t. the left side of (2.31) holds. Using (2.38) this implies $\omega \in \widehat{\mathcal{B}}^{l\uparrow}$. As $\mathcal{B} = \widehat{\mathcal{B}}^{l\uparrow}$ we have $\omega \in \mathcal{B}$. Hence, Σ is asynchronously l-complete.
"$\Leftarrow$" Observe that $\mathcal{B} \subseteq \widehat{\mathcal{B}}^{l\uparrow}$ always holds from (i). We therefore pick $\omega \in \widehat{\mathcal{B}}^{l\uparrow}$. Using (2.38) this implies that the left side of (2.31) holds for ω. As Σ is asynchronously l-complete, this implies $\omega \in \mathcal{B}$. Hence, $\widehat{\mathcal{B}}^{l\uparrow} \subseteq \mathcal{B}$ implying $\mathcal{B} = \widehat{\mathcal{B}}^{l\uparrow}$.

□

As the SAlCA of Σ is constructed from the set of dominos D_{l+1} with length $l+1$ which can be obtained locally from $\mathcal{B}$, the resulting abstraction coincides with the one obtained from the ω-closure $\overline{\Sigma}$ of Σ.

Proposition 2.32. *Given a dynamical system $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ and its ω-closure $\overline{\Sigma} = (\mathbb{N}_0, W, \overline{\mathcal{B}})$ it holds that*

$$\widehat{\Sigma}^{l\uparrow} = \widehat{\overline{\Sigma}}^{l\uparrow}. \tag{2.40}$$

Proof. Observe that Lemma 2.4 (ii) implies $\forall k \in \mathbb{N}_0 \,.\, \mathcal{B}|_{[k,k+l]} = \overline{\mathcal{B}}|_{[k,k+l]}$. Hence,

$$\mathcal{B}|_{[0,l-1]} = \overline{\mathcal{B}}|_{[0,l-1]} \text{ and } \bigcup_{k\in\mathbb{N}_0} \mathcal{B}|_{[k,k+l]} = \bigcup_{k\in\mathbb{N}_0} \overline{\mathcal{B}}|_{[k,k+l]}.$$

Substituting these equalities in (2.38) immediately yields $\widehat{\mathcal{B}}^{l\uparrow} = \widehat{\overline{\mathcal{B}}}^{l\uparrow}$. □

Realizing SAlCA

As SAlCA are asynchronously l-complete by construction they can always be realized by the SM constructed in Proposition 2.28. Hence, the construction of a finite state machine $\widehat{Q}^{l\uparrow}$ from a given dynamical system Σ (with finite external signal space W) using SAlCA can be summarized by the following corollary and is illustrated in Figure 2.6. The corollary can be derived from the results Figure 2.6 refers to.

Corollary 2.33. *The SAlCA $\widehat{\Sigma}^{l\uparrow} = (\mathbb{N}_0, W, \widehat{\mathcal{B}}^{l\uparrow})$ of a dynamical system $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ with finite external signal space W can be represented by the ASSDS $\widehat{\Sigma}^{l\uparrow}_S = (\mathbb{N}_0, W, \widehat{X}^{l\uparrow}, \widehat{\mathcal{B}}^{l\uparrow}_S)$ with $\widehat{X}^{l\uparrow}$ and $\widehat{\mathcal{B}}^{l\uparrow}_S$ as in Proposition 2.12. Furthermore, $\widehat{\Sigma}^{l\uparrow}_S$ can be realized by the FSM $\widehat{Q}^{l\uparrow} = (\widehat{X}^{l\uparrow}, W, \widehat{\delta}^{l\uparrow}, \widehat{X}^{l\uparrow}_0)$ given in Proposition 2.28.*

Summarizing the construction of the "standard" realization $\widehat{Q}^{l\uparrow}$ of the SAlCA of Σ, as depicted in Figure 2.6, it obviously suffices to know the set of dominos

$$D_{l+1} = \bigcup_{k\in\mathbb{N}_0} \mathcal{B}|_{[k,k+l]} = \bigcup_{k\in\mathbb{N}_0} \overline{\mathcal{B}}|_{[k,k+l]} \tag{2.41}$$

obtained from Σ or $\overline{\Sigma}$ to construct $\widehat{Q}^{l\uparrow}$. This observation is indicated by the dashed line in Figure 2.6 and summarized in the following corollary. This corollary can be derived

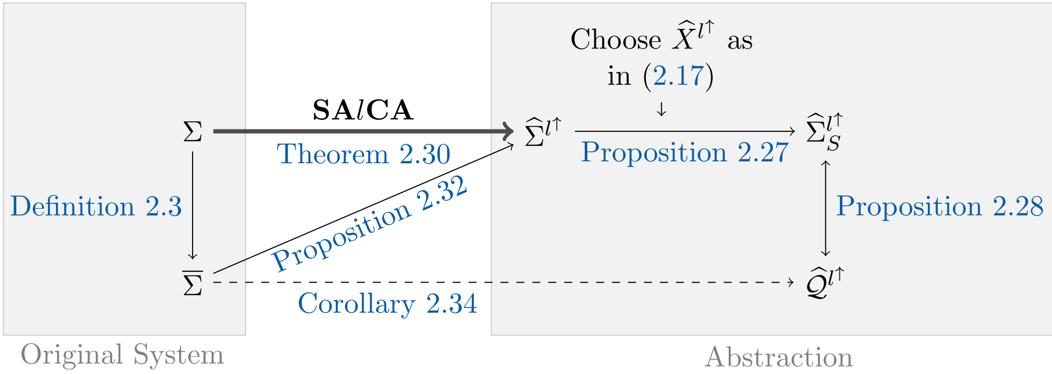

Figure 2.6.: Illustration of the construction of an SM $\widehat{\mathcal{Q}}^{l\uparrow}$ realizing the SAlCA of a given dynamical system Σ. The dashed line illustrates the "shortcut" formalized in Corollary 2.34.

from the results illustrated in Figure 2.6 and will be revisited in Chapter 6 where a new set of realizations for SAlCA is introduced.

Corollary 2.34. *Given a dynamical system $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ s.t. $|W| \neq \infty$, the SAlCA of Σ is realized by the finite, reachable and live state machine $\widehat{\mathcal{Q}}^{l\uparrow} = (\widehat{X}^{l\uparrow}, W, \widehat{\delta}^{l\uparrow}, \widehat{X}_0^{l\uparrow})$ s.t.*

$$\widehat{X}^{l\uparrow} = \left(\bigcup_{r\in[0,l-1]} \overline{\mathcal{B}}|_{[0,r-1]}\right) \cup \left(\bigcup_{t\in\mathbb{N}_0} \overline{\mathcal{B}}|_{[t,t+l-1]}\right), \tag{2.42a}$$

$$\widehat{X}_0^{l\uparrow} = \{\lambda\}, \text{ and} \tag{2.42b}$$

$$\widehat{\delta}^{l\uparrow} = \left\{(\widehat{x}, w, \widehat{x}\cdot w) \middle| |\widehat{x}|_L < l \wedge \widehat{x}\cdot w \in \overline{\mathcal{B}}|_{[0,|\widehat{x}|_L]}\right\} \cup \left\{(\widehat{x}, w, (\widehat{x}\cdot w)|_{[1,l-1]}) \middle| |\widehat{x}|_L = l \wedge \widehat{x}\cdot w \in \textstyle\bigcup_{t'\in\mathbb{N}_0} \overline{\mathcal{B}}|_{[t',t'+l]}\right\}. \tag{2.42c}$$

$\overline{\mathcal{B}}$ and $\mathcal{B}$ can be used interchangeably in (2.42).

Concluding the Example

The behaviors constructed by the domino games discussed in Example 2.7 characterize the strongest asynchronous 1-, 2- and 3-complete approximations for the system Σ in (2.7), respectively. Realizations for the strongest asynchronous 1- and 2-complete approximations using the constructions from Proposition 2.28 are shown in Figure 2.7.

As the system Σ in (2.7) is asynchronously 3-complete, its behavior coincides with that of its strongest asynchronous 3-complete approximation. Hence, the corresponding FSM is shown in Figure 2.5. Observe that the FSM realizing $\widehat{\Sigma}_S^{1\uparrow}$ and the evolution law Σ_{ψ_1}

realizing $\widehat{\Sigma}_S^{1\Uparrow}$ depicted in Figure 2.7 (left) and Figure 2.3 (left), respectively, coincide. This is a direct consequence from Remark 2.9 since $\widehat{\Sigma}^{1\Uparrow}$ is strictly time invariant as discussed in Example 2.3.

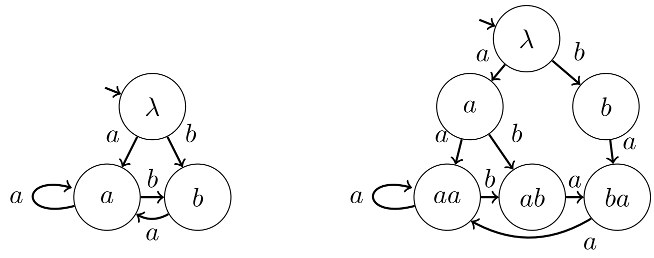

Figure 2.7.: FSMs realizing the strongest asynchronous 1-complete approximation $\widehat{\Sigma}_S^{1\uparrow}$ (left) and the strongest asynchronous 2-complete approximation $\widehat{\Sigma}_S^{2\uparrow}$ (right) of (2.7).

Summarizing the results of our running example, we have the following. The system under consideration, $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ in (2.7), is time invariant but not strictly time invariant. Σ is synchronously 2-complete and can therefore be realized by the time dependent evolution law Σ_{ψ_2} depicted in Figure 2.3 (right). It is asynchronously 3-complete (but not asynchronously 2-complete) and can therefore be realized by the FSM depicted in Figure 2.5. Its strongest asynchronous 2-complete approximation $\widehat{\Sigma}^{2\uparrow}$ is of cause also a synchronous 2-complete approximation of Σ, but not the strongest one. In fact, as Σ is synchronously 2-complete and asynchronously 3-complete, $\widehat{\Sigma}^{2\Uparrow} = \widehat{\Sigma}^{3\uparrow} = \Sigma$, and therefore $\widehat{\mathcal{B}}^{2\Uparrow} = \widehat{\mathcal{B}}^{3\uparrow} = \mathcal{B} \subset \widehat{\mathcal{B}}^{2\uparrow}$.

3. Quotient Based Abstractions (QBA)

In this chapter we briefly outline the basic construction of quotient based abstractions (QBA). In Section 3.1 we discuss them in their usual setting without reference to behavioral systems theory and assume the "original" system to be modeled by a transition system. Thereafter, in Section 3.2, we draw the connection between transition systems and state machines. As the latter model the step-by-step evolution of discrete behaviors, this builds a bridge to behavioral systems theory. Finally, we show in Section 3.3 how to fully reduce the construction of QBA to state machines to obtain a consistent notation for SAlCA and QBA in subsequent chapters.

3.1. Transition System Based Construction

We start this section on the construction of QBA by modeling the system to be abstracted by a so called transition system.

Definition 3.1. *A* transition system *(TS) is a tuple $\mathcal{T} = (X, X_0, U, f, Y, h)$, where X is the set of states, X_0 is the set of initial states, U is the set of inputs, Y is the set of outputs, $f : X \times U \to 2^X$ is a transition map and $h : X \to Y$ is an output function.*

Observe that TS are output deterministic by definition as h is a single valued function. This implies that whenever Y is finite it induces a finite partition on the state space of $\mathcal{T}$. This observation motivates the use of Y as the state space of the QBA of $\mathcal{T}$ which results in an identity output function of this abstraction.

Definition 3.2. *Given a TS $\mathcal{T} = (X, X_0, U, f, Y, h)$, its quotient TS is defined by $\widehat{\mathcal{T}}^{\triangledown} = (\widehat{X}^{\triangledown}, \widehat{X}_0^{\triangledown}, U, \widehat{f}, Y, \widehat{h})$ s.t.*

$$\widehat{X}^{\triangledown} := \{y \in Y | \exists x \in X \,.\, y = h(x)\}\,, \tag{3.1a}$$

$$\widehat{X}_0^{\triangledown} := \{y \in Y | \exists x \in X_0 \,.\, y = h(x)\}\,, \tag{3.1b}$$

$$\widehat{f}(\widehat{x}, u) := \left\{ \widehat{x}' \middle| \exists x, x' \in X \,.\, \begin{pmatrix} \widehat{x} = h(x) \\ \wedge \widehat{x}' = h(x') \\ \wedge x' \in f(x, u) \end{pmatrix} \right\},\ \textit{and} \tag{3.1c}$$

$$\widehat{h}(\widehat{x}) = \widehat{x}. \tag{3.1d}$$

3.2. State Machines vs. Transition Systems

Both state machines (SM) from Definition 2.15 and transition systems (TS) from Definition 3.1 model the step by step evolution of a system. While SM are equipped with an (unspecified) signal space W which might influence or be influenced by the state evolution of this SM, TS model the evolution of a system by two different maps. I.e., the input (which is assumed to be fed into the system by a controller) influences the evolution of the state and the output is generated by the system based on the current state. To obtain a connection between SM and TS we use the following assumption.

Assumption 1. *The set of external symbols W is composed of a set of input symbols U and a set of output symbols Y, hence $W = U \times Y$.*

Before we relate SM and TS we use Assumption 1 to define some properties of SM with inputs and outputs.

Definition 3.3. *Let $\mathcal{Q} = (X, U \times Y, \delta, X_0)$ be an SM. Then the sets of* admissible inputs and outputs *of a state $x \in X$ are defined by*[1]

$$\mathrm{U}_\delta(x) := \left\{u{\in}U \middle| \exists y{\in}Y, x'{\in}X \;.\; (x,u,y,x'){\in}\delta\right\} \quad \textit{and} \tag{3.2a}$$
$$\mathrm{H}_\delta(x) := \left\{y \in Y \middle| \exists u \in U, x' \in X \;.\; (x,u,y,x') \in \delta\right\}, \tag{3.2b}$$

respectively. Furthermore, $\mathcal{Q}$ is said to be output deterministic *if*

$$\forall x \in X \;.\; \mathrm{H}_\delta(x) \neq \emptyset \Rightarrow |\mathrm{H}_\delta(x)| = 1 \tag{3.2c}$$

and $\mathcal{Q}$ is said to have separable dynamics *if*

$$\forall x, x' \in X, u \in U \;.\; \big((x,u,y,x') \in \delta \Rightarrow \forall y' \in \mathrm{H}_\delta(x) \;.\; (x,u,y',x') \in \delta\big). \tag{3.2d}$$

Intuitively, $\mathcal{Q}$ has separable dynamics, i.e., (3.2d) holds, if the transition relation δ can be *separated* into a transition map f and an output function h. By investigating (3.2c) and (3.2d) it is obvious that an output deterministic SM always has separable dynamics. However, if $\mathcal{Q}$ is not output deterministic the latter is not always true as illustrated in the following example.

Example 3.1. Consider the state machine $\mathcal{Q}$ depicted in Figure 3.1. Now observe that $(x_1, u_1, y_1, x_2) \in \delta$ and $\mathrm{H}_\delta(x_1) = \{y_1, y_2\}$, but $(x_1, u_1, y_2, x_2) \notin \delta$, showing that (3.2d) does not hold. ◁

If we consider state machines with inputs and outputs, the only difference between TS and SM is given by the fact that the former requires the transition map f and the

[1] Slightly abusing notation, we denote the triple (x, w, x') with $w = (u, y)$ simply by a quadruple (x, u, y, x'), meaning $(x, (u, y), x')$.

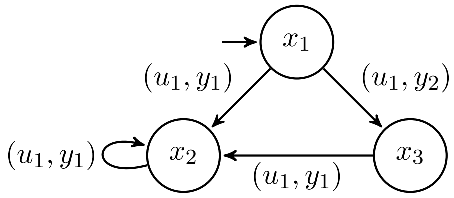

Figure 3.1.: State machine $\mathcal{Q}$ considered in Example 3.1.

output map h to be separable while the latter allows the next state to be dependent on the last output via the next state relation δ. Intuitively, both versions coincide if $\mathcal{Q}$ has separable dynamics. As the latter is implied by output determinism and we have defined TS to be output deterministic, an output deterministic SM can be equivalently modeled by a TS. To formalize this we first define how to "rewrite" SM in TS and and vice versa.

Definition 3.4. *Let $\mathcal{Q} = (X, U \times Y, \delta, X_0)$ be an output deterministic state machine. Then $\mathcal{Q}$ defines the transition system $\mathcal{T} = (X, X_0, U, f, Y, h)$ s.t.*

$$x' \in f(x, u) \;\Leftrightarrow\; \exists y \in Y \,.\, (x, u, y, x') \in \delta \tag{3.3a}$$

$$y = h(x) \;\Leftrightarrow\; y \in \mathrm{H}_\delta(x) \tag{3.3b}$$

Definition 3.5. *The transition system $\mathcal{T} = (X, X_0, U, f, Y, h)$ defines the state machine $\mathcal{Q} = (X, U \times Y, \delta, X_0)$ s.t.*

$$\delta := \left\{(x, u, y, x') \middle| x' \in f(x, u) \wedge y = h(x)\right\}. \tag{3.4}$$

Using these definitions we have the following obvious result.

Lemma 3.6. *Let $\mathcal{Q}_1$ be an output deterministic SM, $\mathcal{T}_1$ the TS defined by $\mathcal{Q}_1$ via Definition 3.4 and $\mathcal{Q}_2$ the SM defined by $\mathcal{T}_1$ via Definition 3.5. Then $\mathcal{Q}_1 = \mathcal{Q}_2$.*

Proof. We pick x, x', u and y and show that $(x, u, y, x') \in \delta_1 \Leftrightarrow (x, u, y, x') \in \delta_2$:
"$\Rightarrow$" (3.3) gives $x' \in f_1(x, u)$ and $y = h_1(x)$, hence (using (3.4)) $(x, u, y, x') \in \delta_2$.
"$\Leftarrow$" (3.4) gives $x' \in f_1(x, u)$ and $y = h_1(x)$. Using (3.3), there exists $\tilde{y}$ s.t. $(x, u, \tilde{y}, x') \in \delta_1$ and using (3.2b), we know that $\tilde{y} \in \mathrm{H}_{\delta_1}(x)$. Now as $\mathcal{Q}_1$ is output deterministic we have $\{\tilde{y}\} = \mathrm{H}_{\delta_1}(x)$, hence $\tilde{y} = h_1(x) = y$, and therefore $(x, u, y, x') \in \delta_1$. □

Lemma 3.7. *Let $\mathcal{T}_1$ be a TS, $\mathcal{Q}_1$ the SM defined by $\mathcal{T}_1$ via Definition 3.5 and $\mathcal{T}_2$ the TS defined by $\mathcal{Q}_1$ via Definition 3.4. Then $\mathcal{T}_1 = \mathcal{T}_2$.*

Proof. Follows the same lines as the proof of Lemma 3.6 and is therefore omitted. □

Remark 3.1. Conceptually, TS and SM with inputs and outputs coincide with the concept of Moore and Mealy machines, respectively. However, it should be noted that translating a Mealy into a Moore machine usually postpones the output to the next state, e.g., the transition $(x, u, y, x') \in \delta$ results in $y = h(x')$. Observe from Equation 3.3 that our translation from SM to TS results in $y = h(x)$, instead. ◁

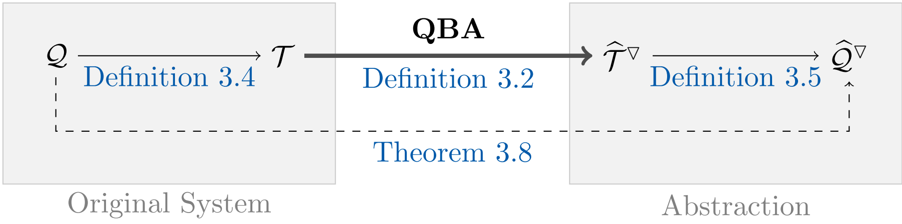

Figure 3.2.: Illustration of the construction of the quotient state machine $\widehat{\mathcal{Q}}^{\triangledown}$ of $\mathcal{Q}$ using QBA.

3.3. State Machine Based Construction

Using the results from the previous section, both the TS $\mathcal{T}$ modeling the "original" system and its quotient TS $\widehat{\mathcal{T}}^{\triangledown}$ can be equivalently represented by output deterministic SM $\mathcal{Q}$ and $\widehat{\mathcal{Q}}^{\triangledown}$ as depicted in Figure 3.2. To reduce the discussion of QBA to SM we introduce a shortcut from $\mathcal{Q}$ to $\widehat{\mathcal{Q}}^{\triangledown}$, as depicted by the dashed line in Figure 3.2.

Theorem 3.8. *Given an output deterministic SM $\mathcal{Q} = (X, U \times Y, \delta, X_0)$, the state machine $\widehat{\mathcal{Q}}^{\triangledown} = (\widehat{X}^{\triangledown}, U \times Y, \widehat{\delta}^{\triangledown}, \widehat{X}_0^{\triangledown})$ constructed from $\mathcal{Q}$ as depicted by the solid path in Figure 3.2 is characterized by*

$$\widehat{X}^{\triangledown} = \{y \in Y | \exists x \in X \; . \; y \in \mathrm{H}_\delta(x)\}\,, \tag{3.5a}$$

$$\widehat{X}_0^{\triangledown} = \{y \in Y | \exists x \in X_0 \; . \; y \in \mathrm{H}_\delta(x)\}\,, \tag{3.5b}$$

$$\widehat{\delta}^{\triangledown} = \left\{ (\widehat{x}, u, y, \widehat{x}') \middle| \exists x, x' \in X \; . \; \begin{pmatrix} \widehat{x} \in \mathrm{H}_\delta(x) \\ \wedge \widehat{x}' \in \mathrm{H}_\delta(x') \\ \wedge (x, u, y, x') \in \delta \end{pmatrix} \right\}. \tag{3.5c}$$

Furthermore, $\widehat{\mathcal{Q}}^{\triangledown}$ is output deterministic and $\mathrm{H}_{\widehat{\delta}^{\triangledown}}(\widehat{x}) = \{\widehat{x}\}$.

Proof. - Observe from Definition 3.4 that the state space of $\mathcal{T}$ is given by X and using (3.3) and the fact that $\mathcal{Q}$ is output deterministic implies

$$h(x) = y \Leftrightarrow y \in \mathrm{H}(x). \tag{3.6}$$

- This immediately implies that (3.1a) and (3.5a) as well as (3.1b) and (3.5b) coincide.
- Now observe that combining (3.1c), (3.1d) and (3.4) implies that

$$\widehat{\delta}^{\triangledown} = \left\{(\widehat{x}, u, \widehat{x}, \widehat{x}') \middle| \exists x, x' \in X \; . \; \left(\widehat{x} = h(x) \wedge \widehat{x}' = h(x) \wedge x' \in f(x, u)\right)\right\}. \tag{3.7}$$

- Using (3.3) and (3.6) in (3.7) yields

$$\widehat{\delta}^{\triangledown} = \left\{(\widehat{x}, u, \widehat{x}, \widehat{x}') \middle| \exists x, x' {\in} X \; . \; \left(\widehat{x} {\in} \mathrm{H}_\delta(x) \wedge \widehat{x}' {\in} \mathrm{H}_\delta(x) \wedge (x, u, \widehat{x}, x') {\in} \delta\right)\right\} \tag{3.8}$$

implying that (3.5c) and (3.8) coincide and $\mathrm{H}_{\widehat{\delta}^{\triangledown}}(\widehat{x}) = \{\widehat{x}\}$. □

4. Modeling the Original System

In Chapter 2 and Chapter 3 it was shown how finite state abstractions can be constructed using SAlCA and QBA if the "original" system evolves over a discrete time axis $\mathbb{N}_0$ and has a finite external signal space W. In this chapter we discuss how those methods can be applied to more general dynamical systems $\breve{\Sigma} = (\breve{T}, \breve{W}, \breve{\mathcal{B}})$ whose signals evolve over a right-unbounded time axis $\breve{T} \subseteq \mathbb{R}_0^+$ taking values in a signal space $\breve{W}$ s.t. $|\breve{W}| = \infty$.

The common starting point of methods generating finite state abstractions of such systems is the definition of a finite set W of symbols. In the literature on SAlCA this set of symbols W is usually assumed to be predefined by the system that is abstracted. I.e., W defines the symbolic interface between the system and the controller, resulting from limited sensing (e.g., a sensor that can only detect threshold crossings) and/or limited actuation (e.g., a valve that can only be fully opened or closed). In contrast the literature on QBA usually assumes full sensing and actuating capabilities but defines a finite set of output symbols Y as the set of equivalence classes of a state space partition. The latter is usually induced by a given specification the subsequently to be designed controller should guarantee. In this case, U is usually defined as a (finite) set of input trajectories guiding the system from one cell of the state space partition to another. Hence, in the setting of QBA the set of symbols W is composed of a finite set of input symbols U and a finite set of output symbols Y.

In both cases, prior to the abstraction process, a model of the external symbolic dynamics of the system $\breve{\Sigma} = (\breve{T}, \breve{W}, \breve{\mathcal{B}})$ w.r.t. W needs to be constructed. In this chapter we will first formalize the intuition outlined above by modeling a dynamical system $\breve{\Sigma} = (\breve{T}, \breve{W}, \breve{\mathcal{B}})$ with finite external symbol set W as a so called ϕ-dynamical system (see [54]) with internal time axis $\breve{T}$ and external time axis $\mathbb{N}_0$ in Section 4.1. By generalizing the asynchronous state property from Definition 2.17 to ϕ-dynamical systems we show in Section 4.2 that, whenever the latter holds, the external symbolic dynamics of a ϕ-dynamical system can be modeled by an ASSDS. As the step by step evolution of an ASSDS can be realized by an SM, this observation results in a unified setting to apply SAlCA and QBA.

4.1. Asynchronous State Space ϕ-Dynamical Systems (ASSϕDS)

We consider a state space dynamical system $\breve{\Sigma}_S = (\breve{T}, \breve{W} \times \breve{X}, \breve{\mathcal{B}}_S)$ whose trajectories evolve over a right-unbounded time axis $\breve{T} \subseteq \mathbb{R}_0^+$ and take values in a dense signal space $\breve{W}$. This system is additionally equipped with a set of external symbols W and a triggering mechanism which relates the time axis $\breve{T}$ of the dynamical system $\breve{\Sigma}_S$ to a sequence of symbols enumerated by a discrete time axis $T = \mathbb{N}_0$. This mechanism results in a time scale transformation from internal time $\breve{T}$ to external time T. Obviously, the sequence of external symbols $\omega \in W^{\mathbb{N}_0}$ is determined by the internal dynamics described by $\breve{\mathcal{B}}_S$ but also by the chosen (or given) triggering mechanism. To handle such models with distinct internal and external time axes but coupled dynamics, we use the notion of ϕ-dynamical systems introduced in [54].

Definition 4.1. *A* ϕ-dynamical system *is a tuple* $\breve{\Sigma}^\phi = (\breve{T}, T, \breve{W}, W, \phi)$*, consisting of internal and external time axes* $\breve{T}$ *and* T*, internal and external signal spaces* $\breve{W}$ *and* W*, and a signal relation* $\phi \subseteq \breve{W}^{\breve{T}} \times W^T \times \mathcal{T}$*, where*[1]

$$\mathcal{T} = \left\{ \theta : \breve{T} \rightharpoonup T \,\middle|\, \theta \text{ is surjective and monotonically increasing} \right\}$$

is the set of suitable time scale transformations.

The inverse time scale transformation[2] $\theta^{-1} : T \to 2^{\breve{T}}$ is then given by

$$\theta^{-1}(k) = \{t \in \breve{T} \mid \theta(t) = k\}. \tag{4.1}$$

Furthermore, the projection of ϕ to $\breve{W}^{\breve{T}}$ and W^T is referred to as the internal and external behavior of $\breve{\Sigma}^\phi$, respectively.

As the original system is modeled by an ASSDS $\breve{\Sigma}_S = (\breve{T}, \breve{W} \times \breve{X}, \breve{\mathcal{B}}_S)$, it seems natural to also include the state space $\breve{X}$ in the third component of $\breve{\Sigma}^\phi$ to obtain a state space ϕ-dynamical system. However, for the term "state space system" to be sound, we need to ensure that the state property is preserved by the relation ϕ. Based on the discussion in Chapter 2, we restrict our attention to the asynchronous state property as defined in Definition 2.17.

Definition 4.2. $\breve{\Sigma}_S^\phi = (\breve{T}, T, \breve{W} \times \breve{X}, W, \phi_S)$ *with* $\phi_S \subseteq (\breve{W} \times \breve{X})^{\breve{T}} \times W^T \times \mathcal{T}$ *is an* asynchronous state space ϕ-dynamical system *(ASSϕDS) if*

$$\begin{aligned}&\forall \left((\breve{\omega}, \breve{\xi}), \omega, \theta\right), \left((\breve{\omega}', \breve{\xi}'), \omega', \theta'\right) \in \phi_S, k, k' \in T, t \in \theta^{-1}(k), t' \in \theta'^{-1}(k') \,. \\ &\quad \left(\breve{\xi}(t) = \breve{\xi}'(t')\right) \Rightarrow \left((\breve{\omega}, \breve{\xi}) \wedge_{t'}^{t} (\breve{\omega}', \breve{\xi}'), \omega \wedge_{k'}^{k} \omega', \theta \wedge_{t'}^{t} \tilde{\theta}'\right) \in \phi_S,\end{aligned} \tag{4.2}$$

[1] Here, $\rightharpoonup$ denotes a partial function.
[2] If θ^{-1} is not set valued, i.e., $\forall k \in T \,.\, |\theta^{-1}(k)| = 1$, by slightly abusing notation, the unique element $t_k \in \theta^{-1}(k)$ is denoted by $\theta^{-1}(k)$ itself and $t_k = \theta^{-1}(k)$ is used.

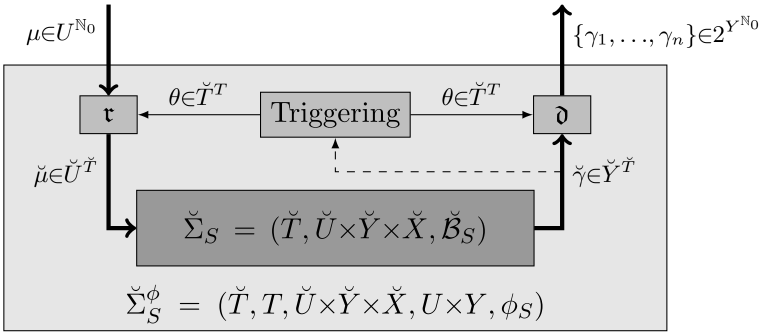

Figure 4.1.: ASSDS $\breve{\Sigma}_S$ considered in Example 4.1-4.3 equipped with a finite set of input symbols U and output symbols Y and a triggering mechanism relating input and output trajectories $(\breve{\mu}, \breve{\gamma})$ to sequences of input and output symbols (μ, γ) via $\mathfrak{r}$ and $\mathfrak{d}$.

where $\forall t \in \breve{T} \,.\, \tilde{\theta}'(t) = \theta'(t) + k - k'$.

Remark 4.1. Observe that when using the synchronous state property (see (2.16) in Section 2.2) in Definition 4.2 it makes a difference if one requires synchronization in the external or the internal time instances, or both. It was therefore suggested in [54, Def. 2] to call $\breve{\Sigma}_S^\phi$ an *externally synchronized* state space ϕ-dynamical system if $k = k'$ in (4.2) and a *synchronized* state space ϕ-dynamical system if $k = k'$ *and* $t = t'$ in (4.2). However, as these concepts will not be used in the remainder of this thesis we restrict our attention to the asynchronous version in Definition 4.2. ◁

To illustrate the intuitive idea of ASSϕDS we consider an ASSDS with inputs and outputs, i.e., $\breve{W} = \breve{U} \times \breve{Y}$ equipped with a finite set of input symbols U and output symbols Y and an external time axis $T = \mathbb{N}_0$, as depicted in Figure 4.1. We discuss three different signal relations ϕ resulting from different triggering mechanisms θ in the following examples.

Example 4.1. Consider the system depicted in Figure 4.1 and assume that the output can only be measured every τ seconds. This results in time-triggered discretization and can be modelled by a time scale transformation θ_a s.t.

$$\forall k \in \mathbb{N}_0 \,.\, \theta_a^{-1}(k) = k\tau, \tag{4.3a}$$

where $\tau \in \breve{T}$ is a fixed time discretization parameter.
Now assume that the sensor measuring $\breve{y}$ can only detect if the current measurement lies in certain bounds. This can be modelled by defining a finite cover $\Lambda = \{A_y \mid y \in Y \wedge A_y \subseteq \breve{Y}\}$ of the output space $\breve{Y}$ and a quantizer $\mathfrak{d} : \breve{Y} \to 2^Y$ s.t.

$$\mathfrak{d}(\breve{y}) = \{y \in Y | \breve{y} \in A_y\}\,. \tag{4.3b}$$

Hence, at each triggering time k, an output symbol $y \in \mathfrak{d}(\breve{\gamma}(k\tau))$ is generated by the sensor which can be used by a controller to issue a symbolic input $u \in U$. Generally, this input can be translated by the map $\mathfrak{r}$ in any input trajectory $\breve{\mu} \in \breve{U}^{[0,\tau)}$. For simplicity we assume that the set of input events U is a finite subset of the input space, i.e., $U \subseteq \breve{U}$, and $\mathfrak{r}$ is realized by a zero-order-hold, i.e.,

$$\mathfrak{r}(u) = \{\breve{\mu} \in \breve{U}^{[0,\tau)} \mid \forall t \in [0,\tau) \,.\, \breve{\mu}(t) = u\}. \tag{4.3c}$$

Using the maps $\mathfrak{d}$ and $\mathfrak{r}$ it is easy to see that we can define ϕ_a such that $((\breve{\mu},\breve{\gamma}),(\mu,\gamma),\theta_a) \in \phi_a$ if

$$\forall k \in \mathbb{N}_0 \,.\, \begin{pmatrix} \gamma(k) \in \mathfrak{d}(\breve{\gamma}(\theta_a^{-1}(k))) \\ \wedge \breve{\mu}|_{[\theta_a^{-1}(k),\theta_a^{-1}(k+1))} \in \mathfrak{r}(\mu(k)) \end{pmatrix}. \tag{4.3d}$$

Observe that in this case θ_a does not need to be included in ϕ as it is given by (4.3a) for all choices of internal and external signals. ◁

Example 4.2. Now we assume that the sensor measuring $\breve{y}$ can only detect threshold-crossings and issues an output event whenever this happens. This results in output event-triggered discretization and can be modeled in many different ways. For simplicity, we assume again the cover Λ and the quantizer $\mathfrak{d}$ from Example 4.1 but additionally assume that each cell A_y is an open set. Then the output event-triggered discretization can be modeled by assuming that an event is triggered whenever a cell A_y is left. This results in the time scale transformation θ_b s.t.

$$\gamma(0) \in \mathfrak{d}(\breve{\gamma}(0)) \quad , \quad \theta_b^{-1}(0) = 0 \quad \text{and} \tag{4.4a}$$

$$\forall k > 0 \,.\, \theta_b^{-1}(k) = \text{glb}\left\{t \geq \theta_b^{-1}(k-1) \middle| \breve{\gamma}(t) \notin \mathfrak{d}^{-1}(\gamma(k-1))\right\}, \tag{4.4b}$$

where glb denotes the greatest lower bound and $\forall y \in Y \,.\, \mathfrak{d}^{-1}(y) = A_y$.

The outlined output event-triggered discretization is illustrated in Figure 4.2 using the cover $\Lambda = \{[0,11),(9,21),(19,31),(29,40]\}$ over $\breve{Y} = [0,40]$ and an arbitrary trajectory $\breve{\gamma} \in \breve{Y}^{\breve{T}}$ (see Figure 4.2 (top)) resulting in a time scale transformation $\theta_b \in T^{\breve{T}}$ (see Figure 4.2 (middle)).

Using the map $\mathfrak{r}$ from Example 4.1 we can furthermore define ϕ_b s.t. $((\breve{\mu},\breve{\gamma}),(\mu,\gamma),\theta_b) \in \phi_b$ if (4.4a) holds and

$$\forall k > 0 \,.\, \begin{pmatrix} \theta_b^{-1}(k) = \text{glb}\left\{t \geq \theta_b^{-1}(k-1) \middle| \breve{\gamma}(t) \notin \mathfrak{d}^{-1}(\gamma(k-1))\right\} \\ \wedge \gamma(k) \in \mathfrak{d}(\breve{\gamma}(\theta_b^{-1}(k))) \\ \wedge \breve{\mu}|_{[\theta_b^{-1}(k),\theta_b^{-1}(k+1))} \in \mathfrak{r}(\mu(k)) \end{pmatrix}. \tag{4.4c}$$

As θ_b depends on $\breve{\gamma}$ in (4.4c), contrary to Example 4.1, θ_b needs to be included in the relation ϕ_b, which motivated its definition in Definition 4.1. Furthermore, observe in

(4.4c) that $\theta_b^{-1}(k+1)$ is unknown when $\breve{\mu}(\theta^{-1}(k))$ needs to be applied. However, zero-order-hold is typically implemented such that the input signal $\breve{\mu}$ is kept constant until an event is triggered, issuing an update of the current value of $\breve{\mu}$, which resolves this issue. $\triangleleft$

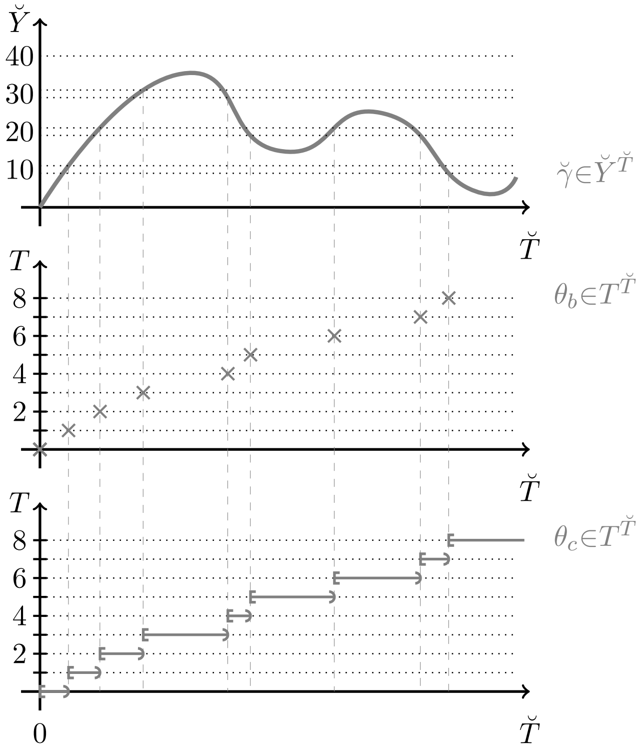

Figure 4.2.: Illustration of point to point (θ_b) and set to point (θ_c) time scale transformations as constructed in Example 4.2 and Example 4.3, respectively, using the cover $\Lambda = \{[0,11),(9,21),(19,31),(29,40]\}$ over $\breve{Y} = [0,40]$.

Remark 4.2. It is important to note that in behavioral models only infinite trajectories are included. Even though this is very reasonable for the internal dynamics of the control system, i.e., for $\breve{\Sigma}_S$, one can easily observe, that output-event triggered discretization of an infinite output signal $\breve{\gamma} \in \breve{Y}^{\breve{T}}$, as discussed in Example 4.2, might result in a finite string of output symbols $\gamma \cdot y$ if the cell A_y is never left by the trajectory $\breve{\gamma}$. Obviously, one could repeat the symbol y infinitely often to generate an infinite sequence of output symbols. However, if one has to know that $\breve{\gamma}$ will never leave A_y to do so, as suggested in [67, Def. 7.2], this generates a non-causal time-triggering mechanism violating Equation 4.2. This can be avoided if the symbol y is repeated after a fixed time

$t_{\mathfrak{d}}$, if $\breve{\gamma}$ has not left A_y during this time period. This would combine event triggered with slow time triggered discretization. ◁

Example 4.3. Recall from Example 4.1 and Example 4.2 that the inverses θ_a^{-1} and θ_b^{-1} are not set-valued, i.e., different points in $\mathrm{dom}(\theta_\mathrm{a})$ and $\mathrm{dom}(\theta_\mathrm{b})$ are mapped to different points in $\mathbb{N}_0$. We therefore call θ_a and θ_b *point to point time scale transformations*. It is easy to see that θ_b can be used to define a *set to point time scale transformation* θ_c s.t.

$$\forall k \in \mathbb{N}_0 \; . \; \theta_c^{-1}(k) = \left[\theta_b^{-1}(k), \theta_b^{-1}(k+1)\right), \tag{4.5}$$

depicted in Figure 4.2 (bottom). Obviously, θ_c^{-1} is set valued and every point in $\breve{T}$ is in the domain of θ_c. While this does not make a difference for the external sequences of symbols (μ, γ) it will have some implications for the similarity relations defined subsequently in Chapter 5. For now it is easy to see that we can construct a signal relation ϕ_c s.t. $((\breve{\mu}, \breve{\gamma}), (\mu, \gamma), \theta_c) \in \phi_c$ if (4.4a), (4.4c) and (4.5) hold. ◁

4.2. External ASSDS of ASSϕDS

Recall from the previous section that Definition 4.2 ensures that the state remains the only past information required to decide on the future evolution of $\breve{\Sigma}_S^\phi$, including the next triggering instant. Moreover, as t and t' are required to be in the domain of θ the future evolution of $\breve{\Sigma}_S^\phi$ is fully determined by states reached at times corresponding to a triggering instant via θ. This set of states is referred to as the *set of externally reachable states*.

Definition 4.3. *Let $\breve{\Sigma}_S^\phi = (\breve{T}, T, \breve{W} \times \breve{X}, W, \phi_S)$ be an ASSϕDS. Then*

$$\breve{X}^{\phi,k} := \left\{ x \in X \middle| \exists \left((\breve{\omega}, \breve{\xi}), (\omega), \theta \right) \in \phi_S, t \in \theta^{-1}(k) \; . \; \breve{\xi}_2(t) = x \right\} \tag{4.6}$$

is the set of states reachable at external time $k \in T$ and $\breve{X}^\phi = \bigcup_{k \in T} \breve{X}^{\phi,k}$ is the set of externally reachable states *of $\breve{\Sigma}_S^\phi$.*

Observe that $\breve{X}^\phi$ is still likely to be infinite as $\breve{X}$ is infinite. If θ is given by the set-to-point time scale transformation constructed in Example 4.3 we even have $\breve{X}^\phi = \breve{X}$. The latter is also true for the special case of a unique time axis, i.e., $\breve{T} = T$, and an identity time scale transformation, i.e., $\theta(t) = t$.

As an important intermediate result we now show that an ASSDS Σ_S modeling the external dynamics of $\breve{\Sigma}_S^\phi$ can be constructed using the externally reachable state space $\breve{X}^\phi$ of $\breve{\Sigma}_S^\phi$ as its state space.

Theorem 4.4. *Let $\breve{\Sigma}_S^\phi = (\breve{T}, T, \breve{W} \times \breve{X}, W, \phi)$ be an ASSϕDS, $X := \breve{X}^\phi$ and*

$$\mathcal{B}_S := \left\{ (\omega, \xi) \middle| \exists ((\breve{\omega}, \breve{\xi}), \omega', \theta) \in \phi \;.\; \begin{pmatrix} \omega = \omega' \\ \wedge \forall k \in T \;.\; \exists t \in \theta^{-1}(k) \;.\; \breve{\xi}(t) = \xi(k) \end{pmatrix} \right\}. \tag{4.7}$$

Then $\Sigma_S = (T, W \times X, \mathcal{B}_S)$ is an asynchronous state space dynamical system.

Proof. To show that (2.25) holds for Σ_S, we pick $(\omega_1, \xi_1), (\omega_2, \xi_2) \in \mathcal{B}_S$ and $k_1, k_2 \in T$ s.t. $\xi(k_1) = \xi(k_2)$ and show that $(\omega, \xi) = (\omega_1, \xi_1) \wedge_{k_2}^{k_1} (\omega_2, \xi_2) \in \mathcal{B}_S$.

- Using (4.7) there exist $(\breve{\omega}_1, \breve{\xi}_1), (\breve{\omega}_2, \breve{\xi}_2) \in \breve{\mathcal{B}}_S$ and $\theta_1, \theta_2 \in \mathcal{T}$ s.t.

  $$((\breve{\omega}_1, \breve{\xi}_1), (\omega_1), \theta_1),\; ((\breve{\omega}_2, \breve{\xi}_2), (\omega_2), \theta_2) \in \phi$$

  and for all $k \in T$

  $$\exists t \in {\theta_1}^{-1}(k) \;.\; \breve{\xi}_1(t) = \xi_1(k) \quad \text{and} \quad \exists t \in {\theta_2}^{-1}(k) \;.\; \breve{\xi}_2(t) = \xi_2(k).$$

- Now we define new timescale transformations $\tilde{\theta}_1 : \breve{T} \to T$ and $\tilde{\theta}_2 : \breve{T} \to T$ s.t. for all $t \in \breve{T}$ and $k \in T$ it holds that

  $$\tilde{\theta}_i(t) = k \Leftrightarrow \left(t \in \theta_i^{-1}(k) \wedge \breve{\xi}_i(t) = \xi_i(k) \right). \tag{4.8}$$

- Now pick $t_1 = \tilde{\theta}_1^{-1}(k_1)$ and $t_2 = \tilde{\theta}_2^{-1}(k_2)$ and observe from (4.8) that $t_1 \in {\theta_1}^{-1}(k_1)$, $t_2 \in {\theta_2}^{-1}(k_2)$ and

  $$\breve{\xi}_1(t_1) = \xi_1(k_1) = \xi_2(k_2) = \breve{\xi}_2(t_2). \tag{4.9}$$

- Defining $\theta = \theta_1 \wedge_{t_2}^{t_1} (\theta_2 + k_1 - k_2)$ and $(\breve{\omega}, \breve{\xi}) = (\breve{\omega}_1, \breve{\xi}_1) \wedge_{t_2}^{t_1} (\breve{\omega}_2, \breve{\xi}_2)$, (4.9) now implies $\left((\breve{\omega}, \breve{\xi}), \omega, \theta\right) \in \phi$ from (4.2) in Definition 4.2.
- It remains to show that $\forall k \in T \;.\; \exists t \in \theta^{-1}(k) \;.\; \breve{\xi}(t) = \xi(k)$.
  - Let $k < k_1$ and pick $t = \tilde{\theta_1}^{-1}(k)$ giving $t < t_1$. Using the construction of the signals θ, $\breve{\xi}$ and ξ via concatenation, (2.1) implies

    $$\theta(t) = \theta_1(t), \quad \breve{\xi}(t) = \breve{\xi}_1(t) \quad \text{and} \quad \xi(k) = \xi_1(k). \tag{4.10}$$

    Now recall from (4.8) that $t \in {\theta_1}^{-1}(k)$ and $\breve{\xi}_1(t) = \xi_1(k)$. Combining the latter with (4.10) therefore yields $t \in \theta^{-1}(k)$ and $\breve{\xi}(t) = \xi(k)$.
  - Let $k \geq k_1$ and pick $t = \tilde{\theta_2}^{-1}(k - k_1 + k_2) + t_1 - t_2$ giving $t \geq t_1$. Using the construction of the signals θ, $\breve{\xi}$ and ξ via concatenation, (2.1) implies

    $$\theta(t) = \theta_2(t - t_1 + t_2) + k_1 - k_2,\; \breve{\xi}(t) = \breve{\xi}_2(t - t_1 + t_2) \text{ and } \xi(k) = \xi_2(k - k_1 + k_2). \tag{4.11}$$

    Now recall from (4.8) that $t - t_1 + t_2 \in {\theta_2}^{-1}(k - k_1 + k_2)$ and $\breve{\xi}_2(t - t_1 + t_2) = \xi_2(k - k_1 + k_2)$. Combining the latter with (4.11) therefore yields $\theta(t) = k$ (hence $t \in \theta^{-1}(k)$) and $\breve{\xi}(t) = \xi(k)$.

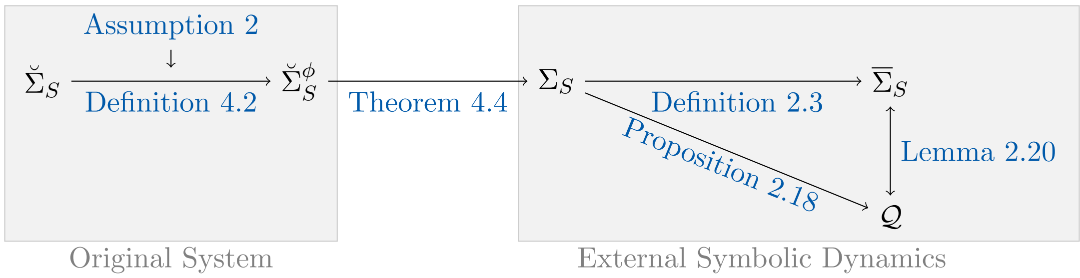

Figure 4.3.: Connection between the original system $\breve{\Sigma}_S$ and the state machine $\mathcal{Q}$ modeling the step by step evolution of its external symbolic dynamics under Assumption 2.

□

Definition 4.5. *Given the premises of Theorem 4.4, $\Sigma_S = (T, W \times X, \mathcal{B}_S)$ is the external ASSDS of $\breve{\Sigma}_S^{\phi}$.*

Summarizing the preceding discussion, the external symbolic dynamics of $\breve{\Sigma}_S$ can be modeled by an ASSDS Σ_S if the following assumption holds.

Assumption 2. *The internal dynamics of the original system can be modeled by an ASSDS $\breve{\Sigma}_S = (\breve{T}, \breve{W} \times \breve{X}, \breve{\mathcal{B}}_S)$ equipped with a finite set of external symbols W and a signal map ϕ_S s.t. (4.2) holds.*

The chain of constructions to obtain Σ_S from $\breve{\Sigma}_S$ under Assumption 2 are summarized in Figure 4.3. This picture is completed by recalling from Proposition 2.28 and Lemma 2.20, that Σ_S induces an SM $\mathcal{Q}$ modeling its step-by-step evolution and realizing its ω-closure $\overline{\Sigma}_S$. We will see in Chapter 6 and Chapter 7, that the latter observation allows us to consider a unified setting for SAlCA and QBA resulting in a formal comparison of the resulting abstract state machines in Chapter 8.

5. Simulation Relations

The definition of SAlCA was motivated in Chapter 2 by the fact that the behavior $\widehat{\mathcal{B}}^{l\uparrow}$ resulting from the simplified domino game played with $l+1$ long pieces of the symbolic behavior $\mathcal{B} \subseteq W^{\mathbb{N}_0}$ is an overapproximation of the latter, hence $\mathcal{B} \subseteq \widehat{\mathcal{B}}^{l\uparrow}$. It was furthermore shown that equality holds iff $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$ is asynchronously l-complete. Hence, the construction of SAlCA and its relation to the system Σ is based on inclusion (or equality) of the external behavior $\mathcal{B} \subseteq W^{\mathbb{N}_0}$.

Contrary, in the literature on QBA usually the stronger properties of similarity and bisimilarity are used to relate the abstraction to the original model. These properties are formalized by constructing so called simulation and bisimulation relations between the state spaces of two models in a step-by-step fashion, ensuring that trajectories visiting only related states at every time point produce the same *output* trajectory. Hence, the existence of a simulation or bisimulation relation implies, respectively, behavioral inclusion and equality of the *output* part of the external behavior, i.e., $\mathcal{B}_Y \subseteq Y^{\mathbb{N}_0}$. However, the inverse implication does usually not hold.

In the spirit of our bridging endeavor it is therefore interesting to investigate if those stronger properties do also hold between the original system and a state space realization of its SAlCA w.r.t. the full external behavior $\mathcal{B} \subseteq W^{\mathbb{N}_0}$. Motivated by the fact that the original system is assumed to be modeled by an ASSϕDS, we first formalize similarity and bisimilarity for the general class of ASSϕDS in Section 5.1. To consider matching of external sequences w.r.t. different sets of symbols, we furthermore use the set

$$V \quad s.t. \quad \pi_V(W) \neq \emptyset \tag{5.1}$$

to define the considered signal space (i.e., $V = W$ for SAlCA and $V = Y$ for QBA with $W = U \times Y$ from Assumption 1).

As the SAlCA is usually given by an ASSDS rather than an ASSϕDS we also need a notion of simulation relations for other system models. We will show in Section 5.2 that ASSϕDS can also be used to model ASSDS and SM. Hence, the notion of simulation relations for ASSϕDS can be directly transfered to ASSDS and SM. Additionally to the formal beauty of this result, using this general approach allows to relate different system models (e.g., an SM $\mathcal{Q}$ to an ASSϕDS $\breve{\Sigma}_S^{\phi}$).

In Section 5.3 the chapter is concluded by a discussion of related notions of similarity which have appeared previously.

5.1. Simulation Relations for ASSϕDS

Asynchronous state space ϕ-dynamical systems (ASSϕDS) as introduced in Definition 4.2 are generally not assumed to be complete. Inspired by [26, Def. 5.21], simulation relations for this system class will therefore be defined in a concatenation based fashion in this section.

Definition 5.1. *Let $\breve{\Sigma}^{\phi}_{S,i} = (\breve{T}_i, T, \breve{W}_i \times \breve{X}_i, W_i, \phi_{S,i})$, $i \in \{1,2\}$ be ASSϕDS and V a set s.t. $\pi_V(W_1) = \pi_V(W_2) \neq \emptyset$. Then a relation $\mathcal{R} \subseteq \breve{X}_1 \times \breve{X}_2$ is a* simulation relation *from $\breve{\Sigma}^{\phi}_{S,1}$ to $\breve{\Sigma}^{\phi}_{S,2}$ w.r.t. V (written $\mathcal{R} \in \mathfrak{S}_V(\breve{\Sigma}^{\phi}_{S,1}, \breve{\Sigma}^{\phi}_{S,2})$) if*

$$\forall \breve{x}_1 \in \breve{X}_1^{\phi,0} \,.\, \left(\exists \breve{x}_2 \in \breve{X}_2^{\phi,0} \,.\, (\breve{x}_1, \breve{x}_2) \in \mathcal{R}\right) \quad \textit{and} \tag{5.2a}$$

$$\begin{aligned}
&\forall\, (\breve{x}_1, \breve{x}_2) \in \mathcal{R}, \left((\breve{\omega}_1, \breve{\xi}_1), \omega_1, \theta_1\right) \in \phi_{S,1}, k_1 \in \mathbb{N}_0, t_1 \in {\theta_1}^{-1}(k_1) \,.\\
&\quad \left(\breve{\xi}_1(t_1) = \breve{x}_1\right) \;\Rightarrow\\
&\qquad \exists\, \left((\breve{\omega}_2, \breve{\xi}_2), \omega_2, \theta_2\right) \in \phi_{S,2}, k_2 \in \mathbb{N}_0, t_2 \in {\theta_2}^{-1}(k_2) \,.\\
&\qquad\quad \left(\begin{array}{l}
\breve{\xi}_2(t_2) = \breve{x}_2\\
\wedge \forall k \geq k_1 \,.\, \pi_V(\omega_1)(k) = \pi_V(\omega_2)(k - k_1 + k_2)\\
\wedge \left(\begin{array}{l}
\forall\, k \geq k_1, t_1' \in {\theta_1}^{-1}(k), t_1' \geq t_1 \,.\\
\quad \exists t_2' \in {\theta_2}^{-1}(k - k_1 + k_2) \,.\, (\breve{\xi}_1(t_1'), \breve{\xi}_2(t_2')) \in \mathcal{R}
\end{array}\right)
\end{array}\right).
\end{aligned} \tag{5.2b}$$

Furthermore, $\mathcal{R} \subseteq \breve{X}_1 \times \breve{X}_2$ is a bisimulation relation *between $\breve{\Sigma}^{\phi}_{S,1}$ and $\breve{\Sigma}^{\phi}_{S,2}$ w.r.t. V (written $\mathcal{R} \in \mathfrak{B}_V(\breve{\Sigma}^{\phi}_{S,1}, \breve{\Sigma}^{\phi}_{S,2})$) if $\mathcal{R} \in \mathfrak{S}_V(\breve{\Sigma}^{\phi}_{S,1}, \breve{\Sigma}^{\phi}_{S,2})$ and[1] $\mathcal{R}^{-1} \in \mathfrak{S}_V(\breve{\Sigma}^{\phi}_{S,2}, \breve{\Sigma}^{\phi}_{S,1})$.*

Remark 5.1. Recall from Definition 4.2 that this thesis defines a state space ϕ-dynamical system as a five-tuple with signal relation $\phi_S \subseteq (\breve{W} \times \breve{X})^{\breve{T}} \times W^T \times \mathcal{T}$ instead of a seven tuple with signal map $\phi_S : \breve{\mathcal{B}} \to 2^{W^T \times \mathcal{T}}$ as in [54, Def. 1]. Additionally, Definition 5.1 differs from the definition of an *asynchronous simulation relation* in [54, Def. 4] in two ways. (i) Definition 5.1 uses a simpler implication structure in (5.2b) to simplify subsequent proofs. However, since the state property holds for $\breve{\Sigma}^{\phi}_{S}$, it can be shown that (5.2b) and (6b) in [54, Def. 4] coincide. (ii) By requiring the relation of initial states in (5.2a) and given point (i), the simulation relation defined in Definition 5.1 actually coincides with the so called 0-initial simulation relation defined in [54, Def. 6]. The existence of the latter is shown to imply the existence of an asynchronous simulation relation in [54, Lem. 1] and is therefore slightly stronger. However, as we only use this type of simulation relation in the remaining part of this thesis, we restrict our attention to this notion. $\triangleleft$

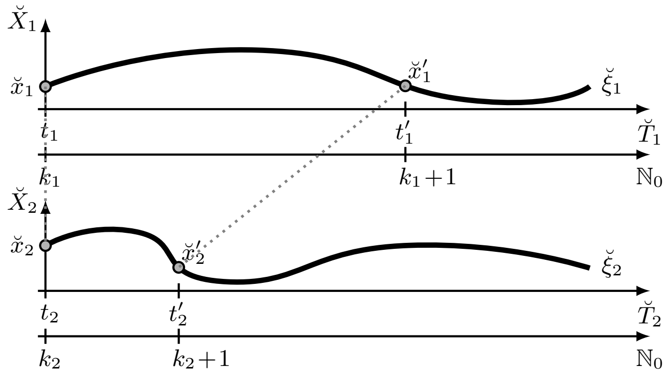

Figure 5.1.: Visualization of the last line in (5.2b) for two point to point time scale transformations θ_1 and θ_2, with ${\theta_1}^{-1}(k_1) = \{t_1\}$, ${\theta_1}^{-1}(k_1+1) = \{t'_1\}$, ${\theta_2}^{-1}(k_2) = \{t_2\}$ and ${\theta_2}^{-1}(k_2+1) = \{t'_2\}$. Dotted lines connect related states.

To generate some intuition for the simulation relations constructed in Definition 5.1, we will discuss (5.2b) using some graphical illustrations. For this purpose assume that we have signals $\big((\breve{\omega}_1, \breve{\xi}_1), \omega_1, \theta_1\big) \in \phi_{S,1}$, and $\big((\breve{\omega}_2, \breve{\xi}_2), \omega_2, \theta_2\big) \in \phi_{S,2}$ such that the states $\breve{x}_1 = \breve{\xi}_1(t_1)$ and $\breve{x}_2 = \breve{\xi}_2(t_2)$, with $k_1 = \theta_1(t_1)$ and $k_2 = \theta_2(t_2)$, are related. To simulate $\breve{\Sigma}^{\phi}_{S,1}$, the system $\breve{\Sigma}^{\phi}_{S,2}$ must exhibit two properties, namely (i) the signal $\nu_2 = \pi_V(\omega_2)$ generated by $\breve{\Sigma}^{\phi}_{S,2}$ after k_2 must be identical to the external signal $\nu_1 = \pi_V(\omega_1)$ generated by $\breve{\Sigma}^{\phi}_{S,1}$ after k_1 and (ii) the state trajectories $\breve{\xi}_1$ and $\breve{\xi}_2$ need to stay related for all future external time instants.

Concerning the second property, the nature of θ significantly influences how restrictive this requirement is. For example, having a point to point time scale transformation in both systems only requires state trajectories to be related at sampling points (Figure 5.1), while a set to point time scale transformation, for example, requires state trajectories to be related at all future times (Figure 5.2). However, as clearly visible in Figure 5.1 and 5.2, both cases allow for a stretching or shrinking of time between related state trajectories. If both systems have an identity time scale transformation (and therefore $\breve{T} = \breve{T}_1 = \breve{T}_2 = T$) this stretching or shrinking of time is no longer allowed, as shown in Figure 5.3.

A Partial Order for ASSϕDS

Using Definition 5.1 we can formally define an ordering on the set of ASSϕDS in the usual way and show that simulation and bisimulation relations are, respectively, preorders and equivalence relations for the set of ASSϕDS.

[1]As usual, $\mathcal{R}^{-1} := \{(\breve{x}_2, \breve{x}_1) \mid (\breve{x}_1, \breve{x}_2) \in \mathcal{R}\}$.

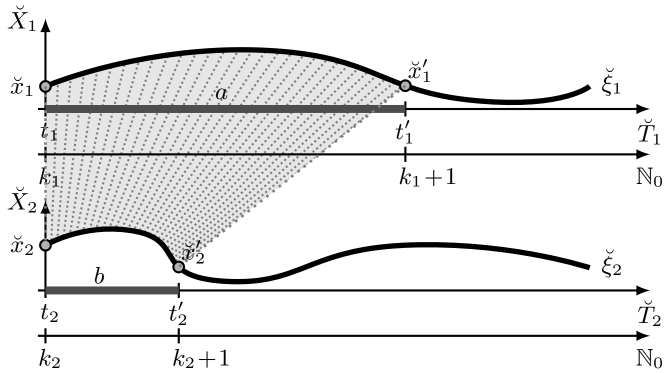

Figure 5.2.: Visualization of the last line in (5.2b) for two set to point time scale transformations θ_1 and θ_2, with $a = {\theta_1}^{-1}(k_1) = [t_1, t'_1)$ and $b = {\theta_2}^{-1}(k_2) = [t_2, t'_2)$. Dotted lines connect related states.

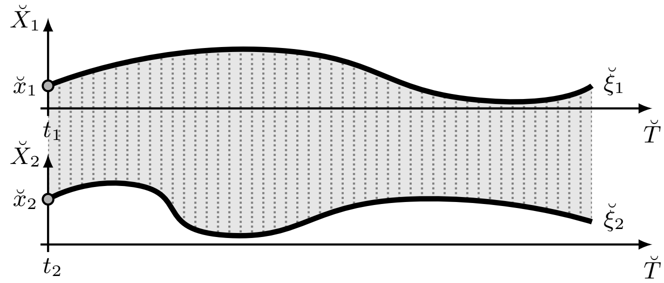

Figure 5.3.: Visualization of the last line in (5.2b) for two identity time scale transformations, i.e., $\forall\, t \in \breve{T}\ .\ t = \theta_1(t) = \theta_2(t)$.

Definition 5.2. *Given the premises of Definition 5.1, $\breve{\Sigma}^{\phi}_{S,1}$ is* simulated *by $\breve{\Sigma}^{\phi}_{S,2}$ w.r.t. V (written $\breve{\Sigma}^{\phi}_{S,1} \preceq_V \breve{\Sigma}^{\phi}_{S,2}$) if there exists a relation $\mathcal{R} \in \mathfrak{S}_V(\breve{\Sigma}^{\phi}_{S,1}, \breve{\Sigma}^{\phi}_{S,2})$ and $\breve{\Sigma}^{\phi}_{S,1}$ and $\breve{\Sigma}^{\phi}_{S,2}$ are* bisimilar *w.r.t. V (written $\breve{\Sigma}^{\phi}_{S,1} \cong_V \breve{\Sigma}^{\phi}_{S,2}$) if there exists a relation $\mathcal{R} \in \mathfrak{B}_V(\breve{\Sigma}^{\phi}_{S,1}, \breve{\Sigma}^{\phi}_{S,2})$.*

Theorem 5.3. *The relations $\preceq_V$ and $\cong_V$ are a preorder and an equivalence relation for the set of ASSϕDS, respectively.*

Proof. To simplify notation we denote the conjunction on the right hand side of (5.2b) by Ω, i.e.,

$$\Omega_{ab} := \begin{pmatrix} \breve{\xi}_b(t_b) = \breve{x}_b \\ \wedge \forall k \geq k_a\ .\ \pi_V(\omega_a)(k) = \pi_V(\omega_2)(k - k_a + k_b) \\ \wedge \begin{array}{l} \forall\, k \geq k_a, t'_a \in {\theta_a}^{-1}(k), t'_a \geq t_a\ . \\ \quad \exists t'_b \in {\theta_b}^{-1}(k - k_a + k_b)\ .\ (\breve{\xi}_a(t'_a), \breve{\xi}_b(t'_b)) \in \mathcal{R} \end{array} \end{pmatrix}.$$

We first show that $\preceq_V$ is a preorder, i.e., $\preceq_V$ is reflexive and transitive. To prove *reflexivity*, we pick an arbitrary $\breve{\Sigma}_S^\phi$, construct

$$\mathcal{R} = \left\{(\breve{x}_1, \breve{x}_2) \in \breve{X} \times \breve{X} \middle| \breve{x}_1 = \breve{x}_2\right\}$$

and show that (5.2) holds. First observe that (5.2a) holds by construction. To see that this is also true for (5.2b), observe that whenever there exists

$$\left((\breve{\omega}_1, \breve{\xi}_1), \omega_1, \theta_1\right) \in \phi, k_1 \in \mathbb{N}_0, t_1 \in {\theta_1}^{-1}(k_1) \text{ s.t. } \left(\breve{\xi}_1(t_1) = \breve{x}_1\right)$$

for some $(\breve{x}_1, \breve{x}_2) \in \mathcal{R}$ (i.e., $\breve{x}_1 = \breve{x}_2$) we can pick

$$\left((\breve{\omega}_2, \breve{\xi}_2), \omega_2, \theta_2\right) = \left((\breve{\omega}_1, \breve{\xi}_1), \omega_1, \theta_1\right),\ k_2 = k_1 \text{ and } t_2 = t_2$$

to make (5.2b) hold for any V with $\pi_V(W) \neq \emptyset$.

To prove *transitivity*, we pick arbitrary $\breve{\Sigma}_{S,1}^\phi, \breve{\Sigma}_{S,2}^\phi, \breve{\Sigma}_{S,3}^\phi$ and V s.t. $\pi_V(W_1) = \pi_V(W_2) = \pi_V(W_3) \neq \emptyset$ and $\breve{\Sigma}_{S,1}^\phi \preceq_V \breve{\Sigma}_{S,2}^\phi$ and $\breve{\Sigma}_{S,2}^\phi \preceq_V \breve{\Sigma}_{S,3}^\phi$, i.e., there exist simulation relations $\mathcal{R}_{12} \in \mathfrak{S}_V(\breve{\Sigma}_{S,1}^\phi, \breve{\Sigma}_{S,2}^\phi)$ and $\mathcal{R}_{23} \in \mathfrak{S}_V(\breve{\Sigma}_{S,2}^\phi, \breve{\Sigma}_{S,3}^\phi)$. Now we define

$$\mathcal{R}_{13} = \mathcal{R}_{12} \circ \mathcal{R}_{23} := \left\{(\breve{x}_1, \breve{x}_3) \in \breve{X}_1 \times \breve{X}_3 \middle| \exists \breve{x}_2 \in \breve{X}_2 \ .\ \begin{pmatrix} (\breve{x}_1, \breve{x}_2) \in \mathcal{R}_{12} \\ \wedge (\breve{x}_2, \breve{x}_3) \in \mathcal{R}_{23} \end{pmatrix}\right\} \tag{5.3}$$

and show that (5.2) holds for $\mathcal{R}_{13}$ and V, implying $\breve{\Sigma}_{S,1}^\phi \preceq_V \breve{\Sigma}_{S,3}^\phi$.

- Observe that (5.2a) holds for $\mathcal{R}_{12}$ and $\mathcal{R}_{23}$, implying

$$\forall \breve{x}_1 \in \breve{X}_{10} \ .\ \left(\exists \breve{x}_2 \in \breve{X}_{20}, \breve{x}_3 \in \breve{X}_{30} \ .\ (\breve{x}_1, \breve{x}_2) \in \mathcal{R}_{12} \wedge (\breve{x}_2, \breve{x}_3) \in \mathcal{R}_{23}\right) \tag{5.4}$$

  and therefore (using (5.3)) (5.2a) holds for $\mathcal{R}_{13}$.
- To show (5.2b), we fix

$$(\breve{x}_1, \breve{x}_3) \in \mathcal{R}_{13},\ \left((\breve{\omega}_1, \breve{\xi}_1), \omega_1, \theta_1\right) \in \phi_1,\ k_1 \in \mathbb{N}_0,\ t_1 \in {\theta_1}^{-1}(k_1)\ \ s.t.\ \breve{\xi}_1(t_1) = \breve{x}_1$$

  and show that Ω_{13} holds.
  · Using (5.4) we know that there exist $\breve{x}_2 \in \breve{X}_2$ and $\breve{x}_3 \in \breve{X}_3$ s.t. $(\breve{x}_1, \breve{x}_2) \in \mathcal{R}_{12}$ and $(\breve{x}_2, \breve{x}_3) \in \mathcal{R}_{23}$. As $\mathcal{R}_{12}, \mathcal{R}_{23}$ are simulation relations, we know that (5.2b) holds. This implies that we can fix

$$\left((\breve{\omega}_2, \breve{\xi}_2), \omega_2, \theta_2\right) \in \phi_2,\ k_2 \in \mathbb{N}_0,\ t_2 \in {\theta_2}^{-1}(k_2) \text{ s.t. } \Omega_{12} \text{ holds and}$$
$$\left((\breve{\omega}_3, \breve{\xi}_3), \omega_3, \theta_3\right) \in \phi_3,\ k_3 \in \mathbb{N}_0,\ t_3 \in {\theta_3}^{-1}(k_3) \text{ s.t. } \Omega_{23} \text{ holds.}$$

- Observe, that the first lines of Ω_{13} and Ω_{23} are equivalent.
- Using the second line of Ω_{12} and Ω_{23} we get

$$\begin{aligned}\pi_V(\omega_1)(k) &= \pi_V(\omega_2)(k - k_1 + k_2)\\ &= \pi_V(\omega_3)((k - k_1 + k_2) - k_2 + k_3) = \pi_V(\omega_3)(k - k_1 + k_3)\end{aligned}$$

  implying that the second line of Ω_{13} also holds.
- The third lines of Ω_{12} and Ω_{23} imply

$$\forall\, k \geq k_1, t_1' \in {\theta_1}^{-1}(k), t_1' \geq t_1 \,.\, \begin{pmatrix} \exists t_2' \in {\theta_2}^{-1}(k - k_1 + k_2) \,.\, (\breve{\xi}_1(t_1'), \breve{\xi}_2(t_2')) \in \mathcal{R}_{12}\\ \wedge \exists t_3' \in {\theta_3}^{-1}((k - k_1 + k_2) - k_2 + k_3) \,.\, (\breve{\xi}_2(t_2'), \breve{\xi}_3(t_3')) \in \mathcal{R}_{23} \end{pmatrix}$$

  Using (5.3) this implies that the last line of Ω_{13} holds.

Now we show that the relation $\cong_V$ is an equivalence relation, i.e., $\cong_V$ is reflexive, transitive and symmetric. From Definition 5.2 and Definition 5.1 it follows that the relation $\cong_V$ is defined by two simulation relations. Therefore reflexivity and transitivity follows from the first part of this proof. To prove *symmetry*, we pick arbitrary $\breve{\Sigma}^{\phi}_{S,1}, \breve{\Sigma}^{\phi}_{S,2}$ and V s.t. $\pi_V(W_1) = \pi_V(W_2) \neq \emptyset$ and show

$$\left(\breve{\Sigma}^{\phi}_{S,1} \cong_V \breve{\Sigma}^{\phi}_{S,2}\right) \Rightarrow \left(\breve{\Sigma}^{\phi}_{S,2} \cong_V \breve{\Sigma}^{\phi}_{S,1}\right).$$

It follows immediately from Definition 5.2 that for any bisimulation relation $\mathcal{R}$ between $\breve{\Sigma}^{\phi}_{S,1}$ and $\breve{\Sigma}^{\phi}_{S,2}$ w.r.t. V we can pick $\tilde{\mathcal{R}} = \mathcal{R}^{-1}$ as a bisimulation relation between $\breve{\Sigma}^{\phi}_{S,2}$ and $\breve{\Sigma}^{\phi}_{S,1}$ w.r.t. V, implying $\breve{\Sigma}^{\phi}_{S,2} \cong_V \breve{\Sigma}^{\phi}_{S,1}$. □

Behavioral Inclusion

As related states generate the same external events (see (5.2b)) the existence of a simulation relation from one system to another one implies that the external behavior of the first is a subset of the second one. As an immediate consequence, external behavioral equivalence is obtained if two systems are bisimilar. This is formalized in Proposition 5.4 which generalizes [26, Thm. 5.41] to ASSϕDS.

Proposition 5.4. *Given the premises of Definition 5.1 and $\mathcal{B}_i = \pi_V(\phi_{S,i}),\ i \in \{1, 2\}$, it holds that*

$$\left(\breve{\Sigma}^{\phi}_{S,1} \preceq_V \breve{\Sigma}^{\phi}_{S,2}\right) \Rightarrow (\mathcal{B}_1 \subseteq \mathcal{B}_2) \quad \textit{and} \quad \left(\breve{\Sigma}^{\phi}_{S,1} \cong_V \breve{\Sigma}^{\phi}_{S,2}\right) \Rightarrow (\mathcal{B}_1 = \mathcal{B}_2).$$

Proof. Observe, that $\mathcal{B}_1 \subseteq \mathcal{B}_2$ iff

$$\forall \Big((\breve{\omega}_1, \breve{\xi}_1), \omega_1, \theta_1\Big) \in \phi_{S,1} \;.\; \exists \Big((\breve{\omega}_2, \breve{\xi}_2), \omega_2, \theta_2\Big) \in \phi_{S,2} \;.\; \pi_V(\omega_1) = \pi_V(\omega_2)$$

Now fix $\Big((\breve{\omega}_1, \breve{\xi}_1), \omega_1, \theta_1\Big) \in \phi_{S,1}$. Since $\Sigma_{S,1} \preceq_V \Sigma_{S,2}$, (5.2a) and (5.2b) hold for $k = 0$. Using (4.6), we can pick $t_1 \in {\theta_1}^{-1}(0)$ and $\Big((\breve{\omega}_2, \breve{\xi}_2), \omega_2, \theta_2\Big) \in \phi_{S,2}, t_2 \in {\theta_2}^{-1}(0)$ s.t.

$$\forall k \geq 0 \;.\; \pi_V(\omega_1)(k) = \pi_V(\omega_2)(k).$$

As $T \subseteq \mathbb{R}_0^+$, this implies $\pi_V(\omega_1) = \pi_V(\omega_2)$ what proves the first statement. The second statement follows immediately from the first and the definition of the relation $\cong_V$ in Definition 5.1 and Definition 5.2. □

5.2. Using ASSϕDS as a Unified Model

In Chapter 4, ASSϕDS were introduced as a way to model systems where (possibly continuous) internal dynamics are related to external event sequences using a signal map ϕ. While this is the main usage of ASSϕDS in this thesis we will show in this section that ASSϕDS can also serve as a common modeling framework for other state space models used in this thesis, namely ASSDS and SM. This allows to define relations among them with the understanding that their respective ASSϕDS are related via Definition 5.1.

$\phi_{\mathbf{i}}$-Extensions of ASSDS and SM

We start the formalization of the above intuition by rewriting a given ASSDS into an ASSϕDS. This can be done by introducing an identity signal map which does neither change the time axis nor state or event trajectories, resulting in the following ASSϕDS.

Definition 5.5. *Let $\Sigma_S = (T, W \times X, \mathcal{B}_S)$ be an ASSDS and*

$$\phi_{\mathbf{i}}(\mathcal{B}_S) = \left\{ ((\xi, \omega), \omega', \theta_{\mathbf{i}}) \in \mathcal{B}_S \times W^T \times \mathcal{T} \;\middle|\; \begin{pmatrix} \omega = \omega' \\ \wedge \forall k \in T \;.\; \theta_{\mathbf{i}}(k) = k \end{pmatrix} \right\} \tag{5.5}$$

the identity signal relation. Then $[\Sigma_S]^{\phi_{\mathbf{i}}} := (T, T, W \times X, W, \phi_{\mathbf{i}}(\mathcal{B}_S))$ is the $\phi_{\mathbf{i}}$-extension *of Σ_S.*

Lemma 5.6. *Let Σ_S be an ASSDS and $[\Sigma_S]^{\phi_{\mathbf{i}}}$ its $\phi_{\mathbf{i}}$-extension. Then*
(i) $[\Sigma_S]^{\phi_{\mathbf{i}}}$ is an ASSϕDS and
(ii) Σ_S is the external ASSDS of $[\Sigma_S]^{\phi_{\mathbf{i}}}$ defined via Definition 4.5.

Proof. We show both statements separately.

(i) We have to show that (4.2) holds for $\phi_{\mathbf{i}}(\mathcal{B}_S)$. Hence, pick

$$((\omega,\xi),\omega,\theta_{\mathbf{i}}),\left((\omega',\xi'),\omega',\theta_{\mathbf{i}}\right)\in\phi_{\mathbf{i}}(\mathcal{B}_S),k,k'\in T\quad s.t.\quad \xi(k)=\xi'(k').$$

Now observe that $(\tilde{\omega},\tilde{\xi})=(\omega,\xi)\wedge_{k'}^{k}(\omega',\xi')\in\mathcal{B}_S$ as Σ_S is an ASSDS and obviously $\tilde{\omega}=\omega\wedge_{k'}^{k}\omega'$. Furthermore, observe that

$$\tilde{\theta}=\theta_{\mathbf{i}}\wedge_{k'}^{k}\left(\theta_{\mathbf{i}}+k-k'\right)=\theta_{\mathbf{i}},$$

implying that $\left((\tilde{\omega},\tilde{\xi}),\tilde{\omega},\tilde{\theta}\right)\in\phi_{\mathbf{i}}(\mathcal{B}_S)$, what proves the statement.

(ii) It can be easily observed from (5.5) that (4.7) holds by definition of $\phi_{S,\mathbf{i}}$. Furthermore, as $\breve{T}=T$ (4.6) implies $X^{\phi}=X$. With this the statement follows immediately from Definition 4.5.

□

Lemma 5.6 implies that we can use an ASSDS and its $\phi_{\mathbf{i}}$-extension interchangeably to model the same dynamics, as depicted in Figure 5.4 (middle). Similarly, it was shown in Section 2.6, that ASSDS and SM can be used interchangeably whenever the former is complete and the latter is live and reachable. Using this assumption, the $\phi_{\mathbf{i}}$-extension $[\mathcal{Q}]^{\phi_{\mathbf{i}}}$ of a live and reachable SM $\mathcal{Q}$ can be constructed via its induced ASSDS as depicted in Figure 5.4 (bottom). Obviously, we can formally also define the $\phi_{\mathbf{i}}$-extension $\left[\breve{\Sigma}_S^{\phi}\right]^{\phi_{\mathbf{i}}}$ of an ASSϕDS $\breve{\Sigma}_S^{\phi}$, as depicted in Figure 5.4 (top), evaluating directly to $\breve{\Sigma}_S^{\phi}$.

Hence, given a model[2] $\mathcal{M}\in(\text{ASS}\phi\text{DS}\cup\text{ASSDS}\cup\text{SM}^{\star})$, without loss of generality, we can use $\mathcal{M}$ and $[\mathcal{M}]^{\phi_{\mathbf{i}}}$ interchangeably. This leads to the following definition of simulation relations for these models via Definition 5.1.

Definition 5.7. *Given two state space models $\mathcal{M}_i\in(\text{ASS}\phi\text{DS}\cup\text{ASSDS}\cup\text{SM}^{\star})$, $i\in\{1,2\}$ with state spaces X_i and external signal spaces W_i, let $[\mathcal{M}_i]^{\phi_{\mathbf{i}}}$ denote their $\phi_{\mathbf{i}}$-*extensions *constructed from $\mathcal{M}_i$ as depicted in Figure 5.4 and V a set s.t. $\pi_V(W_1)=\pi_V(W_2)\neq\emptyset$.*

Then a relation $\mathcal{R}\subseteq X_1\times X_2$ is a simulation relation *from $\mathcal{M}_1$ to $\mathcal{M}_2$ w.r.t. V (written $\mathcal{R}\in\mathfrak{S}_V(\mathcal{M}_1,\mathcal{M}_2)$) if $\mathcal{R}\in\mathfrak{S}_V([\mathcal{M}_1]^{\phi_{\mathbf{i}}},[\mathcal{M}_2]^{\phi_{\mathbf{i}}})$. Analogously, $\mathcal{R}\subseteq X_1\times X_2$ is a* bisimulation relation *between $\mathcal{M}_1$ and $\mathcal{M}_2$ w.r.t. V (written $\mathcal{R}\in\mathfrak{B}_V(\mathcal{M}_1,\mathcal{M}_2)$) if $\mathcal{R}\in\mathfrak{B}_V([\mathcal{M}_1]^{\phi_{\mathbf{i}}},[\mathcal{M}_2]^{\phi_{\mathbf{i}}})$.*

Simulation Relations for live and reachable SM

Recall from Section 5.1 that simulation relations are defined in a concatenation-based fashion in Definition 5.1 as their external behavior might not be complete. However,

[2]Slightly abusing notation we associate the abbreviations for each model with the set of all models in this class and denote by $\text{SM}^{\star}$ the class of all live and reachable SM.

$$\left[\breve{\Sigma}_S^{\phi}\right]^{\phi_{\mathbf{i}}} := \breve{\Sigma}_S^{\phi}$$

$$\begin{tikzcd}
\Sigma_S \arrow[r, bend left, "\text{Definition 5.5}"] \arrow[r, phantom, "\text{Lemma 5.6}"] & \left[\Sigma_S\right]^{\phi_{\mathbf{i}}} \arrow[l, bend left, "\text{Definition 4.5}"]
\end{tikzcd}$$

$$\begin{tikzcd}
\mathcal{Q} \arrow[r, bend left, "\text{Proposition 2.19}"] \arrow[r, phantom, "\text{Lemma 2.21}"] & \overline{\Sigma}_S \arrow[l, bend left, "\text{Proposition 2.18}"] \arrow[r, bend left, "\text{Definition 5.5}"] \arrow[r, phantom, "\text{Lemma 5.6}"] & \left[\overline{\Sigma}_S\right]^{\phi_{\mathbf{i}}} \arrow[l, bend left, "\text{Definition 4.5}"] =: [\mathcal{Q}]^{\phi_{\mathbf{i}}}
\end{tikzcd}$$

Figure 5.4.: Constructing the $\phi_{\mathbf{i}}$-extensions of an ASSϕDS $\breve{\Sigma}_S^{\phi}$, an ASSDS Σ_S and an SM* (i.e., a live and reachable SM) $\mathcal{Q}$ (top to bottom).

SM where shown in Lemma 2.20 to only realize complete dynamical systems. Hence, the $\phi_{\mathbf{i}}$-extension of an SM is always complete. This allows to simplify the definition of simulations relations for SM via Definition 5.7 and Definition 5.1 as follows.

Theorem 5.8. *Let $\mathcal{Q}_i = (X_i, W_i, \delta_i, X_{i0})$, $i \in \{1,2\}$ be two live and reachable SM and V s.t. $\pi_V(W_1) = \pi_V(W_2) \neq \emptyset$. Then $\mathcal{R} \in \mathfrak{S}_V(\mathcal{Q}_1, \mathcal{Q}_2)$ iff*

$$\forall x_1 \in X_{10} \,.\, (\exists x_2 \in X_{20} \,.\, (x_1, x_2) \in \mathcal{R}) \quad \textit{and} \tag{5.6a}$$

$$\begin{aligned}
&\forall\, (x_1, x_2) \in \mathcal{R}, w_1 \in W_1, x_1' \in X_1 \,. \\
&\quad (x_1, w_1, x_1') \in \delta_1 \Rightarrow \exists x_2' \in X_2, w_2 \in W_2 \,.\, \begin{pmatrix} (x_2, w_2, x_2') \in \delta_2 \\ \wedge \pi_V(w_1) = \pi_V(w_2) \\ \wedge (x_1', x_2') \in \mathcal{R} \end{pmatrix}.
\end{aligned} \tag{5.6b}$$

Proof. Let $\Sigma_{S,i}$ be the ASSDS induced by $\mathcal{Q}_i$ and $\Sigma_{S,i}^{\phi} = [\Sigma_{S,i}]^{\phi_{\mathbf{i}}}$ the $\phi_{S,\mathbf{i}}$-*extensions* of $\Sigma_{S,i}$. Then it follows form Figure 5.4 (middle), Proposition 2.19 and (5.5) that

$$\Sigma_{S,i}^{\phi} = [\mathcal{Q}_i]^{\phi_{\mathbf{i}}} = (T, T, W_i \times X_i, W_i, \phi_{\mathbf{i}}(\mathcal{B}_f(\mathcal{Q}_i))).$$

Using (4.3) this implies that $X_i^{\phi_{\mathbf{i}},0} = X_{i0}$ and hence (5.2a) and (5.6a) coincide. It therefore suffices to show that (5.6b) holds for $\mathcal{Q}_1$ and $\mathcal{Q}_2$ iff (5.2b) holds for $\Sigma_{S,1}^{\phi}$ and $\Sigma_{S,2}^{\phi}$.

"$\Rightarrow$": To prove this statement we pick $\mathcal{R}$ s.t. (5.6b) holds and show that (5.2b) also holds for $\mathcal{R}$, i.e., we pick

$$(x_1, x_2) \in \mathcal{R},\ ((\omega_1, \xi_1), \omega_1, \theta_1) \in \phi_1\ ,\ k_1 \in \mathbb{N}_0 \quad s.t. \quad \xi_1({\theta_1}^{-1}(k_1)) = x_1$$

and show that (5.2b:r) is true[3] .
First observe that (5.5) implies $(\omega_1, \xi_1) \in \mathcal{B}_f(\mathcal{Q}_1)$ and ${\theta_1}^{-1}(k) = k$ for all $k \in \mathbb{N}_0$, hence $\xi_1(k_1) = x_1$. Now it follows from (2.20b) that

$$\forall k \in \mathbb{N}_0 \,.\, (\xi_1(k), \omega_1(k), \xi_1(k+1)) \in \delta_1.$$

For all $k \geq k_1$ we can therefore pic $x_1' = \xi_1(k+1)$ and $w = \omega_1(k)$ (in particular for $k = k_1$) and observe that (5.6b:l) holds. As we know that (5.6b) holds for $\mathcal{R}$ this implies that (5.6b:r) holds for every choice of $k \geq k_1$ above.
Now consider the particular case $k = k_1$. Then (5.6b:r) implies (among other things) the existence of $x_2' \in X_2$ s.t. $((\xi_1(k_1+1), x_2') \in \mathcal{R}$. As (5.6b:l) holds for all $k \geq k_1$ we can therefore aply (5.6b) again implying the existence of $x_2'' \in X$ s.t. $(\xi_1(k_1+2), x_2'') \in \mathcal{R}$. We can therefore repetitively apply (5.6b:l) for all tuples $(\omega_1(k), \xi_1(k))$ reached at time $k \geq k_1$ along the trajectories $(\omega_1, \xi_1) \in \mathcal{B}_f(\mathcal{Q}_1)$ and obtain a tuple of trajectories (ω_2, ξ_2) s.t.

$$\forall k \geq k_1 \,.\, \begin{pmatrix} (\xi_2(k-k_1+k_2), \omega_2(k-k_1+k_2), \xi_2(k-k_1+k_2+1)) \in \delta_2 \\ \wedge \pi_V(\omega_1)(k) = \pi_V(\omega_2)(k-k_1+k_2) \\ \wedge (\xi_1(k), \xi_2(k-k_1+k_2)) \in \mathcal{R} \end{pmatrix} \tag{5.7}$$

from (5.6b:r) where $\xi_2(k_2) = x_2$. As $\mathcal{Q}_2$ is reachable, it follows from (2.23), the first line of (5.7) and (2.20b) that $(\omega_2, \xi_2) \in \mathcal{B}_f(\mathcal{Q}_2)$. Now (5.5) implies $((\omega_2, \xi_2), \omega_2, \theta_2) \in \phi_2$ with ${\theta_2}^{-1}(k) = k$ for all $k \in \mathbb{N}_0$. This immediately implies that the first line of (5.2b:l) holds and that the second and third line of (5.2b:l) follow from the second and third line of (5.7).

"$\Leftarrow$": To prove this statement we pick $\mathcal{R}$ s.t. 5.2b holds and show that 5.6b also holds for $\mathcal{R}$, i.e., we pick

$$(x_1, x_2) \in \mathcal{R}, w \in W, x_1' \in X_1 \quad s.t. \quad (x_1, w, x_1') \in \delta_1$$

and show that (5.6b:r) is true.
As $(x_1, w, x_1') \in \delta_1$ and $\mathcal{Q}$ is live and reachable, we know from (2.23) that there exists $(\omega_1, \xi_1) \in \mathcal{B}_f(\mathcal{Q}_1)$ and $k_1 \in \mathbb{N}_0$ s.t. $x_1 = \xi_1(k_1)$. Using (2.20b) we can pick (ω_1, ξ_1) s.t. $x_1' = \xi_1(k_1+1)$ and $w = \omega_1(k_1)$. Now (5.5) implies the existence of $((\omega_1, \xi_1), \omega_1, \theta_1) \in \phi_1$ s.t.

$$\xi_1({\theta_1}^{-1}(k_1)) = \xi_1(k_1) = x_1 \quad \text{and} \quad \xi_1({\theta_1}^{-1}(k_1+1)) = \xi_1(k_1+1) = x_1',$$

implying that (5.2b:l) holds. As we know that (5.2b) holds for $\mathcal{R}$ this implies that (5.2b:r) holds. Now (5.2b:r) and (5.5) imply the existence of ξ_2 and ω_2 s.t.

$$(\omega_2, \xi_2) \in \mathcal{B}_f(\mathcal{Q}_2), \; \xi_2(k_2) = x_2, \text{ and } \omega_2(k_2) = w.$$

[3]To simplify notation we denote the left and right sides of the implication in (5.2b) by (5.2b:l) and (5.2b:r), respectively. The same convention is used of the implication in (5.6b).

Now it follows from (2.20b) that $(x_2, w, \xi_2(k_2+1)) \in \delta_2$. We can therefore pick $x_2' = \xi_2(k_2+2)$ and observe that (5.6b:r) holds from (5.2b:r). □

Remark 5.2. Recall from Section 3.2 that output deterministic SM can be equivalently represented by TS. Hence, substituting (3.3) in (5.6) and choosing $V = Y$ yields

$$\forall x_1 \in X_{10} \,.\, \exists x_2 \in X_{20} \,.\, (x_1, x_2) \in \mathcal{R} \quad \text{and} \tag{5.8a}$$

$$\forall\, (x_1, x_2) \in \mathcal{R}, x_1' \in X_1, u_1 \in U_1 \,.\, \left(\begin{array}{l} h_1(x_1) = h_2(x_2) \\ \wedge x_1' \in f_1(x_1, u_1) \Rightarrow \exists x_2' \in X_2, u_2 \in U_2 \,.\, \left(\begin{array}{l} x_2' \in f_2(x_2, u_2) \\ \wedge (x_1', x_2') \in \mathcal{R} \end{array} \right) \end{array} \right). \tag{5.8b}$$

It is easy to see that (5.8) coincides with the usual definition of simulation relations for TS, as e.g. in [67, Def. 4.7]. ◁

5.3. Connections to Related Notions of Similarity

To the best of the authors knowledge the only work on similarity relations in the behavioral setting was done by Julius and van der Schaft [27, 26]. In this work behavioral systems Σ with *state maps* φ are used to define simulation relations and it was proven in [26, Thm.5.27] that the constructed bisimulation relations are only equivalence relations if these state maps are past induced and Markovian.

Interestingly, the dynamics of the tuple (Σ, φ) used in [26] can equivalently be modeled by a dynamical system $\Sigma_S = (T, W \times X, \mathcal{B}_S)$ and the properties of past inducedness and Markovianity, given in [26, Sec.3.3.2], coincide with the asynchronous state property in (2.25). Hence, if (Σ, φ) is past induced and Markovian it can be equivalently modeled by an ASSDS Σ_S. In this case we could define its $\phi_{\mathbf{i}}$-extension $[(\Sigma, \varphi)]^{\phi_{\mathbf{i}}}$ by the second line in Figure 5.4 and apply Definition 5.7. However, as discussed in Remark 5.1, the simulation relation defined in Definition 5.1 (as used in Definition 5.7) is a weaker notion then the one previously presented in [54, Def. 4]. Using [54, Def. 4] in Definition 5.7 instead, it can be shown that the resulting bisimulation relation coincides with the one given in [26, Def. 5.21].

The idea of constructing $\phi_{\mathbf{i}}$-extensions of state-space models with unique time axis from Figure 5.4 can be applied equivalently to other system models. The simplest one are state machines and transition systems, already discussed in Theorem 5.8 and Remark 5.2, respectively. Analogously, using $T = \mathbb{R}_0^+$ and $W = U \times Y \times D$, where D is the disturbance space, we can construct $\mathcal{B}_S$ such that Σ_S captures the dynamics of a linear time invariant system used by van der Schaft in [75]. There, the inputs and outputs are required to match for bisimilar systems, implying $V = U \times Y$. Now constructing the $\phi_{\mathbf{i}}$-extension

$[\Sigma_S]^{\phi_\mathbf{i}}$ of Σ_S and applying Definition 5.7 results in a bisimulation relation which can be shown to coincide with the one given in [75, Def. 2.1].

All aforementioned notions of (bi)similarity focus on relating systems with unique time scale resulting in a state trajectory matching requirement as depicted in Figure 5.3. To the best of the authors knowledge, the only work explicitly introducing simulations and bisimulations between systems with different time scales was presented by Davoren and Tabuada [8] using so called general flow systems (GFS) [9].

Interestingly, the intuitive interpretation of the different simulation relations depicted in Figure 5.1 - 5.3 coincides with the idea behind the r-, p- and t-simulation relations constructed in [8] for GFL, respectively. For discrete and continuous systems, which are a subclass of GFS, the simulation relation in Definition 5.1 can therefore be shown to reproduce the relations in [8] by choosing appropriate time scale transformations. However, Definition 5.1 extends the constructions in [8] for the mentioned subclass of GFS by including external trajectories. Furthermore, relating two systems with different time scale transformations allows for a larger variety of relations when using Definition 5.1 instead of r-, p- and t-simulation relations constructed in [8].

6. A New Approach to Realizing SAlCA

In the original version of QBA, discussed in Section 3.1, the abstraction is obtained from a transition system, modeling the "original" system. In this model the external signal space is usually divided into inputs and outputs. On the other hand, SAlCA are constructed from a behavior $\mathcal{B}$ evolving over the discrete time axis $T = \mathbb{N}_0$ and taking values in some signal space W, where the division of the latter into inputs and outputs is not necessarily assumed. However, if the abstraction should be used for control purposes, $W = U \times Y$ is a practical choice of the external signal space W. Hence, using Assumption 1 when constructing SAlCA builds a bridge to the setting of QBA.

However, as a direct consequence of the abstraction procedure for SAlCA, using Assumption 1 implies that both U and Y are used to construct the state space of $\widehat{\mathcal{Q}}^{l\uparrow}$ (see (2.42a) in Corollary 2.34). Contrary, QBA use only the output space to construct the abstract state space (see (3.5a) in Theorem 3.8). Hence, an obvious way to connect both settings is to use $W = Y$ when constructing SAlCA. However, as the labels of the transitions in $\widehat{\delta}^{l\uparrow}$ are given by $w \in W$ (see (2.42c) in Corollary 2.34), the resulting finite state machine $\widehat{\mathcal{Q}}^{l\uparrow}$ cannot contain input information in this case, hence cannot be used for control purposes as outlined above. To circumvent this problem, we suggest a different way of constructing $\widehat{\mathcal{Q}}^{l\uparrow}$ in this chapter.

In Section 6.1 we will first recall how the construction of SAlCA from Section 1.2 can be applied to the original model given by an ASSϕDS from Chapter 4 in an obvious manner. This reveals that, using Assumption 2, the standard realization $\widehat{\mathcal{Q}}^{l\uparrow}$ of the resulting SAlCA can be equivalently obtained by using the state machine $\mathcal{Q}$ instead of $\check{\Sigma}_S^{\phi}$ in Figure 4.3 as the "original" model. Using this observation, we derive a construction of $\widehat{\mathcal{Q}}^{l\uparrow}$ from $\mathcal{Q}$ directly in Section 6.2 and show, how this allows to incorporate different choices of abstract state spaces.

Thereafter, we will take the first step in our bridging endeavor in Section 6.3 and Section 6.4, where we investigate how the newly constructed realizations of SAlCA are related, respectively, to the original model and among each other. These results reveal a nice connection to QBA, which allow for simulation relations to the original system by construction.

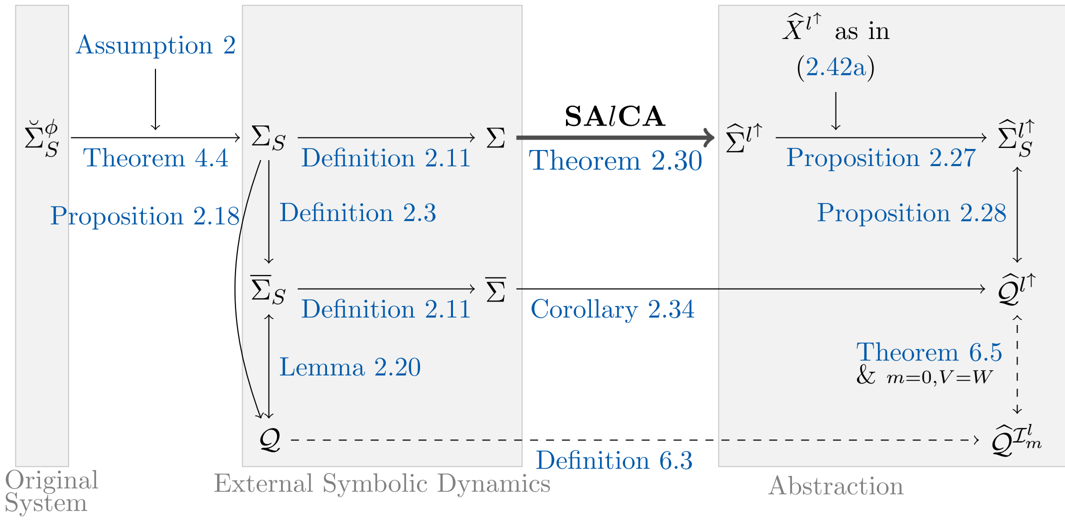

Figure 6.1.: Connection between the original system modeled by an ASSϕDS $\breve{\Sigma}_S^{\phi}$ and the standard realization $\widehat{\mathcal{Q}}^{l\uparrow}$ of its SAlCA under Assumption 2. The dashed lines illustrate the construction of $\widehat{\mathcal{Q}}^{l\uparrow}$ form $\mathcal{Q}$ directly, as discussed in Section 6.2.

6.1. Constructing SAlCA from ASSϕDS

The construction of the state machine $\widehat{\mathcal{Q}}^{l\uparrow}$ from Corollary 2.34 depicted in Figure 2.6 is based on a behavioral system model $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$. The basic idea of SA$l$CA is to construct an SM whose external behavior $\widehat{\mathcal{B}}^{l\uparrow}$ exactly mimics the behavior $\overline{\mathcal{B}}$ over finite time intervals of length $l+1$, hence is the result of the simplified domino-game discussed in Section 2.8.

It was shown in Chapter 4 that under Assumption 2 the external dynamics of the "original" system can be modeled by an ASSDS $\Sigma_S = (\mathbb{N}_0, W \times X, \mathcal{B}_S)$. In this case, Σ can obviously be obtained from Σ_S by simply projecting its behavior to the external signal space W. This observation allows to connect Figure 2.6 and Figure 4.3 to obtain the construction of $\widehat{\mathcal{Q}}^{l\uparrow}$ abstracting the external behavior of $\breve{\Sigma}_S$ as depicted in Figure 6.1.

By investigating Figure 6.1 it is obvious that the construction of $\widehat{\mathcal{Q}}^{l\uparrow}$ does not require realizability of $\breve{\Sigma}_S^{\phi}$ by a state machine. Nevertheless, if the latter is true, $\overline{\mathcal{B}}$ can be obtained from the SM $\mathcal{Q}$ as $\overline{\mathcal{B}} = \pi_W(\mathcal{B}_f(\mathcal{Q}))$ (from Lemma 2.20). However, if $\mathcal{Q}$ is given and used as the "original" model, we can construct $\widehat{\mathcal{Q}}^{l\uparrow}$ from $\mathcal{Q}$ directly without taking a "detour" over constructing $\overline{\mathcal{B}}$ first. We will see in the following section that this builds the bridge to QBA as it allows for different choices of abstract state spaces.

6.2. Constructing Realizations of SAlCA from Infinite SM

Given Assumption 2, the abstraction process of SAlCA will be reduced in this section to the construction of an abstract SM $\widehat{\mathcal{Q}}$ from the given infinite SM

$$\mathcal{Q} = (X, W, \delta, X_0), \; s.t. \; (2.23) \text{ holds}, \tag{6.1a}$$

as depicted by the dashed lines in Figure 6.1. Recall that (2.23) holds for $\mathcal{Q}$ in Figure 6.1 from Proposition 2.18. While the construction of $\widehat{\mathcal{Q}}^{l\uparrow}$ requires the state space to be constructed from W, we also want to incorporate the possibility to only consider a subspace of W, e.g., only inputs U or only outputs Y) for the state space construction to build a bridge to QBA. To simplify notation, we therefore introduce the set

$$V \quad s.t. \quad \pi_V(W) \neq \emptyset \quad \text{and} \quad |V| < \infty \tag{6.1b}$$

as the choice of considered signals for the state space construction. Assuming that V is finite ensures that the obtained abstraction is an FSM. To simplify notation for this state space construction we introduce the behavior

$$\mathcal{B}_V := \pi_V(\mathcal{B}_f(\mathcal{Q})) \quad \text{and} \quad \mathcal{B}_{SV} := \pi_{V\times X}(\mathcal{B}_f(\mathcal{Q})) \tag{6.1c}$$

and extend its time axis from $\mathbb{N}_0$ to $\mathbb{Z}$ by pre-appending each $\nu \in \mathcal{B}_V$ (resp. $(\nu, \xi) \in \mathcal{B}_{SV}$) with the special symbol $\diamond$, e.g., $\nu = v_0 v_1 v_2 \ldots \in \mathcal{B}_V \subseteq (V)^{\mathbb{N}_0}$ is transformed to ${}^{\diamond}\nu = \ldots \diamond\diamond\diamond v_0 v_1 v_2 \ldots \in {}^{\diamond}\mathcal{B}_V \subseteq (V \cup \{\diamond\})^{\mathbb{Z}}$. This extension is also used for other behaviors in this section and always denoted by the left-raised $\diamond$ annotation. With this slight change of notation, we redefine the domino-set in (2.37) to

$$D_{l+1} := \bigcup_{k' \in \mathbb{N}_0} {}^{\diamond}\mathcal{B}_V|_{[k'-l,k']}. \tag{6.1d}$$

Playing the domino game from Section 2.8 with the set in (6.1d) results in the abstract behavior

$${}^{\diamond}\widehat{\mathcal{B}}_V^{l\uparrow} := \left\{ \nu \,\middle|\, \begin{pmatrix} \forall k < 0 \,.\, \nu(k) = \diamond \\ \wedge \forall k \in \mathbb{N}_0 \,.\, \nu|_{[k-l,k]} \in D_{l+1} \end{pmatrix} \right\}. \tag{6.2}$$

Slightly abusing our terminology from Section 2.8, we call $\widehat{\Sigma}_V^{l\uparrow} := (\mathbb{N}_0, V, \widehat{\mathcal{B}}_V^{l\uparrow})$ the SAlCA of Σ w.r.t. V. This is justified, as the behavior ${}^{\diamond}\widehat{\mathcal{B}}_V^{l\uparrow}$ coincides with ${}^{\diamond}\widehat{\mathcal{B}}^{l\uparrow}$ for $V = W$, what is shown in the following proposition. However, if $V \neq W$, e.g., $V = Y$, the left-extension of the behavior $\pi_V(\widehat{\mathcal{B}}^{l\uparrow})$ is generally only a subset of ${}^{\diamond}\widehat{\mathcal{B}}_V^{l\uparrow}$. Intuitively, this is due to the fact that playing the domino game with dominos over $W = U \times Y$ might restrict the allowed moves due to the additional matching requirement on the past inputs.

Proposition 6.1. *Given (6.1) and the system models depicted in Figure 6.1 it holds that ${}^{\diamond}\widehat{\mathcal{B}}^{l\uparrow} = {}^{\diamond}\widehat{\mathcal{B}}_V^{l\uparrow}$ if $V = W$. Hence, $\widehat{\Sigma}_V^{l\uparrow} = \widehat{\Sigma}^{l\uparrow}$ if $V = W$.*

Proof. Let Σ in Figure 6.1 be the dynamical system $\Sigma = (\mathbb{N}_0, W, \mathcal{B})$. Then it follows from Proposition 2.18 and Lemma 2.20 (i) that $\overline{\mathcal{B}} = \pi_W(\mathcal{B}_f(\mathcal{Q}))$ and therefore $\mathcal{B}_V = \overline{\mathcal{B}}$ if $V = W$. Now recall from (2.38) and Proposition 2.32 that this implies

$$\widehat{\mathcal{B}}^{l\uparrow} = \left\{ \nu \middle| \begin{pmatrix} \omega|_{[0,l-1]} \in \mathcal{B}_V|_{[0,l-1]} \\ \wedge \forall k \in \mathbb{N}_0 \, . \, \omega|_{[k,k+l]} \in \bigcup_{k' \in \mathbb{N}_0} \mathcal{B}_V|_{[k',k'+l]} \end{pmatrix} \right\}. \tag{6.3}$$

Now consider the left extensions ${}^{\diamond}\widehat{\mathcal{B}}^{l\uparrow}$ and ${}^{\diamond}\mathcal{B}_V$ and the domino set in (6.1d) for $V = W$. Then the second line of (6.2) is obtained from (6.3) by substituting $\bigcup_{k' \in \mathbb{N}_0} \mathcal{B}_V|_{[k',k'+l]}$ by the domino set D_{l+1} and shifting the restriction window for ω from $[k, k+l]$ to $[k-l, k]$. By additionally requiring $\forall k < 0 \, . \, \nu(k) = \diamond$ it is ensured that for any time $k < l$ only dominos with $l-k$-diamonds on the left can be attached in the domino game. This immediately implies that at time $k = l-1$ a domino with only one diamond and l symbols from W is attached, which can only be obtained from ${}^{\diamond}\mathcal{B}_V|_{[-1,l]}$ ensuring $\omega|_{[0,l-1]} \in \mathcal{B}_V|_{[0,l-1]}$. $\square$

Using Proposition 6.1 it is now easy to see that the abstract state space $\widehat{X}^{l\uparrow}$ in (2.42a) of Corollary 2.34 can be redefined to $\{\diamond\}^l \cup D_l$ if $V = W$ is chosen. This simply means that all states with length $r < l$ are pre-appended by $l-r$ diamonds while all states with length l are left unchanged. Hence $\{\diamond\}^l \cup D_l$ is obviously a good candidate for an abstract state space, where each state has length l, which significantly simplifies the subsequent construction of abstract state machines.

Corresponding External Event Sequences

As it is our goal in this section to construct abstract state machines from $\mathcal{Q}$ directly, we would like to derive the transition relation $\widehat{\delta}$ directly from δ. As a first step, we investigate the connection between the state spaces, i.e., how a string $\zeta \in \{\diamond\}^l \cup D_l$ can correspond to a state $x \in X$ of $\mathcal{Q}$. Observe that ζ is a string of length l and x is a state reached at a particular time $k \in \mathbb{N}_0$. We consider the cases where ζ is generated by $\mathcal{Q}$ immediately *before*, immediately *after* or *while* x was reached. This leads us to a set of intervals

$$\mathcal{I}_m^l = [m-l, m-1] \quad s.t. \quad l, m \in \mathbb{N}_0, \text{ and } m \leq l, \tag{6.4}$$

where[1] $[k,k] + \mathcal{I}_0^l = [k-l, k-1]$ corresponds to the first, $[k,k] + \mathcal{I}_l^l = [k, k+l-1]$ corresponds to the second, and all other choices $[k,k] + \mathcal{I}_m^l$ correspond to the third case. Based on (6.4) the set of compatible states are defined in Definition 6.2 and illustrated in Figure 6.2.

[1] The addition of two intervals is interpreted in the usual sense, i.e., $[a,b] + [c,d] = [a+c, b+d]$.

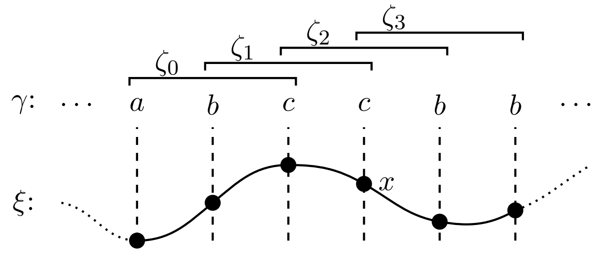

Figure 6.2.: Illustration of corresponding external event sequences $\zeta_m \in \mathrm{E}^{\mathcal{I}^3_m}(x),\ m \in \{0, \ldots, 3\}$ for state $x = \xi(k)$ where $V = Y = \{a, b, c\}$ and $\gamma(k) \in \mathrm{H}_\delta(x),\ k \in \mathbb{N}_0$.

Definition 6.2. *Given* (6.1) *and* (6.4)*, the set of* corresponding external event sequences w.r.t. $\mathcal{I}^l_m$ *of a state* $x \in X$ *is defined by*

$$\mathrm{E}^{\mathcal{I}^l_m}(x) := \left\{ \zeta \middle| \exists (\nu, \xi) \in \mathcal{B}_{SV}, k \in \mathbb{N}_0 \,.\, \begin{pmatrix} \xi(k) = x \\ \wedge \zeta = \nu|_{[k,k]+\mathcal{I}^l_m} \end{pmatrix} \right\}. \tag{6.5}$$

Furthermore, $\mathcal{Q}$ *is* future unique *w.r.t.* $\mathcal{I}^l_m$ *if*

$$\forall x \in X, \zeta, \zeta' \in \mathrm{E}^{\mathcal{I}^l_m}(x) \,.\, \zeta|_{[l-m,l-1]} = \zeta'|_{[l-m,l-1]}. \tag{6.6}$$

Observe that $\zeta, \zeta' \in \mathrm{E}^{\mathcal{I}^l_m}(x)$ in (6.6) are obtained from two trajectories (ν, ξ), $(\nu', \xi') \in \mathcal{B}_{SV}$ passing x at time $k \in \mathbb{N}_0$ and $k' \in \mathbb{N}_0$, respectively, (i.e., $\xi(k) = \xi'(k') = x$) using (6.5). During this restriction of ν (resp. ν') to ζ (resp. ζ') absolute time information is disregarded (see Section 2.1), implying $\zeta|_{[l-m,l-1]} = \nu|_{[k,k+m-1]}$ and $\zeta'|_{[l-m,l-1]} = \nu|_{[k',k'+m-1]}$. Therefore, $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}^l_m$ if for all states $x \in X$ all trajectories passing x have the same m-long (non-strict) future of external events, i.e. $\nu|_{[k,k+m-1]} = \nu'|_{[k',k'+m-1]}$. Using this intuition it is easy to see that $\mathcal{Q}$ is always *future unique* w.r.t. $\mathcal{I}^l_0 = [-l, -1]$, as this interval has no future.

$\mathcal{I}^l_m$-Abstract State Machines

We now proceed by constructing m different abstract state machines of $\mathcal{Q}$ using the outlined correspondence between X and D_l via $\mathcal{I}^l_m$.

Definition 6.3. *Given* (6.1) *and* (6.4)*, the* $\mathcal{I}^l_m$*-abstract state machine of* $\mathcal{Q}$ *is defined by* $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m} = (\widehat{X}^{\mathcal{I}^l_m}, U \times Y, \widehat{\delta}^{\mathcal{I}^l_m}, \widehat{X}^{\mathcal{I}^l_m}_0)$ *s.t.*

$$\widehat{X}^{\mathcal{I}^l_m} := \left\{ \zeta \middle| \exists x \in X \,.\, \zeta \in \mathrm{E}^{\mathcal{I}^l_m}(x) \right\}, \tag{6.7a}$$

$$\widehat{X}^{\mathcal{I}^l_m}_0 := \left\{ \zeta \middle| \exists x \in X_0 \,.\, \zeta \in \mathrm{E}^{\mathcal{I}^l_m}(x) \right\}, \text{ and} \tag{6.7b}$$

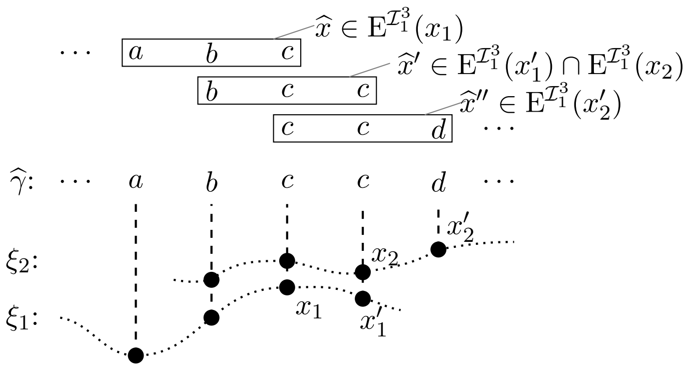

Figure 6.3.: Illustration of construction of abstract states $\widehat{x}, \widehat{x}', \widehat{x}'' \in \widehat{X}^{\mathcal{I}_1^3}$ and trajectories $\widehat{\xi}$ and $\widehat{\gamma}$ from the original system $\mathcal{Q}$ where $V = Y = \{a, b, c, d\}$.

$$\widehat{\delta}^{\mathcal{I}_m^l} := \left\{ (\widehat{x}, w, \widehat{x}') \;\middle|\; \begin{pmatrix} \widehat{x}'|_{[0,l-m-1]} = \left(\widehat{x}|_{[0,l-m-1]} \cdot \pi_V(w)\right)|_{[1,l-m]} \\ \wedge \widehat{x}|_{[l-m,l-1]} = \left(\pi_V(w) \cdot \widehat{x}'|_{[l-m,l-2]}\right)|_{[0,m-1]} \\ \wedge \exists x, x' \in X \;.\; \begin{pmatrix} \widehat{x} \in \mathrm{E}^{\mathcal{I}_m^l}(x) \\ \wedge \widehat{x}' \in \mathrm{E}^{\mathcal{I}_m^l}(x') \\ \wedge (x, w, x') \in \delta \end{pmatrix} \end{pmatrix} \right\}. \tag{6.7c}$$

The construction of the abstract state machines in Definition 6.3 are illustrated in Figure 6.3 and can be interpreted as follows. Using (6.7a) instead of $\widehat{X}^{\mathcal{I}_m^l} = D_l \cup \{\diamond\}^l$ ensures that $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ is live and reachable, which is purely cosmetic but allows to simplify subsequent proofs. The last line in the conjunction of (6.7c) simply says that we have a transition in $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ from $\widehat{x}$ to $\widehat{x}'$ if there is a transition in $\mathcal{Q}$ between any two states compatible with $\widehat{x}$ and $\widehat{x}'$, respectively. However, the first two lines in the conjunction of (6.7c) additionally ensure that $\widehat{x}$ and $\widehat{x}'$ obey the rules of the domino game, i.e.,

$$\widehat{x}|_{[1,l-1]} = \widehat{x}'|_{[0,l-2]}$$

and the current external event $v = \pi_V(u, y)$ is contained in either $\widehat{x}$ or $\widehat{x}'$ or both, at the position corresponding to the current time point, i.e.,

$$\begin{aligned} &v = \widehat{x}'(l-1) \text{ if } m = 0, \\ &v = \widehat{x}(l-m) = \widehat{x}'(l-1-m) \text{ if } 0 < m < l \text{ and} \\ &v = \widehat{x}(0) \text{ if } m = l. \end{aligned}$$

Obviously, one could neglect these two conditions and still obtains a valid abstraction. We will discuss this generalized abstraction briefly in Remark 6.3.

Realizing SAlCA

Recall that the abstract state machines $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ constructed in Definition 6.3 only ensure that symbolic sequences taking values in the signal space V obey the rules of the domino game from Section 2.8 played with dominos from (6.1d). Therefore, $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ only realizes $\widehat{\Sigma}^{l\uparrow}_V$ w.r.t. V.

Theorem 6.4. *Given (6.1) and $\widehat{\Sigma}^{l\uparrow}_V = (\mathbb{N}_0, V, \widehat{\mathcal{B}}^{l\uparrow}_V)$, let $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ be defined as in Definition 6.3. Then $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ realizes $\widehat{\Sigma}^{l\uparrow}_V$ w.r.t. V.*

Proof. See Section A.1.1 on page 134. □

As an intuitive consequence of Theorem 6.4, choosing $m = 0$ and the full external event set $V = W$ when constructing $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ in Definition 6.3 yields the "standard" realization $\widehat{\mathcal{Q}}^{l\uparrow}$ of SAlCA in Corollary 2.34 if all states of length $r < l$ are pre-appended by $l - r$ diamonds. This is formalized by a trivial bisimulation in the following theorem.

Theorem 6.5. *Given (6.1) with $V = W$, let $\widehat{\mathcal{Q}}^{l\uparrow}$ be constructed as depicted by one of the solid paths in Figure 6.1 and $\widehat{\mathcal{Q}}^{\mathcal{I}^l_0}$ as in Definition 6.3. Furthermore, let*

$$\mathcal{R} = \left\{ (\widehat{x}, {}^{\diamond}\widehat{x}) \in \widehat{X}^{l\uparrow} \times \widehat{X}^{\mathcal{I}^l_0} \,\middle|\, {}^{\diamond}\widehat{x} = \diamond^{l-|\widehat{x}|} \cdot \widehat{x} \right\} \tag{6.8}$$

where $\diamond^r$ is an r-long string containing only diamonds. Then $\mathcal{R} \in \mathfrak{B}_W(\widehat{\mathcal{Q}}^{l\uparrow}, \widehat{\mathcal{Q}}^{\mathcal{I}^l_0})$.

Proof. See Section A.1.2 on page 135. □

Observe from Theorem 6.5 that $\widehat{\mathcal{Q}}^{l\uparrow}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}^l_0}$ are equivalent up to a trivial renaming of states. This result finally explains the dashed lines in Figure 6.1 from $\mathcal{Q}$ to $\widehat{\mathcal{Q}}^{l\uparrow}$. Observe, that we need to restrict the connection between the two SM to the case $m = 0$ and $V = W$ as $\widehat{\Sigma}^{l\uparrow}_S$ is assumed to be constructed from $\widehat{\Sigma}^{l\uparrow}$ using the state space $\widehat{X}^{l\uparrow}$ in (2.42a). If we drop these restrictions, we get a slightly different connection.

Proposition 6.6. *Given $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ as in Definition 6.3, the dynamical system $\widehat{\Sigma}^{l\uparrow}_S = (\mathbb{N}_0, W \times \widehat{X}^{\mathcal{I}^l_m}, \mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}))$ is an ASSDS and realized by $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ w.r.t. $W \times \widehat{X}^{\mathcal{I}^l_m}$. Furthermore, $\widehat{\Sigma}^{l\uparrow}_S$ is an asynchronous state space representation of $\widehat{\Sigma}^{l\uparrow}_V = (\mathbb{N}_0, V, \widehat{\mathcal{B}}^{l\uparrow}_V)$ w.r.t. V, i.e., $\pi_V(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m})) = \widehat{\mathcal{B}}^{l\uparrow}_V$.*

Proof. The first claim follows immediately from Proposition 2.19. As $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ is live and reachable by definition, the second claim follows from Lemma 2.21. Finally, $\pi_V(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m})) = \widehat{\mathcal{B}}^{l\uparrow}_V$ directly follows from Theorem 6.4. □

Using Proposition 6.6 we obtain a new chain of constructions from the original system $\breve{\Sigma}^{\phi}_S$ to the new realizations $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ of its SAlCA w.r.t. V as depicted in Figure 6.4. It should

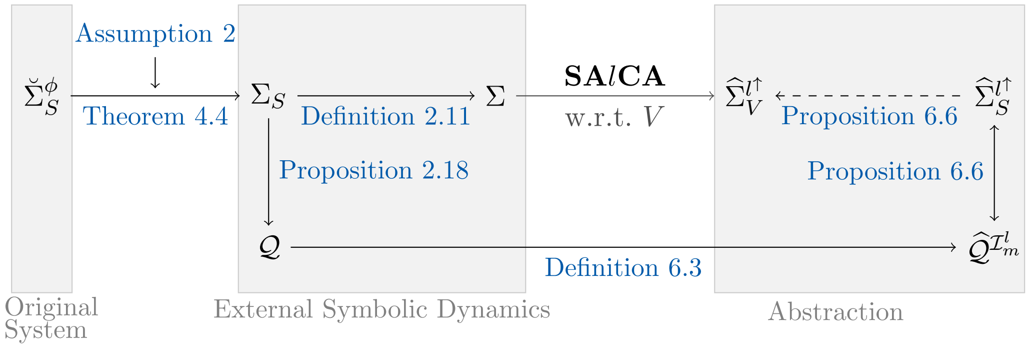

Figure 6.4.: Connection between the original system modeled by an ASSϕDS $\breve{\Sigma}_S^{\phi}$ and the new realizations $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ of its SAlCA w.r.t. V under Assumption 2.

be noted that whenever $V \neq W$, input information is only contained in $\widehat{\Sigma}_S^{l\uparrow}$ but not in $\widehat{\Sigma}_V^{l\uparrow}$. Hence, in this case $\widehat{\Sigma}_S^{l\uparrow}$ cannot be constructed from $\widehat{\Sigma}_V^{l\uparrow}$. However, if $V = W$ the connection between $\widehat{\Sigma}_V^{l\uparrow}$ and $\widehat{\Sigma}_S^{l\uparrow}$ as depicted in Figure 6.1 can be established by choosing the state space $\widehat{X}^{\mathcal{I}_m^l}$ instead of $\widehat{X}^{l\uparrow}$.

Remark 6.1. In the context of SlCA a state machine $\widehat{\mathcal{Q}}^{l+}$ was introduced in [47] whose state at time k represents the string of external symbols from time $k-l+1$ to time k, i.e., from the interval $[k,k]+\mathcal{I}_1^l$. While the state sets of $\widehat{\mathcal{Q}}^{l+}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_1^l}$ coincide, their transition structure slightly differs. This is a consequence of the fact that $\widehat{\mathcal{Q}}^{l+}$ was intended to serve as a set-valued observer for the states of $\mathcal{Q}$ in [47] and not as an abstraction for controller synthesis. $\triangleleft$

6.3. Relating the Abstraction to the "Original" System

Using Assumption 2 the original system is modeled by an ASSϕDS $\breve{\Sigma}_S^{\phi}$. However, we have seen in the previous chapter that the abstract state machines $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ can be derived from $\mathcal{Q}$ directly, as depicted in Figure 6.4. It is therefore interesting to investigate if the question of relation between the "original" and the "abstract" system can also be reduced to investigating the relation between the SM $\mathcal{Q}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ as the latter can be done locally.

Similarity of $\breve{\Sigma}_S^{\phi}$ and $\widehat{\Sigma}_S^{l\uparrow}$

Using the general notion of similarity for ASSϕDS and the construction of $\phi_{\mathbf{i}}$-extensions for ASSDS and SM we can show on a very general level when similarity of $\breve{\Sigma}_S^{\phi}$ and $\widehat{\Sigma}_S^{l\uparrow}$ can be reduced to similarity of $\mathcal{Q}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$.

Theorem 6.7. *Given the system models depicted in Figure 6.4 and (6.1b) it holds that*

$$\mathcal{R} \in \mathfrak{S}_V(\mathcal{Q}, \widehat{\mathcal{Q}}^{\mathcal{I}_m^l}) \Leftrightarrow \mathcal{R} \in \mathfrak{S}_V(\breve{\Sigma}_S^{\phi}, \widehat{\Sigma}_S^{l\uparrow}), \tag{6.9a}$$

$$\mathcal{R} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}, \mathcal{Q}) \Leftarrow \ \mathcal{R} \in \mathfrak{S}_V(\widehat{\Sigma}_S^{l\uparrow}, \breve{\Sigma}_S^{\phi}) \quad \textit{and} \tag{6.9b}$$

$$\Sigma_S \textit{ is complete} \Rightarrow \left(\mathcal{R} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}, \mathcal{Q}) \Rightarrow \mathcal{R} \in \mathfrak{S}_V(\widehat{\Sigma}_S^{l\uparrow}, \breve{\Sigma}_S^{\phi})\right). \tag{6.9c}$$

Proof. See Section A.1.3 on page 136. □

Using Definition 5.2 we have the following immediate consequence of Theorem 6.7.

Corollary 6.8. *Given the system models depicted in Figure 6.4 and (6.1b) it holds that*

$$\mathcal{Q} \preceq_V \widehat{\mathcal{Q}}^{\mathcal{I}_m^l} \Leftrightarrow \breve{\Sigma}_S^{\phi} \preceq_V \widehat{\Sigma}_S^{l\uparrow} \qquad \textit{and} \tag{6.10a}$$

$$\Sigma_S \textit{ is complete} \Rightarrow \left(\mathcal{Q} \cong_V \widehat{\mathcal{Q}}^{\mathcal{I}_m^l} \Leftrightarrow \breve{\Sigma}_S^{\phi} \cong_V \widehat{\Sigma}_S^{l\uparrow}\right). \tag{6.10b}$$

Intuitively, the result in (6.10b) is not very surprising, as bisimilarity implies behavioral equivalence of all models, hence $\widehat{\mathcal{B}}_V^{l\uparrow} = \mathcal{B}_V = \pi_V(\mathcal{B})$. However the last equality only holds if Σ_S is complete as $\mathcal{B}_V = \pi_V(\overline{\mathcal{B}})$. However, this should not worry us too much, as in this case using SAlCA can never result in an ASSDS $\widehat{\Sigma}_S^{l\uparrow}$ which is bisimilar to $\breve{\Sigma}_S^{\phi}$ as the former is always complete. Hence, for all situations where bisimilarity of $\widehat{\Sigma}_S^{l\uparrow}$ and $\breve{\Sigma}_S^{\phi}$ can be achieved, it can actually be reduced to investigating state machines.

Similarity of $\mathcal{Q}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$

We now investigate under which conditions the obtained abstractions $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ actually simulate the original state machine $\mathcal{Q}$. This requires the construction of a relation between the original state space X and the abstract state space $\widehat{X}^{\mathcal{I}_m^l}$. As $\widehat{X}^{\mathcal{I}_m^l}$ defines a cover for X s.t. each cell is given by all states x corresponding to a string $\zeta \in \widehat{X}^{\mathcal{I}_m^l}$ via $\mathrm{E}^{\mathcal{I}_m^l}$, the latter is a natural choice for a relation between X and $\widehat{X}^{\mathcal{I}_m^l}$.

Recall from Lemma 2.31 (iv) and the construction of $\widehat{\Sigma}_V^{l\uparrow}$ depicted in Figure 6.4 that $\mathcal{B}_V = \widehat{\mathcal{B}}_V^{l\uparrow}$ if $\Sigma_V = (\mathbb{N}_0, V, \mathcal{B}_V)$ is asynchronously l-complete. In this case, we know from Proposition 6.6 that the V-part of the external behavior of $\mathcal{Q}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ coincide. As shown in Proposition 5.4, behavioral equivalence is always necessary for a relation $\mathcal{R}$ to be a bisimulation relation, but usually not sufficient. We therefore introduce a stronger condition, called *state-based asynchronous l-completeness*, to serve the latter purpose.

Definition 6.9. *Given (6.1) and (6.4), we say that $\mathcal{Q}$ is* state-based asynchronously l-complete w.r.t. $\mathcal{I}_m^l$ *if*

$$\forall x \in X, \zeta \in D_{l+1} \,.\, \left(\zeta|_{[0,l-1]} \in \mathrm{E}^{\mathcal{I}_m^l}(x) \Rightarrow \zeta \in \mathrm{E}^{[m-l,m]}(x)\right). \tag{6.11}$$

Theorem 6.10. *Given* (6.1) *and* $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ *as in Definition 6.3, let*

$$\mathcal{R} = \left\{(x, \widehat{x}) \in X \times \widehat{X}^{\mathcal{I}^l_m} \middle| \widehat{x} \in \mathrm{E}^{\mathcal{I}^l_m}(x)\right\}. \tag{6.12}$$

Then it holds that

(i) $\mathcal{R} \in \mathfrak{S}_W(\mathcal{Q}, \widehat{\mathcal{Q}}^{\mathcal{I}^l_m}) \Leftrightarrow \mathcal{Q}$ *is future unique w.r.t.* $\mathcal{I}^l_m$ *and*

(ii) $\mathcal{R}^{-1} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}, \mathcal{Q}) \Leftrightarrow \mathcal{Q}$ *is state-based asyc. l-compl. w.r.t.* $\mathcal{I}^l_m$

Proof. See Section A.1.4 on page 138. □

Remark 6.2. Recall from Definition 2.24 that $\Sigma_V = (\mathbb{N}_0, V, \mathcal{B}_V)$ is asynchronously l-complete if $\nu \in \widehat{\mathcal{B}}^{l\uparrow}_V$ implies $\nu \in \mathcal{B}_V$. Intuitively, the latter is true if any connection of two l-long dominos results in an $l+1$-long domino also consistent with $\mathcal{B}_V$. This is equivalent to requiring that *for all* $\zeta \in D_{l+1}$ there *exists* an $x \in X$ s.t. the implication in (6.11) holds. In contrast, $\mathcal{Q}$ is state based asynchronously l-complete w.r.t. $\mathcal{I}^l_m$ if *for all* $\zeta \in D_{l+1}$ and *for all* $x \in X$ the implication in (6.11) holds. Therefore, asynchronous l-completeness of Σ is always implied by (6.11), but not vice-versa. ◁

Intuitively, $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ simulates $\mathcal{Q}$ w.r.t. V if for every related state pair $(x, \widehat{x}) \in \mathcal{R}$ and every transition $(x, w, x') \in \delta$ which $\mathcal{Q}$ "picks", $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ can "pick" a transition $(\widehat{x}, w', \widehat{x}') \in \widehat{\delta}^{\mathcal{I}^l_m}$ s.t. $v = \pi_V(w) = \pi_V(w')$. However, if $m > 0$, a state $\widehat{x} \in \widehat{X}^{\mathcal{I}^l_m}$ has only outgoing transitions s.t. $v = \widehat{x}(l-m)$. Therefore, $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ can only simulate $\mathcal{Q}$ iff in every state $x \in X$ all outgoing transitions agree on this v, i.e., $\mathcal{Q}$ is "output deterministic" w.r.t. V. For $m > 1$ applying this reasoning iteratively gives the (rather restrictive) condition of future uniqueness of $\mathcal{Q}$.

As the outlined problems are absent for $m = 0$ (as $\mathcal{Q}$ is always future unique w.r.t. $\mathcal{I}^l_0$), $\widehat{\mathcal{Q}}^{\mathcal{I}^l_0}$, which we know to coincide with the original realization $\widehat{\mathcal{Q}}^{l\uparrow}$ of SAlCA for $V = W$, always simulates $\mathcal{Q}$.

Corollary 6.11. *Given* (6.1) *and* $\widehat{\mathcal{Q}}^{\mathcal{I}^l_0}$ *as in Definition 6.3 it holds that* $\mathcal{Q} \preceq_W \widehat{\mathcal{Q}}^{\mathcal{I}^l_0}$.

6.4. Ordering Abstractions based on Simulation Relations

Thinking back to the domino game we know that using longer dominos (i.e., increasing l) gives less freedom in composing them and therefore yields a tighter abstraction. This intuition carries over to the abstract state machines $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$, inducing an ordering in terms of simulation relations.

Theorem 6.12. *Given* (6.1) *and* $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ *as in Definition 6.3, let*

$$\mathcal{R} = \left\{(\widehat{x}_{l+1}, \widehat{x}_l) \in \widehat{X}^{\mathcal{I}^{l+1}_m} \times \widehat{X}^{\mathcal{I}^l_m} \middle| \widehat{x}_l = \widehat{x}_{l+1}|_{[1,l]}\right\}. \tag{6.13}$$

Then it holds that

(i) $\mathcal{R} \in \mathfrak{S}_V(\widehat{Q}^{\mathcal{I}_m^{l+1}}, \widehat{Q}^{\mathcal{I}_m^l})$ and
(ii) $\mathcal{R}^{-1} \in \mathfrak{S}_V(\widehat{Q}^{\mathcal{I}_m^l}, \widehat{Q}^{\mathcal{I}_m^{l+1}}) \Leftrightarrow \widehat{\mathcal{B}}_V^{l\uparrow} = \widehat{\mathcal{B}}_V^{l+1\uparrow}$

Proof. See Section A.1.5 on page 140. ☐

Recall from (6.1c) and Lemma 2.31 (i) and (iv) that asynchronous l-completeness of $\Sigma_V = (\mathbb{N}_0, V, \mathcal{B}_V)$ implies $\widehat{\mathcal{B}}_V^{l\uparrow} = \widehat{\mathcal{B}}_V^{l+1\uparrow}$. Therefore, Theorem 6.12 (ii) implies that the accuracy of the abstraction cannot be increased by increasing $l > r$ if $\Sigma_V = (\mathbb{N}_0, V, \mathcal{B}_V)$ is asynchronously r-complete and m is fixed, e.g., $m = 0$. As the necessary and sufficient property for bisimilarity in Theorem 6.10 is stronger then asynchronous r-completeness (see Remark 6.2), the standard realization $\widehat{Q}^{l\uparrow}$ of SAlCA might never result in a bisimilar abstraction of an r-complete dynamical system, no matter how large l is chosen.

Interestingly, we will show that increasing m, i.e., shifting the interval into the future, results in a “smaller” state machine w.r.t. simulation relations.

Theorem 6.13. *Given (6.1) and $\widehat{Q}^{\mathcal{I}_m^l}$ as in Definition 6.3 with $m < l$, let*

$$\mathcal{R} = \left\{ (\widehat{x}_{m+1}, \widehat{x}_m) \in \widehat{X}^{\mathcal{I}_{m+1}^l} \times \widehat{X}^{\mathcal{I}_m^l} \;\middle|\; \left(\begin{array}{c} \widehat{x}_{m+1}|_{[0,l-2]} = \widehat{x}_m|_{[1,l-1]} \\ \wedge \exists x \in X \;.\; \left(\begin{array}{c} \widehat{x}_{m+1} \in \mathrm{E}^{\mathcal{I}_{m+1}^l}(x) \\ \wedge \widehat{x}_m \in \mathrm{E}^{\mathcal{I}_m^l}(x) \end{array} \right) \end{array} \right) \right\}. \tag{6.14}$$

Then it holds that
(i) $\mathcal{R} \in \mathfrak{S}_V(\widehat{Q}^{\mathcal{I}_{m+1}^l}, \widehat{Q}^{\mathcal{I}_m^l})$ and
(ii) $\mathcal{R}^{-1} \in \mathfrak{S}_V(\widehat{Q}^{\mathcal{I}_m^l}, \widehat{Q}^{\mathcal{I}_{m+1}^l}) \Leftrightarrow \left(\begin{array}{c} \mathcal{Q} \text{ is future unique w.r.t. } \mathcal{I}_{m+1}^l \\ \wedge \mathcal{Q} \text{ is state-based async. } l\text{-complete w.r.t. } \mathcal{I}_m^l \end{array} \right)$.

Proof. See Section A.1.6 on page 142. ☐

It is important to note that future uniqueness and state-based asynchronous l-completeness are incomparable properties, i.e., non is implied by the other. Therefore, there exist situations where $\widehat{Q}^{\mathcal{I}_m^l}$ with $m > 0$ simulates $\mathcal{Q}$ (i.e. $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_m^l$) and $\widehat{Q}^{\mathcal{I}_m^l}$ is tighter than $\widehat{Q}^{\mathcal{I}_0^l}$ in terms of simulation relations. However, if $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_m^l$, it is obviously also future unique w.r.t. $\mathcal{I}_n^l$, $n < m$. In this case, state-based asynchronous l-completeness of $\mathcal{Q}$ w.r.t. $\mathcal{I}_n^l$ implies that (i) $\widehat{Q}^{\mathcal{I}_n^r}$ is bisimilar to $\mathcal{Q}$ (from Theorem 6.10) and (ii) increasing $r > l$ and $m > n$ will not result in a tighter abstraction (from Theorem 6.12 and Theorem 6.13). Hence, by using the additional parameter m when realizing SAlCA, we are able to obtain a bisimilar abstraction if there exists l and m s.t. $\mathcal{Q}$ is both future unique and state-based asynchronous l-complete w.r.t. $\mathcal{I}_m^l$.

Remark 6.3. It is interesting to note that Theorem 6.4 also holds for a more general abstract transition system $\widehat{Q}$ whose transition relation is defined by the last line of the conjunction in (6.7c) only, i.e., it is not required that successive states obey the rules

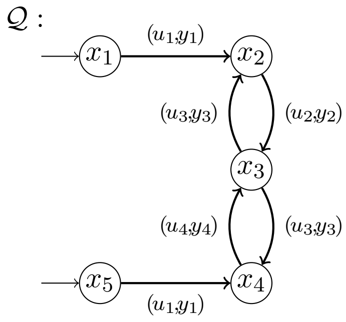

Figure 6.5.: Transition structure of the SM $\mathcal{Q}$ in (6.15).

of the domino game. This choice of $\widehat{\mathcal{Q}}$ obviously gives a weaker abstraction then $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ in Definition 6.3 and does not coincide with $\widehat{\mathcal{Q}}^{l\uparrow}$ in Corollary 2.34 for $m = 0$. However, it solves the issue discussed in Theorem 6.10, i.e., it always simulates $\mathcal{Q}$ for any choice of m. It should be noted though, that Theorem 6.13 (i) does not hold for $\widehat{\mathcal{Q}}$. Actually, different choices of m give incomparable abstractions when the first two lines in (6.7c) are dropped. In this case it strongly depends on the nature of $\mathcal{Q}$ if using the past or the future to construct $\widehat{X}$ is more beneficial in terms of similarity relations. $\triangleleft$

6.5. Example

We conclude this chapter with a detailed example illustrating the construction of $\mathcal{I}_m^l$-abstract state machines and the property of future uniqueness and state-based asynchronous l-completeness for different choices of l and m.

For simplicity, we consider a *finite* state machine

$$\begin{aligned} &\mathcal{Q} = (X, U \times Y, \delta, X_0) \quad s.t. \quad V = Y \quad \text{and} \\ &X = \{x_1, \ldots, x_5\},\ Y = \{y_1, \ldots, y_4\},\ U = \{u_1, \ldots, u_4\} \text{ and } X_0 = \{x_1, x_5\} \end{aligned} \tag{6.15}$$

as the "original" model. The transition structure of $\mathcal{Q}$ is depicted in Figure 6.5, where transitions $(x, u, y, x') \in \delta$ are depicted by an arrow from x to x' labeled by (u, y). It can be inferred from Figure 6.5 that the external behavior of $\mathcal{Q}$ is given by

$$\mathcal{B}_V = \{y_1y_2((y_3y_2)^*(y_3y_4)^*)^\omega, y_1y_4((y_3y_2)^*(y_3y_4)^*)^\omega\}$$

where $(\cdot)^*$ and $(\cdot)^\omega$ denote, respectively, the finite and infinite repetition of the respective string. Furthermore, the sets of 1-long and 2-long dominos obtained from $\mathcal{B}_V$ via (6.1d) are given by

$$D_1 = Y \quad \text{and} \quad D_2 = \{\diamond y_1,\ y_1y_2,\ y_1y_4,\ y_2y_3,\ y_3y_2,\ y_3y_4,\ y_4y_3\}.$$

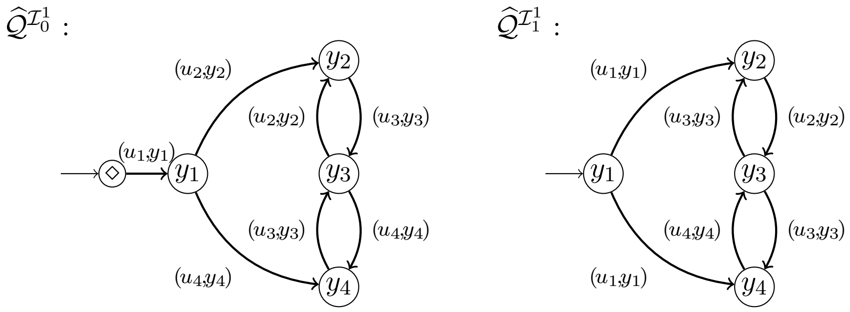

Figure 6.6.: $\mathcal{I}_0^1$- and $\mathcal{I}_1^1$-abstract state machines of $\mathcal{Q}$ in (6.15).

To play the domino-game from Section 2.8 for $l = 1$, i.e., with dominos from the set D_2, we have to pick $\diamond y_1$ as the initial domino and append dominos such that the last element of the first matches the first element of the second domino. It is easy to see that in this example every such combination of dominos yields a sequence contained in $\mathcal{B}_V$. Hence, $\Sigma_V = (\mathbb{N}_0, V, \mathcal{B}_V)$ is asynchronously 1-complete and therefore also asynchronously 2-complete.

Choosing $l = 1$

Using $\mathcal{Q}$ in (6.15) we can construct the $\mathcal{I}_0^1$- and $\mathcal{I}_1^1$-abstract state machines of $\mathcal{Q}$ using Definition 6.3 whose transition structure is depicted in Figure 6.6. Using $\mathcal{I}_0^1 = [-1, -1]$ we consider the immediate past of a state and obtain the following properties of $\mathcal{Q}$ w.r.t $\mathcal{I}_0^1$.

(A1) $\mathcal{Q}$ is *not* state-based asynchronously 1-complete w.r.t. $\mathcal{I}_0^1$:
To see that (6.11) does not hold we choose x_2 and $y_1 y_4 \in D_2$ and observe that $y_1 \in \mathrm{E}^{[-1,-1]}(x_2)$ but $y_1 y_4 \notin \mathrm{E}^{[-1,0]}(x_2)$.

(A2) $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_0^1$ (as this always holds).

Now using $\mathcal{I}_1^1 = [0, 0]$ we consider the current output values of a state and obtain the following properties of $\mathcal{Q}$ w.r.t $\mathcal{I}_1^1$.

(B1) $\mathcal{Q}$ is *not* state-based asynchronously 1-complete w.r.t. $\mathcal{I}_1^1$:
To see that (6.11) does not hold, we choose x_1 and $y_1 y_4 \in D_2$ and observe that $y_1 \in \mathrm{E}^{[0,0]}(x_1)$ but $y_1 y_4 \notin \mathrm{E}^{[0,1]}(x_1)$.

(B2) $\mathcal{Q}$ is future-unique w.r.t. $\mathcal{I}_1^1$:
It is easy to see that $\mathcal{Q}$ is output deterministic. Using (6.6) this immediately implies that $\mathcal{Q}$ is future-unique w.r.t. $\mathcal{I}_1^1$.

Using (A2) and (B2), we know from Theorem 6.10 (i) that the relations

$$\begin{aligned}\mathcal{R}^{\mathcal{I}_0^1} :=& \{(x_1,\diamond)\} \cup \{(x_2,y_1),(x_2,y_3)\} \cup \{(x_3,y_2),(x_3,y_4)\} \\ & \cup \{(x_4,y_1),(x_4,y_3)\} \cup \{(x_5,\diamond)\} \text{ and}\end{aligned} \tag{6.16a}$$

$$\mathcal{R}^{\mathcal{I}_1^1} := \{(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4),(x_5,y_1)\} \tag{6.16b}$$

are simulation relations from $\mathcal{Q}$ to $\widehat{\mathcal{Q}}^{\mathcal{I}_0^1}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$, respectively. It should be noted that every state $x_i \in X$ is related via $\mathcal{R}^{\mathcal{I}_1^1}$ to its unique output $\{y_j\} = \mathrm{H}_\delta(x_i)$, while $x_i \in X$ is related via $\mathcal{R}^{\mathcal{I}_0^1}$ to all possible output events $\mathcal{Q}$ might produce immediately *before* reaching x_i, i.e., the set of y-labels of all incoming transitions.

Using (A1) and (B1) we know from Theorem 6.10 (ii), that both relations are not a bisimilation relation between $\mathcal{Q}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_0^1}$ (resp. $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$). This can be observed from Figure 6.6 by choosing $(x_2,y_1) \in \mathcal{R}^{\mathcal{I}_0^1}$ and $(y_1,(u_4,y_4),y_4) \in \widehat{\delta}^{\mathcal{I}_0^1}$ and observing that x_2 has no outgoing transition labeled by y_4. Similarly, we can choose $(x_1,y_1) \in \mathcal{R}^{\mathcal{I}_1^1}$ and $(y_1,(u_1,y_1),y_4) \in \widehat{\delta}^{\mathcal{I}_1^1}$ and observe that there actually exists an outgoing transition in x_1 labeled by (u_1,y_1) but this transition reaches state x_2 which is not related to y_4 via $\mathcal{R}^{\mathcal{I}_1^1}$.

Choosing $l = 2$

Increasing l and constructing the $\mathcal{I}_0^2$- and $\mathcal{I}_2^2$-abstract state machines of $\mathcal{Q}$ using Definition 6.3 yields the state machines $\widehat{\mathcal{Q}}^{\mathcal{I}_0^2}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_2^2}$ whose transition structure is depicted in Figure 6.7. It is interesting to note that using more information from the past, i.e., using $\mathcal{I}_0^2 = [-2,-1]$ instead of $\mathcal{I}_0^1 = [-1,-1]$, does not render $\mathcal{Q}$ state-based asynchronously l-complete.

(C1) $\mathcal{Q}$ is *not* state-based asynchronously 2-complete w.r.t. $\mathcal{I}_0^2$:
To see that (6.11) does not hold, we choose x_2 and $\diamond y_1 y_4 \in D_3$ and observe that $\diamond y_1 \in \mathrm{E}^{[-2,-1]}(x_2)$ but $\diamond y_1 y_4 \notin \mathrm{E}^{[-2,0]}(x_2)$.

(C2) $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_0^2$ (as this always holds).

Contrary, using more information from the future, i.e., using $\mathcal{I}_2^2 = [0,1]$ instead of $\mathcal{I}_1^1 = [0,0]$, renders $\mathcal{Q}$ state-based asynchronously l-complete. However, in this case the future uniqueness-property is lost.

(D1) $\mathcal{Q}$ is state-based asynchronously 2-complete w.r.t. $\mathcal{I}_2^2$:
Using more future information actually resolves the ambiguity from $\mathcal{I}_1^1$. E.g., choosing x_1 we can only pick $y_1 y_2 y_3 \in D_3$ to obtain $y_1 y_2 \in \mathrm{E}^{[0,1]}(x_1)$, obviously implying $y_1 y_2 y_3 \in \mathrm{E}^{[0,2]}(x_1)$.

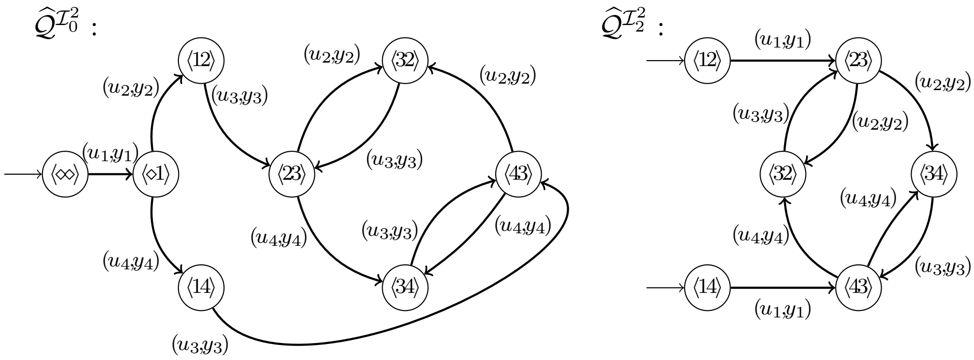

Figure 6.7.: $\mathcal{I}_0^2$- and $\mathcal{I}_2^2$-abstract state machines of $\mathcal{Q}$ in (6.15), where $\langle ij \rangle := y_i y_j$.

(D2) $\mathcal{Q}$ is *not* future-unique w.r.t. $\mathcal{I}_2^2$:
To see that (6.6) does not hold pick x_2 and observe that $y_2 y_3 y_2,\ y_2 y_3 y_4 \in \mathrm{E}^{\mathcal{I}_2^2}(x_1)$ but obviously $y_2 y_3 y_2 \neq y_2 y_3 y_4$.

Using Theorem 6.10 we can now construct relations $\mathcal{R}^{\mathcal{I}_0^2}$ and $\mathcal{R}^{\mathcal{I}_2^2}$ analogously to the ones for $l = 1$ in (6.16). However, now (C1)-(D2) imply that

$$\mathcal{R}^{\mathcal{I}_0^2} \in \mathfrak{S}_V(\mathcal{Q}, \widehat{\mathcal{Q}}^{\mathcal{I}_0^2}) \text{ but } \left(\mathcal{R}^{\mathcal{I}_0^2}\right)^{-1} \notin \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}_0^2}, \mathcal{Q}) \text{ and}$$
$$\left(\mathcal{R}^{\mathcal{I}_2^2}\right)^{-1} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}_2^2}, \mathcal{Q}) \text{ but } \mathcal{R}^{\mathcal{I}_2^2} \notin \mathfrak{S}_V(\mathcal{Q}, \widehat{\mathcal{Q}}^{\mathcal{I}_2^2}).$$

To see that $\mathcal{R}^{\mathcal{I}_2^2}$ is not a simulation relation from $\mathcal{Q}$ to $\widehat{\mathcal{Q}}^{\mathcal{I}_2^2}$ pick $(x_3, y_3 y_4) \in \mathcal{R}^{\mathcal{I}_2^2}$ and $(x_3, u_3, y_3, x_2) \in \delta$ and observe that $y_3 y_4$ does not have an outgoing transition labeled by (u_3, y_3).

Choosing $l > 2$

First recall from (D2) that $\mathcal{Q}$ is not future unique for $\mathcal{I}_2^2$. Using (6.6) this implies that for any interval $\mathcal{I}_m^l$ with $m - 1 \geq 2$ (i.e., any interval with two or more future values) the property of future uniqueness does not hold.

In terms of state-based asynchronous l-completeness the problem is completely opposite. If we use $m - 1 < 2$ (implying future uniqueness of $\mathcal{Q}$ w.r.t. $\mathcal{I}_m^l$ from (A2) and (C2)) $\mathcal{Q}$ can never be state-based asynchronously l-complete for any l as the ambiguity for attaching dominos cannot be resolved by further knowledge about the past. In this case the counterexamples from (A1) and (C1) can always be reused by pre-appending the considered strings by an appropriate number of diamonds. It is rather necessary to look at least two steps into the future, i.e., pick $m - 1 \geq 2$, to resolve this ambiguity as shown in (D1).

Concluding the above discussion there obviously exists no l and m s.t. $\mathcal{Q}$ in (6.15) is both state-based asynchronously l-complete and future unique w.r.t. $\mathcal{I}_m^l$. Therefore, increasing l and m will never result in a bisimilar abstraction of $\mathcal{Q}$ in (6.15).

7. Quotient-based Abstractions with Increasing Precision

As already briefly discussed in Chapter 4, the construction of QBA from a possibly continuous system typically starts by partitioning the state space X into a finite set of equivalence classes which are used as discrete outputs Y when constructing the external discrete event model of the original system. In the considered modeling framework the latter is given by Σ_S, as depicted in Figure 4.3. As Y is chosen to be the set of equivalence classes of a partition on the state space and an output $y \in Y$ is assumed to be directly generated by states rather then transitions, the state machine $\mathcal{Q}$ induced by Σ_S is naturally output deterministic, i.e., can be equivalently represented by a transition system $\mathcal{T}$. This explains why transition systems are commonly used to model the "original" system when using QBA.

The calculation of a suitable set of output symbols Y is typically done iteratively. First, an initial partition of the state space is chosen, e.g., induced by the specification that should be fulfilled by a subsequently designed controller. Then a refinement algorithm is run which terminates if the partition allows to construct a QBA $\widehat{\mathcal{Q}}^{\triangledown}$ which is bisimilar to $\mathcal{Q}$. Hence, with every step, the partition of the state space is refined until it is tight enough for controller synthesis or the algorithm terminates. Therefore, the construction of Y is already part of the abstraction process when using QBA.

Conceptually, there is a strong correspondence of this iterative calculation of the output event set Y in the construction of QBA and the increasing of the parameter l when constructing realizations of SAlCA as outlined in Chapter 6. Recall that in the setting for SAlCA the external event set Y is predefined and only used as the abstract state set of the abstract state machine $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ if $l = 1$ (and $V = Y$). However, when increasing l, the number of abstract states is increased, resulting in a finer cover of the state space, hence in a tighter abstraction.

Therefore, the key in understanding the formal and conceptual differences between QBA and SAlCA lies in the incorporation of the iterative calculation of Y into the abstraction process of QBA. This idea is formalized in Section 7.1 resulting in a chain of abstract SM $\widehat{\mathcal{Q}}^{l\triangledown}$, where the lth abstraction results from the lth step of the partition refinement algorithm. Thereafter, an alternative way to construct $\widehat{\mathcal{Q}}^{l\triangledown}$ is proposed in Section 7.2 to provide a formal basis for the comparison of SAlCA and QBA.

7.1. Incorporating the Partition Refinement Algorithm

In Chapter 6 it was shown how to construct realizations of SAlCA with increasing precision from a SM $\mathcal{Q}$ with predefined external event set W as given in (6.1). Choosing $V = Y$ in (6.1b) results in abstract SM $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ whose state space is given by l-long strings of output events Y. Hence, the "weakest" abstraction uses Y directly as its state space. By increasing l, the number of abstract states is increased, resulting in a finer cover of the state space.

To compare this cover-refinement incorporated in the construction of SAlCA to the partition refinement step usually done prior to applying QBA, we assume the original system to be modeled by an SM

$$\mathcal{Q} = (X, U \times Y, \delta, X_0) \quad s.t. \quad |Y| < \infty, \tag{7.1}$$

define the initial (finite) partition of the state space by

$$\Phi^1 = \left\{ \mathrm{H}_\delta^{-1}(\widehat{y}) \middle| \widehat{y} \in 2^Y \right\} \tag{7.2}$$

and run the re-partitioning algorithm in [11] starting with (7.2). Using SM instead of TS, we restate this algorithm with slightly modified notation. Using this notation, some necessary properties of the resulting partitions are restated from [11] in Lemma 7.2.

Definition 7.1. *Given (7.1) and $l \in \mathbb{N}$, let*

$$\mathrm{T}_\delta(x) := \left\{ x' \in X \middle| \exists u \in U, y \in Y \;.\; (x, u, y, x') \in \delta \right\} \tag{7.3}$$

be the set of input and output independent post states of a state x. Then the l^{th} partition Φ^l of X w.r.t. Y is iteratively defined by[1]

$$\Phi^1 := \left\{ \mathrm{H}_\delta^{-1}(\widehat{y}) \middle| \widehat{y} \in 2^Y \right\} \tag{7.4a}$$

$$\textit{and } \Phi^l := \;\circ_{Z \in \Phi^{l-1}} \Phi^l_Z \tag{7.4b}$$

$$\textit{s.t. } \Phi^l_Z := \left\{ Z' \cap \mathrm{T}_\delta^{-1}(Z) \middle| Z' \in \Phi^{l-1} \right\} \cup \left\{ Z' \setminus \mathrm{T}_\delta^{-1}(Z) \middle| Z' \in \Phi^{l-1} \right\}. \tag{7.4c}$$

Furthermore, $\widehat{Y}^l$ and $\phi^l : X \to \widehat{Y}^l$ denote the set of equivalence classes and the natural projection map of Φ^l, respectively.

Lemma 7.2. *Given (7.1) and Φ^l as in Definition 7.1 it holds that*

$$\forall Z \in \Phi^l, x, x' \in Z \;.\; \mathrm{H}_\delta(x) = \mathrm{H}_\delta(x'), \tag{7.5a}$$

$$\Phi^l = \left\{ Z \in 2^X \middle| \forall Z' \in \Phi^{l-1} . \left(\left(Z \cap \mathrm{T}_\delta^{-1}(Z') \neq \emptyset \right) \Rightarrow \left(Z \subseteq \mathrm{T}_\delta^{-1}(Z') \right) \right) \right\}, \tag{7.5b}$$

and Φ^l is a fixed point of (7.4) if

$$\forall Z, Z' \in \Phi^l . \left(\left(Z \cap \mathrm{T}_\delta^{-1}(Z') \neq \emptyset \right) \Rightarrow \left(Z \subseteq \mathrm{T}_\delta^{-1}(Z') \right) \right). \tag{7.5c}$$

[1]In (7.4b) the operator $\circ_{a \in A} f_a$ composes all functions f_a with $a \in A$ in any order.

Proof. • Show (7.5a): It follows from [11], Prop.3.9 (i) that

$$\forall Z \in \Phi^l \;.\; \exists Z' \in \Phi^{l-1} \;.\; Z \subseteq Z' \tag{7.6}$$

Now recall from (7.4a) that for all $Z^0 \in \Phi^0$ there exists $V \in 2^Y$ s.t. $Z^0 = \mathrm{H}_\delta^{-1}(V)$ and Φ^0 is a partition. Using (7.6) we obtain $Z \subseteq \mathrm{H}_\delta^{-1}(V)$, what proves the statement.
• Show (7.5b): If follows from [11], Prop.3.9(v) that

$$\forall Z' \in \Phi^{l-1}, Z \in \Phi^l \;.\; \begin{pmatrix} Z \cap \mathrm{T}_\delta^{-1}(Z') = \emptyset \\ \vee Z \subseteq \mathrm{T}_\delta^{-1}(Z') \end{pmatrix}$$

Using that $A \Rightarrow B$ is logical equivalent to $\neg A \vee B$ and rewriting the previous statement into set-notation gives (7.5b).
• Show (7.5c): It follows from [11], Prop.3.10 (iii) that Φ^l is a fixed point of (7.4) if $\Phi^{l+1} = \Phi^l$. With this (7.5c) follows immediately from (7.5b). □

Using the set of equivalence classes $\widehat{Y}^l$ and the natural projection map $\phi^l : X \to \widehat{Y}^l$ of Φ^l from Definition 7.1 we can define a QBA for every l by first constructing an output deterministic version of $\mathcal{Q}$.

Definition 7.3. *Given* (7.1) *and* Φ^l *as in Definition 7.1, the output deterministic version of* $\mathcal{Q}$ *w.r.t.* Φ^l *is defined by* $\mathcal{Q}^l = (X, U \times \widehat{Y}^l, \delta^l, X_0)$ *s.t.*

$$\delta^l := \left\{ (x, u, \widehat{y}, x') \middle| \begin{pmatrix} \widehat{y} = \phi^l(x) \\ \wedge \exists y \in Y \;.\; (x, u, y, x') \in \delta \end{pmatrix} \right\}. \tag{7.7}$$

Furthermore, $\widehat{\mathcal{Q}}^{l\triangledown} = (\widehat{X}^{l\triangledown}, U \times \widehat{Y}^l, \widehat{\delta}^{l\triangledown}, \widehat{X}_0^{l\triangledown})$ *denotes the quotient state machine of* $\mathcal{Q}^l$ *as defined in Theorem 3.8.*

It is a well known result in the construction of QBA that the refinement algorithm in Definition 7.1 terminates if the resulting partition allows for the construction of $\mathcal{Q}^l$ and $\widehat{\mathcal{Q}}^{l\triangledown}$ s.t. both models are bisimilar. As this is a classical result from the literature on QBA we omit the proof and only restated the result in our notation.

Theorem 7.4 (e.g. [67], Thm. 4.18)**.** *Given* $\mathcal{Q}^l$ *and* $\widehat{\mathcal{Q}}^{l\triangledown}$ *as in Definition 7.3, let*

$$\mathcal{R} = \left\{ (x, \widehat{x}) \in \left(X \times \widehat{X}^{l\triangledown} \right) \middle| \widehat{x} = \phi^l(x) \right\} \tag{7.8}$$

be a relation. Then
(i) $\mathcal{R} \in \mathfrak{S}_{U \times \widehat{Y}^l}(\mathcal{Q}^l, \widehat{\mathcal{Q}}^{l\triangledown})$ *and*
(ii) $\mathcal{R}^{-1} \in \mathfrak{S}_{\widehat{Y}^l}(\widehat{\mathcal{Q}}^{l\triangledown}, \mathcal{Q}^l) \Leftrightarrow \Phi^l$ *is a fixed-point of* (7.4).

7.2. An Alternative Construction of QBA

Recall from Chapter 6 that the abstract SM $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ outputs events in the set Y while the quotient SM $\widehat{\mathcal{Q}}^{l\triangledown}$ outputs events in the refined set $\widehat{Y}^l$. However, to formally compare both SM using simulation relations as defined in Chapter 5 both models must have the same output event set. We therefore propose a different quotient SM in this section which is directly obtained from $\mathcal{Q}$ and Φ^l and has the output event set Y.

Definition 7.5. *Given (7.1) and Φ^l as in Definition 7.1, the l-th quotient state machine of $\mathcal{Q}$ is defined by $\widehat{\mathcal{Q}}^{l\blacktriangledown} = (\widehat{X}^{l\blacktriangledown}, U \times Y, \widehat{\delta}^{l\blacktriangledown}, \widehat{X}_0^{l\blacktriangledown})$ s.t.*

$$\widehat{X}^{l\blacktriangledown} = \widehat{Y}^l, \tag{7.9a}$$

$$\widehat{X}_0^{l\blacktriangledown} = \left\{\widehat{y} \in \widehat{Y}^l \middle| \exists x \in X_0 \,.\, \widehat{y} = \phi^l(x)\right\}, \quad \textit{and} \tag{7.9b}$$

$$\widehat{\delta}^{l\blacktriangledown} = \left\{(\widehat{x}, u, y, \widehat{x}') \middle| \exists x, x' \in X \,.\, \begin{pmatrix} \widehat{x} = \phi^l(x) \\ \wedge \widehat{x}' = \phi^l(x') \\ \wedge (x, u, y, x') \in \delta \end{pmatrix}\right\}. \tag{7.9c}$$

To show that the construction in Definition 7.5 is sound, we show in the following theorem that the "actual" QBA $\widehat{\mathcal{Q}}^{l\triangledown}$ can be constructed from $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ using a simple re-definition of the output space. The resulting construction of quotient state machines with increasing precision is summarized in Figure 7.1. While the solid line depicts the construction of $\widehat{\mathcal{Q}}^{l\triangledown}$ via $\mathcal{Q}^l$ from Section 7.1, the alternative construction using Definition 7.5 is depicted by a dashed line.

Theorem 7.6. *Given $\widehat{\mathcal{Q}}^{l\triangledown}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ as in Definition 7.3 and Definition 7.5, respectively, it holds that*

$$\widehat{X}^{l\triangledown} = \widehat{X}^{l\blacktriangledown}, \quad \widehat{X}_0^{l\triangledown} = \widehat{X}_0^{l\blacktriangledown} \quad \textit{and} \quad \widehat{\delta}^{l\triangledown} = \left\{(\widehat{x}, u, \widehat{x}, \widehat{x}') \middle| \exists y \in Y \,.\, (\widehat{x}, u, y, \widehat{x}') \in \widehat{\delta}^{l\blacktriangledown}\right\}.$$

Proof. See Section A.2.1 on page 146. □

Using Theorem 7.6 it seems natural to recover the similarity results in Theorem 7.4 for $\mathcal{Q}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$. However, there is a small technicality arising from running the re-partitioning algorithm on SM instead of TS, spoiling this result for bisimilarity.

Intuitively, one step in (7.4) results in a clustering of states with identical post-cells. Hence, a fixed point of (7.4) is obtained if all states in one cell have the same post-cell(s). Using $\mathcal{Q}$ instead of $\mathcal{T}$, post states $\tilde{x}$ of a state x (calculated using T in (7.3)) might depend on the chosen output event $y \in \mathrm{H}_\delta(x)$. E.g., knowing that two states $x, x' \in \mathrm{H}_\delta^{-1}(\{y_1, y_2\})$ have the same post cell does not imply that there exist transitions from x and x' to this cell for both output events y_1 and y_2. However, this is obviously true if $\mathcal{Q}$ has separable dynamics, i.e., if (3.2d) holds.

$$\begin{tikzcd}
 & \mathcal{Q} \arrow[rr, dashed, "\text{Definition 7.5}"] \arrow[dd, "\text{Definition 7.3}"] & & \widehat{\mathcal{Q}}^{l\blacktriangledown} \arrow[dd, dashed, "\text{Theorem 7.6}"'] \\
\Phi^l \arrow[r] \arrow[ur, bend left] & {} & & \\
 & \mathcal{Q}^l \arrow[rr, "\text{Theorem 3.8}"'] & & \widehat{\mathcal{Q}}^{l\triangledown}
\end{tikzcd}$$

Figure 7.1.: The solid line illustrate the construction of the quotient SM $\widehat{\mathcal{Q}}^{l\triangledown}$ from the output deterministic SM $\mathcal{Q}^l$ using QBA. The dashed line illustrates the construction of $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ from $\mathcal{Q}$ directly.

Theorem 7.7. *Given* (7.1), $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ *as in Definition 7.5 and* $\mathcal{R}$ *as in* (7.8), *it holds that*

(i) $\mathcal{R} \in \mathfrak{S}_{U\times Y}(\mathcal{Q}, \widehat{\mathcal{Q}}^{l\blacktriangledown})$ *and*

(ii) (3.2d) *holds for* $\mathcal{Q} \Rightarrow \left(\mathcal{R}^{-1} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{l\blacktriangledown}, \mathcal{Q}) \Leftrightarrow \Phi^l \text{ is a fixed-point of } (7.4)\right)$.

Proof. See Section A.2.2 on page 146. □

It is interesting to observe from the proof of Theorem 7.7 that $\mathcal{R}^{-1} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{l\blacktriangledown}, \mathcal{Q})$ always implies that Φ^l is a fixed-point of (7.4) while the other direction requires $\mathcal{Q}$ to have separable dynamics. Hence, if the latter is not true, it is not guarantied that the quotient state machine $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ obtained from the final partition of the fixed-point algorithm is bisimilar to $\mathcal{Q}$. However, if a fixed-point of (7.4) is reached, no tighter abstraction can be generated using QBA. This observation is supported by the following obvious result on the ordering of quotient state machines based on l.

Theorem 7.8. *Given* (7.1) *and* $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ *as in Definition 7.3, let*

$$\mathcal{R} = \left\{ (\widehat{x}_{l+1}, \widehat{x}_l) \in \left(\widehat{X}^{l+1\blacktriangledown} \times \widehat{X}^{l\blacktriangledown}\right) \middle| \left(\phi^{l+1}\right)^{-1}(\widehat{x}_{l+1}) \subseteq \left(\phi^l\right)^{-1}(\widehat{x}_l) \right\} \tag{7.10}$$

be a relation. Then

(i) $\mathcal{R} \in \mathfrak{S}_{U\times Y}(\widehat{\mathcal{Q}}^{l+1\blacktriangledown}, \widehat{\mathcal{Q}}^{l\blacktriangledown})$ *and*

(ii) $\mathcal{R}^{-1} \in \mathfrak{S}_{U\times Y}(\widehat{\mathcal{Q}}^{l\blacktriangledown}, \widehat{\mathcal{Q}}^{l+1\blacktriangledown}) \Leftrightarrow \Phi^l$ *is a fixed-point of* (7.4).

Proof. See Section A.2.3 on page 148. □

This implies that a bisimilar quotient state machine $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ of $\mathcal{Q}$ can only be constructed using Definition 7.5 if $\mathcal{Q}$ has separable dynamics and (7.4) terminates. We conclude this section by giving two different sensing scenarios for systems with a predefined output space Y which result in a SM $\mathcal{Q}$ modeling the external discrete event system Σ_S which does/does not have separable dynamics.

Example 7.1. Consider a system where each output $y_i \in Y$ is generated by a different sensor and the sensing regions of different sensors overlap. Then $\widehat{y} = \{y_1, y_2\}$ corresponds to the situation where both sensor 1 and sensor 2 are active. In this case, all possible outputs are generated simultaneously, solely based on the current state of the system. It is therefore unintuitive to assume that the system considers only one of them and changes its dynamics according to this choice. The system naturally fulfills (3.2d) and the output event set could simply be redefined to $\widehat{Y} := 2^Y$. ◁

Example 7.2. Consider a system where all measurements are generated by a single sensor with measurement uncertainties. Then $\{y_1, y_2\} = \mathrm{H}_\delta(x)$ technically means that the sensor will either output y_1 or y_2 if the system is in state x, but not both. Hence, y_1 and y_2 cannot be measured simultaneously. In this scenario, $\mathcal{Q}$ must not have separable dynamics, as based on the *single* obtained measurement value, the dynamics of the system might indeed vary. Furthermore, a redefinition of the output event set to $\widehat{Y} := 2^Y$ cannot be practically motivated as the newly defined output $\widehat{y} = \{y_1, y_2\}$ cannot be detected from the outputs of the system. ◁

7.3. Example

We conclude this chapter by revisiting the example in Section 6.5. In particular, we discuss the construction of the l-th quotient state machine of the SM $\mathcal{Q}$ in (6.15) depicted in Figure 6.5. Recall that $\mathcal{Q}$ in (6.15) is output deterministic. As discussed in Chapter 3 this implies that (3.2d) holds, hence $\mathcal{Q}$ has separable dynamics.

Using (7.4) the first and second partition of X w.r.t. Y are given by

$$\Phi^1 = \{\{x_1, x_5\}, \{x_2\}, \{x_3\}, \{x_4\}\} \text{ and} \tag{7.11a}$$
$$\Phi^2 = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}\} \tag{7.11b}$$

with

$$\widehat{Y}^1 = \{\{y_1\}, \{y_2\}, \{y_3\}, \{y_4\}\} \text{ and} \tag{7.12a}$$
$$\widehat{Y}^2 = \{\{y_1y_2\}, \{y_2y_3\}, \{y_3y_2, y_3y_4\}, \{y_4y_3\}, \{y_1y_4\}\}\,. \tag{7.12b}$$

Using (7.5c) in Lemma 7.2 we have the following observations.

(E1) Φ^1 is *not* a fixed-point of (7.4):
To see that (7.5c) does not hold, pick $x_1, x_5 \in \mathrm{H}^{-1}(\{y_1\})$ and observe that $x_1 \in \mathrm{T}_\delta^{-1}(\{y_2\})$ and $x_5 \notin \mathrm{T}_\delta^{-1}(\{y_2\})$, hence $\{y_1\} \cap \mathrm{T}_\delta^{-1}(\{y_2\}) \neq \emptyset$ but $\{y_1\} \not\subseteq \mathrm{T}_\delta^{-1}(\{y_2\})$.

(E2) Φ^2 is a fixed-point of (7.4):
As all cells of Φ^2 are singletons (7.5c) trivially holds.

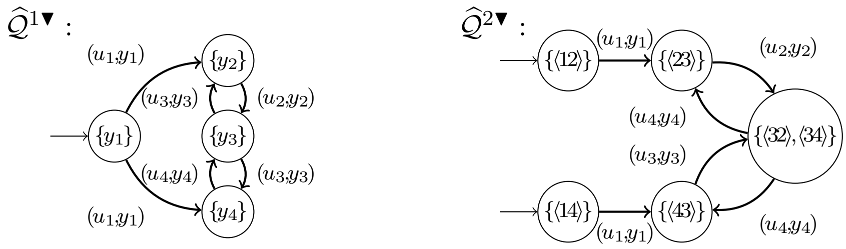

Figure 7.2.: First and second quotient state machines of $\mathcal{Q}$ in (6.15), where $\langle ij\rangle := y_iy_j$.

Now constructing the first and second quotient state machine of $\mathcal{Q}$ using Definition 7.5 yields the SM depicted in Figure 7.2. Using Theorem 7.7 (i) we know that

$$\mathcal{R}^{1\blacktriangledown} :=\{(x_1,\{y_1\}),(x_2,\{y_2\}),(x_3,\{y_3\}),(x_4,\{y_4\}),(x_5,\{y_1\})\} \text{ and} \tag{7.13a}$$

$$\mathcal{R}^{2\blacktriangledown} :=\{(x_1,\{y_1y_2\}),(x_2,\{y_2y_3\}),(x_3,\{y_3y_2,y_3y_4\}),(x_4,\{y_4y_3\}),(x_5,\{y_1y_4\})\} \tag{7.13b}$$

are simulation relations from $\mathcal{Q}$ to $\widehat{\mathcal{Q}}^{1\blacktriangledown}$ and $\widehat{\mathcal{Q}}^{2\blacktriangledown}$, respectively. However, Theorem 7.7 (ii) implies that only $\mathcal{R}^{2\blacktriangledown}$ is a bisimulation relation between $\mathcal{Q}$ and $\widehat{\mathcal{Q}}^{2\blacktriangledown}$. To see that this is not true for $\mathcal{R}^{1\blacktriangledown}$, we choose $(x_1,\{y_1\}) \in \mathcal{R}^{1\blacktriangledown}$ and $(\{y_1\},(u_1,y_1),\{y_4\}) \in \widehat{\delta}^{1\blacktriangledown}$ and observe that there exists only one outgoing transition in x_1 labeled by (u_1,y_1) reaching x_2, which is not related to $\{y_4\}$ via $\mathcal{R}^{1\blacktriangledown}$.

8. Comparison of SAlCA and QBA

Recall from Chapter 4 that under Assumption 1-2 the step-by-step evolution of the "original" system can be modeled by the SM $\mathcal{Q}$ in (7.1). It was furthermore shown in Chapter 6 and Chapter 7 that the abstract FSM resulting from SAlCA and QBA can be directly obtained from $\mathcal{Q}$. Hence, $\mathcal{Q}$ can be used as a common basis for constructing $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$. However, as outlined at various points of the preceding part of this thesis, the assumptions about the external output event set Y contained in Assumption 1 are fundamentally different in both settings. We therefore obtain a comparison for both viewpoints in this chapter.

First, in Section 8.1, we take the viewpoint of SAlCA and assume a predefined output space Y and apply both the constructions of SAlCA and QBA to $\mathcal{Q}$. Surprisingly, we will show that this results in generally incomparable abstract state machines. We will therefore derive necessary and sufficient conditions in terms of properties on $\mathcal{Q}$ which ensure that both abstractions become bisimilar. These conditions will turn out to be rather restrictive.

Thereafter, in Section 8.2, we take the viewpoint of QBA and assume that the output event set can be arbitrary chosen. Hence, we assume that the refinement algorithm in Definition 7.1 is run for l steps first, using an arbitrary initial partition Φ^1. This results in an output deterministic state machine $\mathcal{Q}$ which is used as a basis to construct both the quotient and the $\mathcal{I}^1_1$-abstract state machine. In this special case, we can show that both abstractions coincide. Hence, in this second set up, the necessary and sufficient conditions derived in Section 8.1 are automatically fulfilled.

8.1. The viewpoint of SAlCA

Recall that the construction of the abstract SM $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ in Chapter 6 is based on (6.1) (partially resulting from Assumption 2), while the construction of the quotient SM $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ in Chapter 7 is based on (7.1) (resulting from Assumption 1). To compare both methods we make the following additional assumption.

Assumption 3. *The SM $\mathcal{Q}$ modeling the external discrete event dynamics of $\breve{\Sigma}^{\phi}_{S}$ has separable dynamics.*

Combining (6.1) and (7.1) and using Assumption 3, the SM $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ compared in this section are assumed to be constructed from an SM

$$\mathcal{Q} = (X, U \times Y, \delta, X_0) \quad s.t. \quad (2.23) \text{ and } (3.2d) \text{ holds,} \tag{8.1a}$$

modeling the external discrete event dynamics of $\breve{\Sigma}_S^{\phi}$ as depicted in Figure 4.3. Furthermore, combining (6.1b) with the observation that for QBA only the output set Y is used to construct the states of $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ the set V in (6.1b) is set to

$$V = Y, \text{ i.e., } |Y| < \infty. \tag{8.1b}$$

Redefining the Natural Projection Map of Φ^l

Recall that the abstract state spaces $\widehat{X}^{\mathcal{I}^l_m}$ and $\widehat{X}^{l\blacktriangledown}$ induce a cover (resp. a partition) on the original state space X which is iteratively refined by increasing l. Using Theorem 6.10 and Theorem 7.7 we know that states $x \in X$ are related to abstract states $\widehat{x} \in \widehat{X}^{\mathcal{I}^l_m}$ and $\widehat{x}' \in \widehat{X}^{l\blacktriangledown}$ via $\mathrm{E}^{\mathcal{I}^l_m}$ and ϕ^l, respectively. As the first essential result of our bridging endeavor we now show that these two maps coincide for $m = l$ if Assumption 3 holds.

Proposition 8.1. *Given* (8.1) *and* Φ^l *as in Definition 7.1 it holds that*

$$\Phi^l = \left\{ \left(\mathrm{E}^{\mathcal{I}^l_l}\right)^{-1}(\Upsilon) \middle| \Upsilon \in 2^{(Y)^l} \right\}. \tag{8.2}$$

Proof. See Section A.3.1 at page 149. □

Proposition 8.1 implies that the equivalence classes of Φ^l are given by all sets $\Upsilon \in 2^{(Y)^l}$ of l-long dominos ζ which are consistent with the behavior of $\mathcal{Q}$ and the map $\mathrm{E}^{\mathcal{I}^l_l}$ is the natural projection map of Φ^l taking a state $x \in X$ to its (unique) equivalence class. Using this observation in Definition 7.5 gives the following obvious corollary.

Corollary 8.2. *Given* (7.1) *and* $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ *as in Definition 7.5, it holds that*

$$\widehat{X}^{l\blacktriangledown} = \left\{ \Upsilon \middle| \exists x \in X \,.\, \Upsilon = \mathrm{E}^{\mathcal{I}^l_l}(x) \right\} \tag{8.3a}$$

$$\widehat{X}_0^{l\blacktriangledown} = \left\{ \Upsilon \middle| \exists x \in X_0 \,.\, \Upsilon = \mathrm{E}^{\mathcal{I}^l_l}(x) \right\}, \quad \text{and} \tag{8.3b}$$

$$\widehat{\delta}^{l\blacktriangledown} = \left\{ (\widehat{x}, u, y, \widehat{x}') \middle| \exists x, x' \in X \,.\, \begin{pmatrix} \widehat{x} = \mathrm{E}^{\mathcal{I}^l_l}(x) \\ \wedge \widehat{x}' = \mathrm{E}^{\mathcal{I}^l_l}(x') \\ \wedge (x, u, y, x') \in \delta \end{pmatrix} \right\}. \tag{8.3c}$$

Comparing $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$

Recall that the construction of the $\mathcal{I}_l^l$-abstract SM of $\mathcal{Q}$ using SAlCA in Definition 6.3 is also based on the set $\mathrm{E}^{\mathcal{I}_l^l}$ of corresponding external event sequences. Hence, $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ in Corollary 8.2 is conceptually similar to $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ but does not coincide with the latter. While the abstract states $\widehat{x} \in \widehat{X}^{\mathcal{I}_l^l}$ are strings $\zeta \in (Y)^l$ inducing a cover on the state space X, the abstract states $\widehat{x} \in \widehat{X}^{l\blacktriangledown}$ are sets $\Upsilon \in 2^{(Y)^l}$ inducing a partition on the state space X.

Having this connection between $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ in mind, there are two interesting questions to be asked when comparing them.

(i) Does $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ realize the SAlCA $\widehat{\Sigma}_Y^{l\uparrow} = (\mathbb{N}_0, Y, \widehat{\mathcal{B}}_Y^{l\uparrow})$ (constructed from $\mathcal{Q}$ in (8.1a) as depicted in Figure 6.4) of Σ w.r.t. Y?

(ii) Can we establish an ordering of the SM $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ in terms of simulation relations?

Unfortunately, we will see that non of the above statements is true in general. We will therefore derive necessary and sufficient conditions on the structure of $\mathcal{Q}$ in (8.1a) for those statements to hold. We start by giving the only comparing result that holds in general.

Theorem 8.3. *Given (8.1), $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ as in Definition 7.5 and $\widehat{\Sigma}_Y^{l\uparrow} = (\mathbb{N}_0, Y, \widehat{\mathcal{B}}_Y^{l\uparrow})$ constructed as depicted in Figure 6.4, it holds that $\pi_Y(\mathcal{B}_f(\widehat{\mathcal{Q}}^{l\blacktriangledown})) \subseteq \widehat{\mathcal{B}}_Y^{l\uparrow}$.*

Proof. See Section A.3.2 at page 151. □

As external behavioral inclusion is a necessary condition for the existence of a simulation relation from $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ to $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ (where the external behavior of $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ w.r.t. Y is given by $\widehat{\mathcal{B}}_Y^{l\uparrow}$ from Theorem 6.4) the natural next step is to construct this relation. However, thinking back to the results in Theorem 6.10 (i) and Theorem 7.7 (i) there is not much hope for success, as the existence of a simulation relation from $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ to $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ would imply the existence of a simulation relation from $\mathcal{Q}$ to $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ without the need for future-uniqueness of $\mathcal{Q}$ w.r.t. $\mathcal{I}_l^l$. Not surprisingly, the latter condition will turn out to be necessary and sufficient for the naturally chosen relation from $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ to $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ to be a simulation relation.

For the inverse relation to be a simulation relation from $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ to $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ the following property will turn out to be necessary and sufficient.

Definition 8.4. *Given (8.1) and (6.1d), $\mathcal{Q}$ is* domino consistent *for $l \in \mathbb{N}_0$ if*

$$\forall \zeta \in D_{l+1}, \Upsilon \in 2^{(Y)^l} \,.\, \left(\zeta|_{[0,l-1]} \in \Upsilon \Rightarrow \exists x \in \left(\mathrm{E}^{\mathcal{I}_l^l}\right)^{-1}(\Upsilon) \,.\, \zeta \in \mathrm{E}^{[0,l]}(x)\right) \tag{8.4}$$

Intuitively, domino consistency of $\mathcal{Q}$ implies that whenever a string ζ is part of an abstract state $\widehat{x}$, i.e., $\zeta \in \widehat{x}$, any domino $\zeta' \in D_{l+1}$ that can be attached to ζ in the domino game, i.e., $\zeta'|_{[0,l-1]} = \zeta$, can be attached for this particular abstract state $\widehat{x}$, i.e., there exists a transition from $\widehat{x}$ to $\widehat{x}'$ s.t. $\zeta'|_{[1,l]} \in \widehat{x}'$. Recalling that $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ is constructed to allow all possible moves of the domino game, it becomes intuitively clear why the condition in Definition 8.4 is needed to prove that $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ can simulate $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$.

Theorem 8.5. *Given* (8.1) *and* $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ *and* $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ *as in Definition 6.3 and Definition 7.5, respectively, let*

$$\mathcal{R} = \left\{ (\zeta, \Upsilon) \in \widehat{X}^{\mathcal{I}_l^l} \times \widehat{X}^{l\blacktriangledown} \middle| \zeta \in \Upsilon \right\}. \tag{8.5}$$

Then

(i) $\mathcal{R} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}, \widehat{\mathcal{Q}}^{l\blacktriangledown}) \Leftrightarrow \mathcal{Q}$ *is domino consistent for* l *and*

(ii) $\mathcal{R}^{-1} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{l\blacktriangledown}, \widehat{\mathcal{Q}}^{\mathcal{I}_l^l}) \Leftrightarrow \mathcal{Q}$ *is future unique w.r.t.* $\mathcal{I}_l^l$.

Proof. See Section A.3.3 at page 153. □

Recalling from Proposition 5.4 that the existence of a simulation relation implies behavioral inclusion we can simply combine the results from Theorem 8.3 and Theorem 8.5 (i) to obtain an answer to our first question.

Corollary 8.6. *Given* (8.1), $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ *as in Definition 7.5 and* $\widehat{\Sigma}_Y^{l\uparrow} = (\mathbb{N}_0, Y, \widehat{\mathcal{B}}_Y^{l\uparrow})$ *constructed as depicted in Figure 6.4,* $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ *realizes* $\widehat{\Sigma}_Y^{l\uparrow}$ *w.r.t.* Y *if* $\mathcal{Q}$ *is domino consistent for* l.

Even though, we have only given a sufficient condition in Theorem 8.3 it should be noted that this condition is "almost" necessary in the following sense. The only reason for domino consistency to not be necessary for behavioral equivalence is that for any string $\gamma \in \widehat{\mathcal{B}}_Y^{l\uparrow}$ domino consistency is only required for all cells this string passes through. As in general not every string passes though all cells that contain any of is l-long pieces, domino consistency is only necessary for the cells which are actually passed, i.e., for "almost all" cells.

To wrap up the comparison of $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ we show that future uniqueness of $\mathcal{Q}$ w.r.t. $\mathcal{I}_l^l$ always implies domino consistency of $\mathcal{Q}$.

Lemma 8.7. *Given* (8.1) *and* (6.1d) *it holds that*

$$\mathcal{Q} \text{ is future unique w.r.t. } \mathcal{I}_l^l \Rightarrow \mathcal{Q} \text{ is domino consistent for } l. \tag{8.6}$$

Proof. Using (6.6), future uniqueness of $\mathcal{Q}$ w.r.t. $\mathcal{I}_l^l$ implies that that for all $x \in X$ holds

$$\mathrm{E}^{\mathcal{I}_l^l}(x) \neq \emptyset \Rightarrow |\mathrm{E}^{\mathcal{I}_l^l}(x)| = 1. \tag{8.7}$$

Using (8.3a) this immediately implies $|\widehat{x}| = 1$ for all $\widehat{x} \in \widehat{X}^{l\blacktriangledown}$. Now pick $\zeta \in D_{l+1}$ and $\widehat{x} \in \widehat{X}^{l\blacktriangledown}$ s.t. $\zeta' = \zeta|_{[0,l-1]} \in \widehat{x}$, implying $\widehat{x} = \{\zeta'\}$. As $\zeta \in D_{l+1}$ we know that there exists $(\gamma, \xi) \in \mathcal{B}_{SV}$ and $k \in \mathbb{N}_0$ s.t. $\zeta = \gamma|_{[k,k+l]}$ and therefore $\zeta \in \mathrm{E}^{[0,l]}(\xi(k))$ and $\zeta' \in \mathrm{E}^{\mathcal{I}_l^l}(\xi(k))$. This immediately implies $\xi(k) \in \left(\mathrm{E}^{\mathcal{I}_l^l}\right)^{-1}(\widehat{x})$, what proves the statement. □

As the inverse implication in Lemma 8.7 is generally not true, $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ might actually be a tighter abstraction than $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ if $\mathcal{Q}$ is not future unique w.r.t. $\mathcal{I}_l^l$. However, recall from Theorem 6.10 that in this case $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ does not simulate $\mathcal{Q}$, i.e., might be "too tight" to suitably abstract $\mathcal{Q}$. Hence, assuming future uniqueness of $\mathcal{Q}$ w.r.t. $\mathcal{I}_l^l$ to compare $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ does not exclude interesting scenarios for bisimilarity of the latter two.

Proposition 8.8. *Given* (8.1) *s.t. $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_l^l$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ as in Definition 6.3 and Definition 7.5, respectively, let*

$$\mathcal{R} = \left\{(\zeta, \Upsilon) \in \widehat{X}^{\mathcal{I}_l^l} \times \widehat{X}^{l\blacktriangledown} \middle| \Upsilon = \{\zeta\}\right\}. \tag{8.8}$$

Furthermore, let $\mathcal{R}^{l\uparrow}$ and $\mathcal{R}^{l\blacktriangledown}$ denote the relations defined in (6.12) *and* (7.8)*, respectively. Then it holds that*

(i) $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ *realizes* $\widehat{\Sigma}_Y^{l\uparrow}$ *w.r.t.* Y,

(ii) $\mathcal{R} \in \mathfrak{B}_Y(\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}, \widehat{\mathcal{Q}}^{l\blacktriangledown})$, *hence* $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l} \cong_Y \widehat{\mathcal{Q}}^{l\blacktriangledown}$,

(iii) $\mathcal{R}^{l\uparrow} \in \mathfrak{B}_Y(\mathcal{Q}, \widehat{\mathcal{Q}}^{\mathcal{I}_l^l}) \Leftrightarrow \mathcal{R}^{l\blacktriangledown} \in \mathfrak{B}_Y(\mathcal{Q}, \widehat{\mathcal{Q}}^{l\blacktriangledown})$ *and*

(iv) Φ^l *is a fixed-point of* (7.4) $\Leftrightarrow$ $\mathcal{Q}$ *is state-based asyc. l-compl. w.r.t.* $\mathcal{I}_l^l$.

Proof. See Section A.3.4 on page 156. □

Hence, whenever $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_l^l$ the abstractions $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ are equivalent up to a trivial renaming of states.

Comparing $\widehat{\mathcal{Q}}^{\mathcal{I}_0^l}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$

Up until now we have investigated when $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ is realizing $\widehat{\Sigma}_Y^{l\uparrow}$ and how $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ compares to $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$. However, recall from Theorem 6.4 that choosing $m = 0$, i.e., construction $\widehat{\mathcal{Q}}^{\mathcal{I}_0^l}$ instead of $\widehat{\mathcal{Q}}^{\mathcal{I}_l^l}$ results in the "original" realization of SAlCA. Therefore, we want to conclude our comparison by investigating the connection between $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_0^l}$.

Proposition 8.9. *Given* (8.1) *s.t. $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_l^l$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ and $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ as in Definition 6.3 and Definition 7.5, respectively, it holds that*

(i) $\widehat{\mathcal{Q}}^{l\blacktriangledown} \cong_Y \widehat{\mathcal{Q}}^{\mathcal{I}_l^l} \preceq_Y \widehat{\mathcal{Q}}^{\mathcal{I}_0^l}$ *and*

(ii) $\widehat{\mathcal{Q}}^{l\blacktriangledown} \cong_Y \widehat{\mathcal{Q}}^{\mathcal{I}_l^l} \cong_Y \widehat{\mathcal{Q}}^{\mathcal{I}_0^l}$ *if $\mathcal{Q}$ is state-based asyc. l-compl. w.r.t.* $\mathcal{I}_0^l$.

Proof. Follows from Proposition 8.8 (i), Definition 5.2 and Theorem 6.13. □

It is interesting to note that $\mathcal{Q}$ being state-based asynchronously l-complete w.r.t. $\mathcal{I}_l^l$ does not imply that the latter also holds for $m = 0$ (as shown by the example in Section 6.5). Hence, even if a fixed-point of (7.4) is reached after l-steps and $\mathcal{Q}$ is future-unique w.r.t. $\mathcal{I}_l^l$ this does not imply bisimilarity of $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_0^l}$.

Example

We conclude this section by revisiting the example in Section 6.5 and Section 7.3 to compare the abstractions constructed therein. First observe that the SM $\mathcal{Q}$ depicted in Figure 6.5 is live, reachable and has separable dynamics, i.e., fulfills (8.1). Future uniqueness of $\mathcal{Q}$ w.r.t. $\mathcal{I}_l^l$ was already investigated in Section 6.5 and is given by the properties (B2) and (D2) for $l = 1$ and $l = 2$, respectively. Hence, $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_1^1$ but not w.r.t. $\mathcal{I}_2^2$. Concerning domino consistency, we have the following observations.

(F1) $\mathcal{Q}$ is domino consistent for $l = 1$:
Follows from (D1) and Lemma 8.7.

(F2) $\mathcal{Q}$ is domino consistent for $l = 2$:
Follows from the fact that every 2-long string $\zeta \in D_2$ is only contained in one abstract state $\widehat{x} \in \widehat{X}^{l\blacktriangledown} \subseteq \Upsilon$. Therefore, (8.4) trivially holds.

Now observe that (B2) and Proposition 8.8 implies that $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$ and $\widehat{\mathcal{Q}}^{1\blacktriangledown}$ are identical up to the trivial renaming of states given by $\mathcal{R}$ in (8.8). This is also obvious by investigating Figure 6.6 (right) and Figure 7.2 (left). It can be furthermore observed from (6.16) and (7.13) that

$$\mathcal{R}^{1\blacktriangledown} = \mathcal{R}^{\mathcal{I}_1^1} \circ \mathcal{R}$$

with $\mathcal{R}$ from (8.8). This is actually always true if $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_l^l$ and was used to prove Proposition 8.8 (iii)-(iv).

As $\mathcal{Q}$ is not future unique w.r.t. $\mathcal{I}_2^2$ from (D2) we cannot apply Proposition 8.8 for $l = 2$. However, it follows from (F2) and Corollary 8.6 that $\widehat{\mathcal{Q}}^{2\blacktriangledown}$ realizes the SAlCA $\widehat{\Sigma}_Y^{l\uparrow}$ of Σ w.r.t. Y. This implies that for this particular example using QBA yields a bisimilar abstraction of $\mathcal{Q}$ (from (E2) and Theorem 7.7) which is a realization of the SAlCA $\widehat{\Sigma}_Y^{l\uparrow}$ of Σ w.r.t. Y. However, this realization cannot coincide with any abstract state machine $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ as one of its states is given by a set $\Upsilon \in 2^{(Y)^l}$ with $|\Upsilon| > 1$. In particular, as $\widehat{\mathcal{Q}}^{\mathcal{I}_0^2}$ only simulates $\mathcal{Q}$ but is not bisimilar to the latter (from (C1) and Theorem 6.10) and $\widehat{\mathcal{Q}}^{\mathcal{I}_2^2}$ is only simulated by $\mathcal{Q}$ but not vice versa (from (D2) and Theorem 6.10) we have the following (strict) ordering of abstractions

$$\widehat{\mathcal{Q}}^{\mathcal{I}_2^2} \preceq_Y \widehat{\mathcal{Q}}^{2\blacktriangledown} \preceq_Y \widehat{\mathcal{Q}}^{\mathcal{I}_0^2} \quad \text{with} \quad \widehat{\mathcal{Q}}^{2\blacktriangledown} \cong_Y \mathcal{Q}.$$

8.2. The viewpoint of QBA

In comparison to the previous section we now assume that Y in Assumption 1 can be arbitrarily chosen and outputs are generated based on the current state rather then a transition. In the spirit of Figure 4.3 we therefore assume an arbitrary initial partition Φ^1 on the state space X of $\check{\Sigma}_S$ and run the repartitioning algorithm in Definition 7.1 for

l steps to obtain Y, which is then used to construct $\breve{\Sigma}_S^{\phi}$ and $\mathcal{Q}$ as depicted in Figure 4.3. As Y induces a partition on the state space, $\mathcal{Q}$ is obviously output deterministic. Hence, the abstract and quotient state machines compared in this section are assumed to be constructed from an SM

$$\mathcal{Q} = (X, U \times Y, \delta, X_0) \quad s.t. \quad (2.23), (3.2c) \text{ and } (3.2d) \text{ holds and} \tag{8.9a}$$

$$\forall y \in Y \,.\, \exists x \in X \,.\, \{y\} = \mathrm{H}_\delta(x). \tag{8.9b}$$

Here, (8.9b) holds as Y is assumed to be the set of equivalence classes of a partition of the state space X, i.e., all outputs are "reachable".

As we assume that the partition refinement algorithm is run prior to the construction of abstractions, a further refinement of the cover is not necessary. As $\mathcal{Q}$ is output deterministic, we can furthermore use the "classical" construction of QBA in Theorem 3.8 to obtain $\widehat{\mathcal{Q}}^{\triangledown}$ from $\mathcal{Q}$ using Y both as the set of output events and the set of states of $\widehat{\mathcal{Q}}^{\triangledown}$. To mimic this construction by SAlCA, we use

$$V = Y \quad \text{and} \quad l = 1. \tag{8.9c}$$

Comparing $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$ and $\widehat{\mathcal{Q}}^{\triangledown}$

As we restrict our attention in this section to $l = 1$ and an output deterministic SM $\mathcal{Q}$, $\mathcal{Q}$ is always future unique w.r.t. $\mathcal{I}_1^1$.

Lemma 8.10. *Given (8.9), $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_1^1$.*

Proof. It is shown in (A.20) (see Section A.3.1 on page 149) that $\mathrm{E}^{\mathcal{I}_1^1} = \mathrm{H}_\delta$. Hence, (3.2c) implies for all $x \in X$ that $|\mathrm{E}^{\mathcal{I}_1^1}(x)| = 1$. Hence, (6.6) is trivially fulfilled. □

Lemma 8.10 and Theorem 8.5 already imply bisimilarity of $\widehat{\mathcal{Q}}^{1\blacktriangledown}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$. Moreover, using $\widehat{\mathcal{Q}}^{\triangledown}$ instead of $\widehat{\mathcal{Q}}^{1\blacktriangledown}$ the renaming of states by $\mathcal{R}$ in (8.8) is not required and we obtain a one-to-one matching of $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$ and $\widehat{\mathcal{Q}}^{\triangledown}$.

Theorem 8.11. *Given (8.9) and $\widehat{\mathcal{Q}}^{\triangledown}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$ as defined in Theorem 3.8 and Definition 6.3, respectively, it holds that*

$$\widehat{\mathcal{Q}}^{\triangledown} = \widehat{\mathcal{Q}}^{\mathcal{I}_1^1}.$$

Proof. (8.9b), (6.7a) and (3.5a) immediately imply

$$\widehat{X}^{\mathcal{I}_1^1} = Y = \widehat{X}^{\triangledown} \quad \text{and} \quad \widehat{X}_0^{\mathcal{I}_1^1} = \widehat{X}_0^{\triangledown}.$$

Furthermore, using $l = 1$ the first two lines in (6.7c) evaluate to $\lambda = \lambda$ and are therefore always true. Hence, using $\mathrm{E}^{\mathcal{I}_1^1} = \mathrm{H}_\delta$ (from (A.20) in Section A.3.1 on page 149) (3.5) and (6.7) are also identical, what proves the statement. □

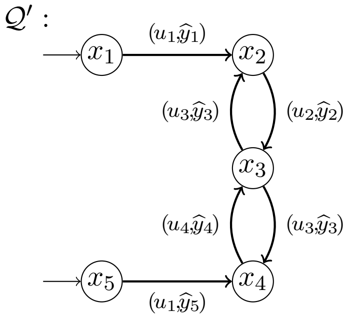

Figure 8.1.: Transition structure of the SM $\mathcal{Q}'$ in (8.10).

Using Lemma 8.10, Theorem 8.11 and Theorem 6.12 we have the following obvious result on the comparison of $\widehat{\mathcal{Q}}^{\triangledown}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_0^l}$.

Corollary 8.12. *Given* (8.9) *and* $\widehat{\mathcal{Q}}^{\triangledown}$ *and* $\widehat{\mathcal{Q}}^{\mathcal{I}_0^1}$ *as defined in Theorem 3.8 and Definition 6.3, respectively, it holds that*
(i) $\widehat{\mathcal{Q}}^{\triangledown} \preceq_Y \widehat{\mathcal{Q}}^{\mathcal{I}_0^1}$ *and*
(ii) $\widehat{\mathcal{Q}}^{\triangledown} \cong_Y \widehat{\mathcal{Q}}^{\mathcal{I}_0^1}$ *if* $\mathcal{Q}$ *is state-based asyc. 1-compl. w.r.t.* $\mathcal{I}_0^1$.

Example

We revisit the example in Section 6.5 and Section 7.3 and adapt it to the setting discussed in this section. For this purpose, assume that Φ^1 in (7.4a) is just an arbitrary partition on the state space X of $\mathcal{Q}$ and Y in (6.15) is the set of equivalence classes of this arbitrarily chosen partition (and not a predefined output event set). Now running the repartitioning algorithm in Definition 7.1 we obtain Φ^2 in (7.11b) as the fixed point of (7.4) (from (F2)). We therefore use $\widehat{Y}^2$ in (7.12b) as the set of considered output events enumerated from 1 to 5 and define

$$\mathcal{Q}' = (X, U \times \widehat{Y}^2, \delta, X_0) \quad s.t. \quad V = \widehat{Y}^2 \tag{8.10}$$

where the transition structure of $\mathcal{Q}'$ is depicted in Figure 8.1. Observe that, in comparison to $\mathcal{Q}$ in Figure 6.5, every state x_i has a unique output $\widehat{y}_i$.

Using Definition 6.3 we can construct the $\mathcal{I}_0^1$- and $\mathcal{I}_1^1$-abstract state machines of $\mathcal{Q}'$ as depicted in Figure 8.2, where the latter coincides with the quotient state machine $\widehat{\mathcal{Q}}^{\triangledown}$ from Theorem 3.8. It is obvious from Figure 8.1 and Figure 8.2 that $\mathcal{Q}'$ and $\widehat{\mathcal{Q}}'^{\mathcal{I}_1^1}$ (resp. $\widehat{\mathcal{Q}}'^{\triangledown}$) are equivalent up to a trivial renaming of states and therefore bisimilar.

However, $\widehat{\mathcal{Q}}'^{\mathcal{I}_1^1}$ (resp. $\widehat{\mathcal{Q}}'^{\triangledown}$) is a tighter abstraction of $\mathcal{Q}'$ then $\widehat{\mathcal{Q}}'^{\mathcal{I}_0^1}$ in terms of simulation relations as $\mathcal{Q}'$ is not state-based asynchronously 1-complete w.r.t. $\mathcal{I}_0^1$. To see that (6.11)

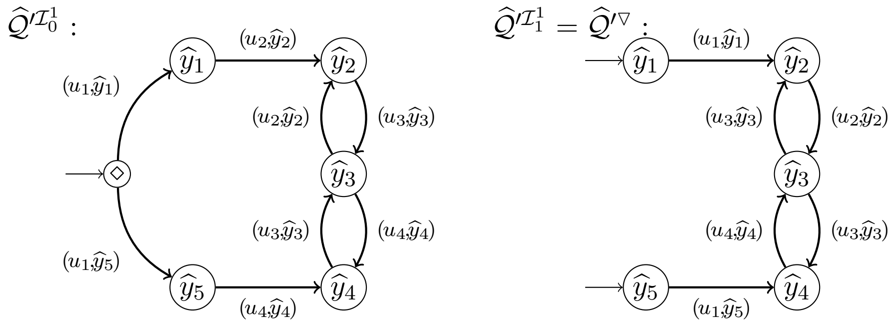

Figure 8.2.: $\mathcal{I}_0^1$- and $\mathcal{I}_1^1$-abstract state machines of $\mathcal{Q}'$ in (8.10).

does not hold, we choose x_1 and $\diamond\widehat{y}_5 \in D_2$ and observe that $\diamond \in \mathrm{E}^{[-1,-1]}(x_1)$ but $\diamond\widehat{y}_5 \notin \mathrm{E}^{[-1,0]}(x_1)$ (where D_2 and $\mathrm{E}()$ are obtained from $\mathcal{Q}'$ rather then from $\mathcal{Q}$).

Summarizing the above observations, we have the following ordering of abstractions

$$\mathcal{Q}' \cong_Y \widehat{\mathcal{Q}}'^{\triangledown} = \widehat{\mathcal{Q}}'^{\mathcal{I}_1^1} \preceq_Y \widehat{\mathcal{Q}}'^{\mathcal{I}_0^1}.$$

8.3. Some Comments on Control and Future Research

In Section 8.1-8.2 we have compared finite state machine abstractions resulting from SAlCA and QBA using the notion of simulation relations. The construction of those abstractions is usually motivated by a control problem involving a finite set of output symbols as discussed in Chapter 1. To use the obtained comparing results, it would therefore be interesting to investigate the usability of the constructed abstractions for control purposes. Unfortunately, given the different settings of SAlCA and QBA, the controller synthesis techniques applied in the literature also differ significantly, as they are usually tailored to the respective setting. We are therefore not aiming at a profound comparison of the latter, but rather emphasize some observations from the running example discussed in Chapter 6-8.

In the literature on QBA so called *alternating simulation relations* are used to evaluate if an abstraction is suitable for control (see e.g. [67, Def.4.19] for a formal definition). It is interesting to note that for any choice of l and m the inverse relation $\mathcal{R}^{-1}$ of $\mathcal{R}$ in Theorem 6.10 (resp. Theorem 7.7) is an alternating simulation relation from $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ (resp. $\widehat{\mathcal{Q}}^{l\blacktriangledown}$) to $\mathcal{Q}$ iff $\mathcal{R}$ is a simulation relation from $\mathcal{Q}$ to $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ (resp. $\widehat{\mathcal{Q}}^{l\blacktriangledown}$) and

$$\forall (x, \widehat{x}) \in \mathcal{R} \;.\; \mathrm{U}_{\widehat{\delta}}(\widehat{x}) \subseteq \mathrm{U}_{\delta}(x). \tag{8.11}$$

Hence, the abstraction must simulate $\mathcal{Q}$ to be suitable for controller synthesis in the setting of QBA.

In our running example we have shown in Section 6.5 that $\mathcal{R}^{\mathcal{I}_2^2}$ is not a simulation relation from $\mathcal{Q}$ to $\widehat{\mathcal{Q}}^{\mathcal{I}_2^2}$ (depicted in Figure 6.7 (right)). Intuitively, this is due to the fact that the abstraction has to "guess" non-deterministically when observing y_2 to which state to move to be able to "follow" the future evolution of $\mathcal{Q}$. Interestingly, it can be observed in Figure 6.6 (right) that the abstraction $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$ also needs to decide to either move to y_2 or y_4 from y_1 when observing the output y_1. However, it was shown in Section 6.5 that $\mathcal{R}^{\mathcal{I}_1^1}$ is a simulation relation from $\mathcal{Q}$ to $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$ and it can be easily observed that (8.11) holds for $\mathcal{R}^{\mathcal{I}_1^1}$ in (6.16). Hence, $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$ is suitable for control in terms of alternating simulation relations, while $\widehat{\mathcal{Q}}^{\mathcal{I}_2^2}$ is not.

Intuitively, this is due to the fact that, using simulation relations, it is implicitly assumed that the abstraction "knows" to which state the original system moves. Therefore, $\widehat{\mathcal{Q}}^{\mathcal{I}_1^1}$ can observe if $\mathcal{Q}$ moves to x_2 or x_4 and can then pick the "right" state, i.e., the related one. Contrary, the states y_3y_2 and y_3y_4 of $\widehat{\mathcal{Q}}^{\mathcal{I}_2^2}$ are related to the same state x_3. Therefore, knowing that $\mathcal{Q}$ moves to x_3 does not help to decide which state to pick in $\widehat{\mathcal{Q}}^{\mathcal{I}_2^2}$ when observing y_2.

The previous argument obviously only works if the abstraction has full state information from the original system when "simulating" its moves. However, in the setting for SAlCA the controller (which is designed based on the abstraction and therefore usually given as a sub-machine of the latter) can only interact with the system through the (predefined) set of output symbols Y. As the state space of $\mathcal{Q}$ is usually infinite while Y is finite, this usually implies that no full state feedback is available. Intuitively, one would therefore need to require that non-determinism in $\widehat{\mathcal{Q}}$ can be resolved without full state information.

This issue was recently discussed in [51] where it is shown that alternating simulation relations are in general not sufficient for abstraction based control if no full state feedback is available. To overcome this issue [51] suggests *feedback refinement relations* for a particular class of transition systems which allow for abstraction based controller synthesis using a predefined set of output events. As applying these ideas to the abstractions constructed in this thesis would require a non-trivial extension of the relations provided in [51], we postpone this idea to future work.

However, even without this formal extension, we can draw the following conclusions from the construction of $\mathcal{I}_m^l$-abstract state machines in Definition 6.3. Observe, that choosing $m = 0$, i.e., considering the original realization of SAlCA, will always result in a deterministic state machine, i.e., observing an output $y \in Y$ fully determines the next state of the abstraction $\widehat{\mathcal{Q}}^{\mathcal{I}_0^l}$. Hence, the issue of unresolved non-determinism discussed above cannot occur in $\widehat{\mathcal{Q}}^{\mathcal{I}_0^l}$ and $\mathcal{R}$ in (6.16) is always a simulation relation from $\mathcal{Q}$ to $\widehat{\mathcal{Q}}^{\mathcal{I}_0^l}$ (from Corollary 6.11). Nevertheless, (8.11) still needs to hold to allow for an alternating simulation relation. For $\widehat{\mathcal{Q}}^{\mathcal{I}_0^1}$ and $\widehat{\mathcal{Q}}^{\mathcal{I}_0^2}$ constructed in Section 6.5 the latter is unfortunately not true as, e.g., $\mathrm{U}_\delta(x_2) = \{u_2\} \subseteq \mathrm{U}_{\widehat{\delta}^{\mathcal{I}_0^1}}(y_1) = \{u_2, u_4\}$ and $(x_2, y_1) \in \mathcal{R}^{\mathcal{I}_0^l}$

(from (6.16)).

Interestingly, this observation draws a nice connection to the conditions for controller synthesis using SlCA in [38]. Therein, the original system is required to have a *free input*, i.e.,

$$\forall x \in X \;.\; \mathrm{U}_\delta(x) = U. \tag{8.12}$$

As (8.12) always implies (8.11), assuming a free input implies that $\mathcal{R}^{\mathcal{I}^l_0}$ is an alternating simulation relation. In this case obviously no state information is needed for the abstraction to simulate the moves of the original system. In this particular scenario alternating simulation relations are therefore sufficient for controller synthesis even with no full state feedback.

Using these insights it would be interesting to investigate which conditions on $\mathcal{Q}$ allow for control based on a predefined set of input and output symbols using abstract state machines $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ with $m > 0$ and quotient state machines $\widehat{\mathcal{Q}}^{l\blacktriangledown}$. As we have shown that increasing m results in tighter abstractions this could be beneficial if $\widehat{\mathcal{Q}}^{\mathcal{I}^l_0}$ is not tight enough for a particular controller synthesis problem and increasing l does not refine the abstraction sufficiently. It seems promising to extend the notion of feedback refinement relations from [51] for this purpose.

It was briefly mentioned in Chapter 1 that controllers for QBA are usually constructed using a variant of *reactive synthesis* (see, e.g., [67, Ch.6]) while SlCA where used for abstraction based control by applying a variant of *supervisory control theory* (SCT) in [38]. The main difference between those synthesis techniques lies in the interpretation of the system model. While reactive synthesis interprets the symbolic abstraction (given as an FSM) as a *Büchi-Automaton*, i.e., a generator of *infinite* strings, SCT usually interprets the FSM as a *deterministic finite automaton* (DFA), i.e., a generator of *finite* strings.

There also exists SCT over infinite strings (see, e.g., [73, 74, 40]). While this synthesis approach is not very popular in the control systems community, it is well suited for the abstraction based controller synthesis problem considered in Part I of this thesis, as using behavioral systems theory to model the original system implicitly assumes that its dynamics evolve over infinite stings. Therefore, completing the bridge between abstraction based controller synthesis using QBA and SAlCA would require the investigation of the connection of SCT over infinite strings and reactive synthesis. While *classical* SCT using DFA was recently compared to reactive synthesis in [10], the connection between reactive synthesis and SCT over infinite strings remains a challenging open problem for future research.

Part II.

A Bridge Between Supervisory Control Theory and Deterministic Pushdown Automata

9. Literature Review and Contributions II

In Part II of this thesis, supervisory control theory (SCT), a symbolic controller synthesis technique, is extended to a larger class of specifications, namely specifications that can be realized by deterministic pushdown automata (DPDA). We first review the original version of SCT in Section 9.1 and discuss the enriched modeling capabilities of DPDA compared to DFA in Section 9.2. In Section 9.3 recent extensions of SCT to other language classes are discussed. Finally, Section 9.4 outlines the incorporation of SCT and DPDA and discusses the contributions of Part II. This chapter is concluded by a list of publications by the author, where some results contained in Part II have appeared previously.

9.1. Supervisory control theory (SCT)

Supervisory control theory (SCT) was established by Ramadge and Wonham [48, 78, 49] for the purpose of controller synthesis on formal languages. Given a finite set of symbols Σ, a language is a set of *finite* sequences of symbols from Σ. In comparison, a discrete behavior $\mathcal{B}$, as used in Part I, consists of *infinite* sequences of symbols. In language theory, discrete behaviors are referred to as ω-languages [39], opposed to $*$-languages, which we only call *languages* in this thesis.

Historically, SCT refers to a controller synthesis problem where the system, usually called *plant*, and the specification are given by languages rather then behaviors. Given a plant and a specification, SCT defines a *minimally restrictive supervisor* as a controller which (i) generates a closed loop system (i.e., the plant interconnected with a controller) respecting the specification, (ii) restricts the plant as little as possible, (iii) does not prevent uncontrollable (i.e., unpreventable) events and (iv) ensures that the system is always able to evolve to a desired situation. This was formalized in [78] by defining the desired closed loop language $\mathcal{K}$ (i.e., the set of desired finite strings of symbols the system should generate under control) as a supremum over the intersection of the plant and specification language respecting the above properties.

The question of how to calculate $\mathcal{K}$ is called the *supervisory control problem* (SCP). In [78], an implementable fixed point algorithm solving the SCP was presented using finite automaton realizations of the involved languages. This fixed point algorithm iteratively executes the following on the product automaton of plant and specification. (i) It removes controllability problems, i.e., situations where the controller attempts to

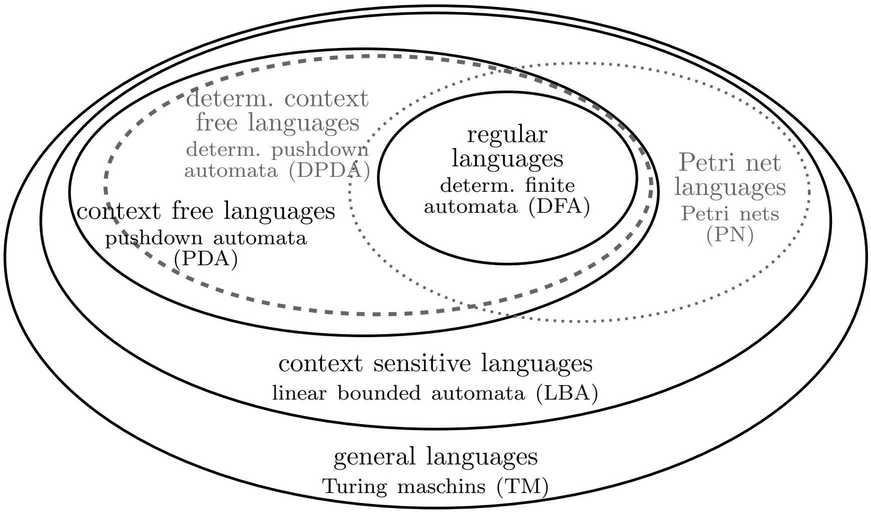

Figure 9.1.: Chomsky Hierarchy (solid black) with the intermediate language class of DCFL (dashed gray) and the class of PNL (dotted grey).

prevent uncontrollable events, minimal restrictively. (ii) It resolves blocking issues, i.e., situations where the closed loop cannot reach a marking state. Obviously, step (i) may generate new blocking issues, while step (ii) may lead to new controllability problems. The algorithm terminates when no more controllability problems or blocking situations are present. It was shown in [78] that this algorithm always terminates. As it calculates a *minimally restrictive* supervisor, obtaining an empty solution $\mathcal{K} = \emptyset$ implies that there exists no solution to the considered SCP.

9.2. Deterministic Pushdown Automata (DPDA)

A very popular method to categorize formal languages is the Chomsky Hierarchy [7, 24], depicted in Figure 9.1. The lowest level of this hierarchy is given by the class of regular languages (REG), realized by deterministic finite automata (DFA). REG are the least expressive languages in the Chomsky Hierarchy, but have the nicest computational properties. Moving upwards in the hierarchy increases expressiveness while decreasing the existence of closure properties.

The class of deterministic context free languages (DCFL) is a strict subclass of the context free languages (CFL) and the smallest language class in the Chromsky Hierarchy fully containing REG. DCFL can be realized by deterministic pushdown automata (DPDA), which are essentially DFA equipped with an infinite stack which is used in a "last in-first out" manner.

Using this stack, DPDA can model two additional phenomena compared to DFA. First, the number of occurrences of different symbols can be matched during the evolution of the system. Second, the ordering of past events can be memorized to reproduce this order in the future evolution of the system in a reverse manner. These additional modeling capabilities were originally introduced to describe arithmetic expressions in computer programs with arbitrary nesting of balanced parentheses. However, also in manufacturing systems, matching the number or order of different production steps is a common task. This motivates the use of DPDA to model the specification in the context of SCT.

9.3. Extensions of SCT

Recall that the fixed point algorithm solving the SCP presented in [78] and discussed in Section 9.1 uses finite automaton realizations of the involved languages. This clearly restricts its applicability to regular plant and specification languages. Interestingly, it was shown in [62] and [33] that the SCP cannot be solved for the case where both the plant and the specification language are DCFL, as DCFL are not closed under intersection. However, for the setting where only the specification language is a DCFL while the plant language is still regular, Griffin suggested an implementable algorithm to solve the SCP in [20, 19]. Unfortunately, as shown by the counterexample in Appendix B.2, Griffin's algorithm does not construct a minimally restrictive supervisor as required by the SCP, contrary to his claim in [20, 19].

Another very popular language class fully containing REG is the class of Petri net languages (PNL). Petri nets where introduced by Carl A. Petri in [43] and are incomparable to DCFL w.r.t. to their expressive power, as depicted in Figure 9.1. SCT was also extended to the class of Petri nets in [16, 17], where both the plant and the specification are modeled by a Petri net. However, to the best of the authors knowledge, it is not possible to solve the SCP over PNL such that the resulting controller is minimally restrictive (see also the surveys [23, 25]). As there exists a class of languages contained in both DCFL and PNL, solving the latter issue for DCFL specifications will generate a solution that is transferable to this subset of PNL specifications.

In [73, 74] SCT was also formalized for regular ω-languages, which are realizable by Büchi-Automata, called ωSCT. While this synthesis approach is not very popular in the control systems community, it would be more natural to use in the context of abstraction based controller synthesis as briefly discussed in Section 8.3 of Part I and shown in [39]. To apply the classical version of SCT in abstraction based controller synthesis from Part I, the finite state symbolic abstraction of the plant resulting from QBA or SAlCA needs to be interpreted as a DFA rather then a Büchi-Automaton. In this case the plant is implicitly assumed to terminate its evolution at some point, as it only produces *finite* sequences of symbols. However, as those finite sequences can be arbitrarily long, the two mentioned interpretations only differ on a conceptual level. Nevertheless it should be kept in mind that controllers synthesized using the classical version of SCT do not ensure that the resulting closed loop is live, i.e., can run for infinite time.

9.4. Outline and Contributions

In Part II of this thesis we show how the SCP can be solved for the setting of a regular plant and a deterministic context free specification language. In comparison to the suggested (but incomplete) solution in [20], we do not restrict ourselves to prefix-closed languages (see Section 10.1 for a formal definition).

Unfortunately, the iterative procedure to solve the SCP for REG suggested in [78] does not readily carry over to the case where the specification is described by a DCFL. To solve the SCP for the latter case, an alternative fixed-point algorithm is needed. Based on [61] and [56] we introduce an alternative procedure in Chapter 10 which is able to cope with DCFL specifications. While this algorithm differs from the standard procedure in [78], it still involves the two basic steps of removing controllability problems and blocking situations, as discussed in Section 9.1. However, in the case where the specification is described by a DCFL, both steps have to be realized for DCFL rather then for REG. Therefore, the desired closed loop language $\mathcal{K}$ is computable in this setting if, for the class of DCFL, (i) there exists an implementable algorithm to remove controllability problems, (ii) there exists an implementable algorithm to remove blocking problems (iii) it is decidable if no more controllability (or blocking) problems are present and (iv) the new fixed point algorithm terminates using the implementable algorithms for (i) and (ii).

Based on previous work in [55] and [56], we will show in Chapter 11 that (i) and (iii) are true. I.e., we will derive an implementable algorithm to remove controllability problems in DPDA and show how to check that no more controllability problems are present. Concerning (ii), an implementable algorithm removing blocking problems in DPDA was suggested in [60, 59]. Its construction is based on the realization of DCFL by context free grammars satisfying the LR(1) determinism property and deterministic parsers. As these realizations of DCFL are substantially different from DPDA, we refrain from summarizing these results in this thesis. Unfortunately, (iv) is still an open problem and left for future research. We have, however, not encountered termination problems in any realistic example.

The implementation of the overall algorithm solving the SCP for deterministic context free specification languages is available as a plug-in in `libFAUDES` [31].

9.5. Publications

In the following publications by the author some results contained in Part II of this thesis have appeared previously or are currently under review.

[55] Anne-Kathrin Schmuck, Sven Schneider, Jörg Raisch, and Uwe Nestmann. Extending supervisory controller synthesis to deterministic pushdown automata—enforcing controllability least restrictively. In *Proceedings of the 12th IFAC - IEEE International Workshop on Discrete Event Systems*, pages 286–293, 2014.

[61] Sven Schneider, Anne-Kathrin Schmuck, Jörg Raisch, and Uwe Nestmann. Reducing an operational supervisory control problem by decomposition for deterministic pushdown automata. In *Proceedings of the 12th IFAC - IEEE International Workshop on Discrete Event Systems*, pages 214–221, 2014.

[56] Anne-Kathrin Schmuck, Sven Schneider, Jörg Raisch, and Uwe Nestmann. Supervisory control synthesis for deterministic context free specification languages – enforcing controllability least restrictively. *Discrete Event Dynamic Systems*, 2015. (to appear).

10. The Supervisory Control Problem over Language Models

In [78], Ramadge and Wonham suggested an iterative procedure to solve the supervisory control problem (SCP) which works well for the case of regular languages but does not readily carry over to the case where the specification is described by a DCFL. To solve the SCP for the latter case, an alternative fixed-point algorithm is needed.

In Section 10.1 we first introduce required notation and define language models (LM) to describe the symbolic dynamics of a system. Based on previous work in [61] and [56], we then provide an alternative iterative algorithm to solve the SCP over LM in Section 10.2, which is able to cope with DCFL specifications.

10.1. Preliminaries

In this section, required notation is introduced and language models are defined, which are used as the basic modeling framework in Part II of this thesis. We furthermore introduce required properties for this system model and show, that the set of all language models forms a complete lattice w.r.t. set inclusion.

Notation

Let Σ be a finite set of symbols or events, called the *alphabet*. This alphabet is, as usual, partitioned into a set of controllable events (preventable by the controller) and a set of uncontrollable events, i.e., $\Sigma = \Sigma_{\mathsf{c}} \cup \Sigma_{\mathsf{uc}}$ with $\Sigma_{\mathsf{c}} \cap \Sigma_{\mathsf{uc}} = \emptyset$, denoted by $\Sigma = \Sigma_{\mathsf{c}} \uplus \Sigma_{\mathsf{uc}}$. Furthermore, Σ^* and Σ^ω denotes the set of all finite- and infinite-length strings of symbols from Σ, respectively, and we define $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$ and $\Sigma^{\leq 1} = \Sigma \cup \{\lambda\}$, where λ is the empty string. The *projection* of a string $w \in \Sigma^*$ on its ith element is denoted by $\pi_i(w)$ and the *concatenation* of two strings $w, w' \in \Sigma^*$ is denoted by $w \cdot w'$ (meaning that w' is appended to w). Moreover, w is defined to be a *prefix* of w', denoted by $w \sqsubseteq w'$, if $\exists w'' \in \Sigma^* \,.\, w \cdot w'' = w'$ and w is defined to be a *strict prefix* of w', denoted by $w \sqsubset w'$, if $\exists w'' \in \Sigma^+ \,.\, w \cdot w'' = w'$.

Language Models

As usual, any subset of Σ^* is called a *language*, denoted by $L \subseteq \Sigma^*$ and the *prefix closure* $\overline{L} \subseteq \Sigma^*$ of $L \subseteq \Sigma^*$ is defined by $\overline{L} := \{w \in \Sigma^* \mid \exists w' \in L \,.\, w \sqsubseteq w'\}$. L is called *prefix-closed* if $L = \overline{L}$.

Remark 10.1. Recall, that we have used discrete behaviors $\mathcal{B}$ in Part I to model the external sequences of symbols generated by the system. Using the alphabet Σ as the finite external symbol set W, those behaviors are simply given by $\mathcal{B} \subseteq \Sigma^\omega$, as they are defined as a set of *infinite* sequences of symbols. In the sence of language theory, a discrete behavior is therefore simply an ω-language (opposed to a $*$-language, which we only call *language* in this thesis). $\triangleleft$

Using languages instead of behaviors to model the symbolic dynamics of a system implicitly assumes that the system will terminate its evolution at some point, as it only produces *finite* sequences of symbols. Therefore, one usually distinguishes between the set of *all* generated finite event sequences, called the (prefix closed) *unmarked* language, and a (usually strict) subset of the latter, called the *marked* language. The marked language contains only those strings after which a termination of the systems evolution is expected, e.g., after the successful completion of a certain task. We therefore describe a system by a *language model*, combining its marked and unmarked language.

Definition 10.1. *Let $L_{\mathrm{um}}, L_{\mathrm{m}} \subseteq \Sigma^*$ be languages over $\Sigma = \Sigma_{\mathsf{c}} \uplus \Sigma_{\mathsf{uc}}$. Then a language model D is defined by the tuple $D = \llparenthesis L_{\mathrm{um}}, L_{\mathrm{m}} \rrparenthesis$ s.t. $\mathcal{L}_{\mathrm{um}}(D) = L_{\mathrm{um}} = \overline{L_{\mathrm{um}}}$ and $\mathcal{L}_{\mathrm{m}}(D) = L_{\mathrm{m}} \subseteq L_{\mathrm{um}}$. The set of all language models over Σ is referred to as LM.*

As languages are partially ordered by set inclusion, not surprisingly, LM form a complete lattice using the following intuitive partial order.

Definition 10.2. *Let $D, D' \in \mathsf{LM}$. Then*

$$D \leq D' \Leftrightarrow \begin{pmatrix} \mathcal{L}_{\mathrm{um}}(D) \subseteq \mathcal{L}_{\mathrm{um}}(D') \\ \wedge \mathcal{L}_{\mathrm{m}}(D) \subseteq \mathcal{L}_{\mathrm{m}}(D') \end{pmatrix}. \tag{10.1}$$

Lemma 10.3 ([61], Lemma 1[1])**.** $(\mathsf{LM}, \leq)$ *is a* complete lattice *with*

$\inf(D, D') := \llparenthesis\, \mathcal{L}_{\mathrm{um}}(D) \cap \mathcal{L}_{\mathrm{um}}(D'), \mathcal{L}_{\mathrm{m}}(D) \cap \mathcal{L}_{\mathrm{m}}(D') \rrparenthesis$ *and*

$\sup(D, D') := \llparenthesis\, \mathcal{L}_{\mathrm{um}}(D) \cup \mathcal{L}_{\mathrm{um}}(D'), \mathcal{L}_{\mathrm{m}}(D) \cup \mathcal{L}_{\mathrm{m}}(D') \rrparenthesis$.

[1] Note that all results from [61] used in this thesis were verified by S. Schneider using the interactive theorem prover Isabelle/HOL [42]. We therefore refrain from providing proofs for those statements.

Controllability and Blocking

Recall that the fixed-point algorithm to solve the SCP over regular languages consists of two steps, namely (i) the removal of controllability problems, i.e., situations where the controller attempts to prevent uncontrollable events minimal restrictively, and (ii) the removal of blocking issues, i.e., situations where the closed loop cannot reach a marking state. To formalize the solution of the SCP over LM we therefore need to introduce the notion of controllability and blocking for languages.

Definition 10.4. *Let $L, L' \subseteq \Sigma^*$ be languages s.t. $\Sigma = \Sigma_{\mathsf{c}} \uplus \Sigma_{\mathsf{uc}}$. Then L is defined to be nonblocking w.r.t. L' (written* $\mathrm{Nb}(L, L')$*) if $L \subseteq \overline{L'}$. Furthermore, $w \in L = \overline{L}$ is defined to be controllable w.r.t. L' (written* $\mathrm{ContS}(L, L', w)$*) if*

$$\forall \mu \in \Sigma_{\mathsf{uc}} \,.\, \big(w\mu \in L' \Rightarrow w\mu \in L\big)\,, \tag{10.2}$$

and $L = \overline{L}$ is defined to be controllable w.r.t. $L' = \overline{L'}$ (written $\mathrm{Cont}(L, L')$*) if*[2]

$$\forall w \in L \,.\, \mathrm{ContS}(L, L', w).$$

Using the notions from Definition 10.4 we can define nonblocking and controllable LM in an obvious manner.

Definition 10.5. *Let $D, D' \in \mathsf{LM}$. Then D is* nonblocking *(written* $\mathrm{NbLM}(D)$*), if* $\mathrm{Nb}(\mathcal{L}_{\mathrm{um}}(D), \mathcal{L}_{\mathrm{m}}(D))$ *and D is* controllable *w.r.t. D' (written* $\mathrm{ContLM}(D, D')$*) if* $\mathrm{Cont}(\mathcal{L}_{\mathrm{um}}(D), \mathcal{L}_{\mathrm{um}}(D'))$.

10.2. A New Iterative Algorithm

In the usual formulation of the SCP in [78] a controller D_C is a *minimally restrictive supervisor* for a plant D_P and a specification D_S, if D_C (i) generates a closed loop $D_{cl} = \inf(D_P, D_C)$ which contains as many words as possible while respecting the specification, (ii) does not prevent uncontrollable events (i.e., D_{cl} is controllable w.r.t. D_P), and (iii) ensures that the system is always able to evolve into a marked word (i.e., D_{cl} is nonblocking). These three properties are formalized in the following definition.

Definition 10.6. *Let $D_P, D_S \in \mathsf{LM}$. Then*

$$D_C \in \mathsf{LM} \quad s.t. \quad \begin{pmatrix} \mathcal{L}_{\mathrm{m}}(\inf(D_P, D_C)) \subseteq \mathcal{L}_{\mathrm{m}}(D_S) \\ \wedge \mathrm{ContLM}(\inf(D_P, D_C), D_P) \\ \wedge \mathrm{NbLM}(\inf(D_P, D_C)) \end{pmatrix} \tag{10.3a}$$

[2] Observe that $\mathrm{Cont}(L, L')$ is equivalent to the standard definition of controllability given by $(L \cdot \Sigma_{\mathsf{uc}}) \cap L' \subseteq L$.

is a sound controller *(written $D_C \in \mathcal{C}_{\text{sound}}(D_P, D_S)$).*

$$D_C \in \mathcal{C}_{\text{sound}}(D_P, D_S) \quad \textit{s.t.} \quad \begin{array}{l} \forall\, D'_C \in \mathcal{C}_{\text{sound}}(D_P, D_S) \;.\\ \quad (\inf(D_P, D'_C) \leq \inf(D_P, D_C)) \end{array} \tag{10.3b}$$

is a maximally permissive controller *(written $D_C \in \mathcal{C}_{\text{mp}}(D_P, D_S)$) and*

$$D_C \in \mathcal{C}_{\text{mp}}(D_P, D_S) \quad \textit{s.t.} \quad \forall D'_C \in \mathcal{C}_{\text{mp}}(D_P, D_S) \;.\; D_C \leq D'_C \tag{10.3c}$$

is the smallest maximally permissive controller *(written $D_C = \mathcal{C}_{\text{smp}}(D_P, D_S)$).*

Note that the first and the last line of (10.3a) imply that for sound controllers $D_C \in \mathcal{C}_{\text{sound}}(D_P, D_S)$, the unmarked closed loop language is a subset of the unmarked specification language, i.e., $\mathcal{L}_{\text{um}}(\inf(D_P, D_C)) \subseteq \mathcal{L}_{\text{um}}(D_S)$. Furthermore, maximally permissive controllers as defined in Definition 10.6 are equivalent to minimally restrictive supervisors as defined in [78].

Interestingly, $D_C = \llparenthesis \emptyset, \emptyset \rrparenthesis$ is a sound controller for arbitrary D_P and D_S, as in this case (10.3a) trivially holds. Calculating a maximal permissive controller excludes this trivial (but sound) solution, as long as a non-empty solution exists. Given a plant D_P and a specification D_S, (10.3b) implies that maximally permissive controllers $D'_C \in \mathcal{C}_{\text{mp}}(D_P, D_S)$ are not unique but generate a unique closed loop $\inf(D_P, D'_C)$. Using (10.3c), this closed loop is equivalent to the smallest maximally permissive controller $D''_C = \mathcal{C}_{\text{smp}}(D_P, D_S)$, which is obviously unique. Using this intuition, D''_C can be obtained by the following supremum over the intersection of D_P and D_S.

Theorem 10.7 ([61], Prop. 2, Thm. 7)**.** *Let $D_P, D_S \in \mathsf{LM}$. Then*

$$\mathcal{C}_{\text{smp}}(D_P, D_S) = \operatorname{Sup}\left\{ \llparenthesis \overline{L_{\text{m}}}, L_{\text{m}} \rrparenthesis \;\middle|\; \begin{pmatrix} L_{\text{m}} \subseteq \mathcal{L}_{\text{m}}(\inf(D_P, D_S)) \\ \wedge \operatorname{Cont}(\overline{L_{\text{m}}}, \mathcal{L}_{\text{um}}(D_P)) \end{pmatrix} \right\}.$$

In [78] the marked language of $\mathcal{C}_{\text{smp}}(D_P, D_S)$ in Theorem 10.7 is called the marked supremal controllable (nonblocking) sublanguage of $\mathcal{L}_{\text{m}}(D_P) \cap \mathcal{L}_{\text{m}}(D_S)$ denoted by $\mathcal{K}$. To calculate $\mathcal{K}$, a monotonic operator $\Omega : 2^{\Sigma^*} \to 2^{\Sigma^*}$, s.t.

$$\Omega(L_{\text{m}}) := \{w \in L_{\text{m}} \mid \forall w' \sqsubseteq w \;.\; \operatorname{ContS}(\overline{L_{\text{m}}}, \mathcal{L}_{\text{um}}(D_P), w')\} \tag{10.4}$$

was introduced in [78, Lemma 2.1].

By iteratively applying Ω starting with $\mathcal{L}_{\text{m}}(D_P) \cap \mathcal{L}_{\text{m}}(D_S) = \mathcal{L}_{\text{m}}(\inf(D_P, D_S))$ one obtains $\mathcal{K}$ as the (unique) greatest fixed point of Ω. Furthermore, it was shown in [78, Sec. 5] that, if D_P and D_S are realizable by DFA, there exists an implementable algorithm operating on the automata level calculating a DFA realizing $\mathcal{K}$. Unfortunately, as pointed out earlier, this algorithm cannot be translated in a straightforward manner to the scenario considered in this thesis. Therefore, we define a new iterative calculation of $\mathcal{C}_{\text{smp}}(D_P, D_S)$ as a basis for an implementable algorithm solving the SCP for DPDA specifications.

Definition 10.8. *Let* $D \in \mathsf{LM}$ *and* $\mathrm{F_c}, \mathrm{F_{nb}} : \mathsf{LM} \to \mathsf{LM}$ *s.t.*

$$\mathrm{F_c}(D) := \llparenthesis \Psi(\mathcal{L}_{\mathrm{um}}(D)), \Omega(\mathcal{L}_{\mathrm{m}}(D)) \rrparenthesis \tag{10.5a}$$

$$\mathrm{F_{nb}}(D) := \llparenthesis \overline{\mathcal{L}_{\mathrm{m}}(D)}, \mathcal{L}_{\mathrm{m}}(D) \rrparenthesis, \tag{10.5b}$$

where $\Psi : 2^{\Sigma^*} \to 2^{\Sigma^*}$ *s.t.*

$$\Psi(L_{\mathrm{um}}) := \{w \in L_{\mathrm{um}} \mid \forall w' \sqsubset w \,.\, \mathrm{ContS}(L_{\mathrm{um}}, \mathcal{L}_{\mathrm{um}}(D_P), w')\} \tag{10.6}$$

and Ω *as in* (10.4)*. Then*

$$\mathcal{U}(\mathrm{F_c}, \mathrm{F_{nb}}, D) := \begin{cases} D & , \ \mathrm{F_c}(D) = D \\ \mathcal{U}(\mathrm{F_c}, \mathrm{F_{nb}}, \mathrm{F_{nb}}(\mathrm{F_c}(D))) & , \ \textit{else} \end{cases} \tag{10.7}$$

is the iterative calculation *of* $\mathrm{F_{nb}} \circ \mathrm{F_c}$ *until* $\mathrm{F_c}$ *returns its input.*

Intuitively, given an LM D, the function $\mathrm{F_c}(D)$ returns a controllable (with respect to D_P) LM with maximal marked language, while the function $\mathrm{F_{nb}}(D)$ returns a maximal nonblocking LM. As both functions can only remove strings from the unmarked and the marked language, it is easy to show, that they are monotone w.r.t. to the partial order $\leq$ defined in Definition 10.2.

Lemma 10.9. *Let* $D \in \mathsf{LM}$*. Then* $\mathrm{F_c}(D) \leq D$ *and* $\mathrm{F_{nb}}(D) \leq D$*.*

Proof. Follows from Def. 17(i), Lemma 5 and Lemma 8 in [61]. □

Observe, that the operator $\mathcal{U}(\mathrm{F_c}, \mathrm{F_{nb}}, D)$, defined in Definition 10.8, iterates over removing noncontrollability and blocking situations until a fixed-point is reached. This iteration can be used to formalize the solution of the SCP in the following obvious way.

Theorem 10.10. *Let* $D_P, D_S \in \mathsf{LM}$*. Then*

$$\mathcal{C}_{\mathrm{smp}}(D_P, D_S) = \mathcal{U}(\mathrm{F_c}, \mathrm{F_{nb}}, \mathrm{F_{nb}}(\inf(D_P, D_S))),$$

if $\mathcal{U}(\mathrm{F_c}, \mathrm{F_{nb}}, \mathrm{F_{nb}}(\inf(D_P, D_S)))$ *terminates.*

Proof. Follows from Lemma 4, Def. 19 and Thm. 11 in [61]. □

Remark 10.2. In contrast to Theorem 10.10, the algorithm presented in [78, Sec. 5] to calculate $\mathcal{K}$, uses, in slightly different notation, the iterator

$$\mathrm{F'_c}(D) = \llparenthesis \Psi'(\mathcal{L}_{\mathrm{um}}(D)), \Psi'(\mathcal{L}_{\mathrm{um}}(D)) \cap \mathcal{L}_{\mathrm{m}}(D) \rrparenthesis \tag{10.8}$$

$$\text{with} \quad \Psi'(L_{\mathrm{um}}) = \{w \in L_{\mathrm{um}} \mid \forall w' \sqsubseteq w \,.\, \mathrm{ContS}(L_{\mathrm{um}}, \mathcal{L}_{\mathrm{um}}(D_P), w')\}.$$

Note that Ψ' in (10.8) differs from Ψ in (10.6) by all-quantifying $w' \sqsubseteq w$ instead of $w' \sqsubset w$. It will be shown in Remark 11.2 in Section 11.3 why Ψ' (in contrast to Ψ) cannot be implemented for DPDA realizations of LM. ◁

11. Enforcing Controllability for DCFL Models Least Restrictively

In this chapter we are aiming at an implementable solution of the SCP where the plant is given by an LM D_P with marked and unmarked REG and the specification is given by an LM D_S with unmarked and marked DCFL. As the intersection of a REG and a DCFL is a DCFL (see [24, p.135]), the unmarked and marked language of $D = \inf(D_P, D_S)$ are also DCFL.

Recall from Theorem 10.10, that the iterative algorithm $\mathcal{U}$ defined in Definition 10.8 is initialized with $D = \inf(D_P, D_S)$ to calculate $D_C = \mathcal{C}_{\mathrm{smp}}(D_P, D_S)$ as its fixed point. Therefore, the desired smallest maximally permissive controller D_C is computable by the algorithm $\mathcal{U}$ in Theorem 10.10 for the setting of a regular plant and a deterministic context free specification language if, for the class of DCFL, (i) $\mathrm{F_c}$ is computable, (ii) $\mathrm{F_{nb}}$ is computable, (iii) it is decidable if $\mathrm{F_c}$ returns its input and (iv) $\mathcal{U}$ terminates for the particular choice of D_P and D_S using the implementable algorithms realizing $\mathrm{F_c}$ and $\mathrm{F_{nb}}$.

We will show in this chapter that (i) and (iii) are true, i.e., we will derive an implementable algorithm realizing $\mathrm{F_c}$ and show how to check whether $\mathrm{F_c}$ returns its input. As this algorithm will use the DPDA realizations of the involved DCFL, we first review different automata, in particular DPDA, and their semantics in Section 11.1 and discuss how they realize LM. In Section 11.2, we introduce a detailed manufacturing example to outline the practical relevance of the SCP considered in this thesis which is used in Section 11.3 to illustrate the functions defined therein. Finally, it is shown in Section 11.4, that the chain of automata manipulations introduced in Section 11.3 actually realizes $\mathrm{F_c}$ on the automaton level. It is furthermore shown in Section 11.4 that it is decidable weather $\mathrm{F_c}$ returns its input.

11.1. Realizing LM by Automata

In this section, we briefly review different automata, in particular DPDA, and show how they realize LM.

Pushdown Automata (PDA)

Pushdown automata are usually defined as follows.

Definition 11.1. *A pushdown automaton (*PDA*) is a tuple*

$$M := (Q, \Sigma, \Gamma, \delta, q_0, \square, F) \tag{11.1}$$

s.t. (i) the state set Q, the external alphabet Σ, the stack alphabet Γ, and the set of transitions δ are finite, (ii) $\delta \subseteq Q \times \Sigma^{\leq 1} \times \Gamma \times \Gamma^* \times Q$ is the stack-dependent next state relation, (iii) $\square \in \Gamma$ is the end-of-stack marker, (iv) $F \subseteq Q$ is the set of marking states, (v) $q_0 \in Q$ is the initial state and $\square$ is the initial stack-content, (vi) $\square$ is always retained as the bottom element of the stack.

When graphically representing PDA, a transition $(q, \sigma, \gamma, s, q') \in \delta$ is usually depicted by an edge from q to q' labeled by σ, γ, s, denoting that by taking this transition, $\sigma \in \Sigma^{\leq 1}$ is generated, $\gamma \in \Gamma$ (called "stack-top") is popped from the top of the stack, and $s \in \Gamma^*$ is pushed onto the stack (with the right-most symbol first). Note that $\sigma \in \Sigma^{\leq 1}$, i.e., σ can either be a symbol from the alphabet or the empty string. Two transitions with the same pre and post state are depicted by one edge with two labels. This is illustrated in the following example.

Example 11.1. Consider the external alphabet $\Sigma = \{a, b\}$, the stack alphabet $\Gamma = \{\square, \bullet\}$, the state set $Q = \{q_1, q_2, q_3\}$, the set of marking states $F = \{q_3\}$ and the initial state q_1. Then M_1 in Figure 11.1 is a PDA.
The word $aabb$ is generated by M_1 using the chain

$$q_1 \begin{bmatrix} \square \end{bmatrix} \xrightarrow{a,\square,\bullet\square} q_1 \begin{bmatrix} \bullet \\ \square \end{bmatrix} \xrightarrow{a,\bullet,\bullet\bullet} q_1 \begin{bmatrix} \bullet \\ \bullet \\ \square \end{bmatrix} \xrightarrow{b,\bullet,\lambda} q_2 \begin{bmatrix} \bullet \\ \square \end{bmatrix} \xrightarrow{b,\bullet,\lambda} q_2 \begin{bmatrix} \square \end{bmatrix} \xrightarrow{\lambda,\square,\square} q_3 \begin{bmatrix} \square \end{bmatrix}$$

of state and stack tuples. E.g., the transition a,•,•• removes one bullet from the top of the stack (hence, this transition can only occur if there is a bullet on top of the stack) and pushes two bullets back in. In total this transition therefore adds one bullet to the stack. ◁

Note that because of (vi) in Definition 11.1,

$$(q, \sigma, \square, s, q') \in \delta \Rightarrow \left(\exists s' \in \Gamma^* \,.\, s = s' \cdot \square\right),$$

i.e., the end-of-stack marker $\square$ is never removed from the stack. Furthermore, as $\sigma \in \Sigma^{\leq 1}$ in Definition 11.1, M can do "silent moves" (called λ-transitions), possibly modifying the stack but not generating an external symbol (e.g., see M_1 in Figure 11.1, from q_2 to q_3). We collect the starting states of all λ-transitions in the set

$$Q_\lambda(M) := \{q \in Q \mid \exists \gamma \in \Gamma, s \in \Gamma^*, q' \in Q \,.\, (q, \lambda, \gamma, s, q') \in \delta\}.$$

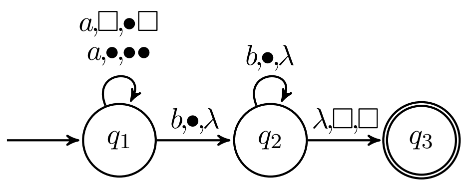

Figure 11.1.: PDA M_1 in Example 11.1.

Configurations of PDA

From a system theoretic point of view a system state would be a pair (q, s), where $q \in Q$ is an (automaton) state and $s \in \Gamma^*$ represents the current stack content. Hence, a PDA with $|Q| < \infty$ may have infinite state space (in a system theoretic sense) since the stack is not restricted. In the computer science literature the pair (q, s) is sometimes referred to as a configuration. For our purposes it turns out to be convenient to add the string of generated external symbols to this pair.

Definition 11.2. *The set of configurations of M is defined by $\mathcal{C}(M) := Q \times \Sigma^* \times \Gamma^*$ and the initial configuration is $(q_0, \lambda, \Box)$.*

In Definition 11.2, $(q, w, s) \in \mathcal{C}(M)$ consists of a state q, a history variable w (storing the external symbols generated), and a stack variable s (storing the current stack content). Obviously, a transition from one configuration to another occurs when M takes a transition $e \in \delta$. To make this step visible, we augment, whenever suitable, configurations by the corresponding incoming state transition. Using this convention, the next configuration transitions are defined as follows.

Definition 11.3. *The next configuration transition of M is defined by $\vdash_M \subseteq ((\delta \cup \{\perp\}) \times \mathcal{C}(M))^2$ s.t.*

$$(e, (q, w, \gamma \cdot s')) \vdash_M (e', (q', w \cdot \sigma, s \cdot s')) \text{ for } e' = (q, \sigma, \gamma, s, q'),$$

where $\perp$ denotes an undefined (or irrelevant) pre-transition.

Starting with the initial configuration, i.e., $(\perp, (q_0, \lambda, \Box))$, and iteratively doing single-step configuration transitions according to Definition 11.3 generates trajectories of configurations (enriched with the taken transitions). This can be conveniently used to define languages generated by PDA.

Definition 11.4. *A derivation of M is defined by*[1] $f : \mathbf{N}_0 \rightharpoonup ((\delta \cup \{\perp\}) \times \mathcal{C}(M))$

$$s.t. \quad \forall n < \max(\mathrm{dom(f)}) \; . \; f(n) \vdash_M f(n+1)$$

[1] Here, $\rightharpoonup$ denotes a partial function.

and is called finite if $\max(\mathrm{dom(f)}) < \infty$ *and infinite otherwise.* $\mathcal{D}(M)$ *and* $\mathcal{D}^\omega(M)$ *denote the sets of all finite and infinite derivations, respectively, and the sets* $\mathcal{D}_\mathrm{I}(M)$ *and* $\mathcal{D}_\mathrm{I}^\omega(M)$ *contain all elements of* $\mathcal{D}(M)$ *and* $\mathcal{D}^\omega(M)$*, respectively, starting with* $f(0) = (\perp, (q_0, \lambda, \Box))$*. Furthermore,*

$$\mathcal{D}_{max}(M, w) := \left\{ f \in \mathcal{D}_\mathrm{I}(M) \middle| \begin{pmatrix} \exists e, q, s \;.\; f(\max(\mathrm{dom(f)})) = (e, (q, w, s)) \\ \wedge \nexists e', q', s' \;.\; (e, (q, w, s)) \vdash_M (e', (q', w, s')) \end{pmatrix} \right\}$$

defines the set of maximal finite derivations of a string $w \in \Sigma^*$*.*

As usual, we call a configuration $c \in \mathcal{C}(M)$ reachable, if there exists an initial derivation $d \in \mathcal{D}_\mathrm{I}(M)$ “passing through it”.

Definition 11.5. *The set of* reachable configurations *of* M *is defined by*

$$\mathcal{C}_\mathrm{reach}(M) := \{c \in \mathcal{C}(M) | \exists d \in \mathcal{D}_\mathrm{I}(M), e \in (\delta \cup \{\perp\}), n \in \mathbf{N}_0 \;.\; d(n) = (e, c)\}\,.$$

PDA vs. LM

The unmarked language of a PDA M is the collection of all finite strings of labels from Σ which M can generate. These strings can obviously be determined by projecting the set of finite initial derivations $\mathcal{D}_\mathrm{I}(M)$ on Σ^*. If $\mathcal{D}_\mathrm{I}(M)$ is restricted to the set of derivations ending in a marking state, the same procedure results in the marked language of M. This is formalized in the following definition, resulting in the obvious connection between PDA and LM via their unmarked and marked language. The class of languages generated by PDA is the class of *context free languages* (CFL).

Definition 11.6. *The marked and the unmarked language generated by* M *are given by*

$$\mathcal{L}_m(M) := \{w \mid (q, w, s) \in \mathcal{C}_\mathrm{reach}(M) \wedge q \in F\} \text{ and} \tag{11.2a}$$

$$\mathcal{L}_{um}(M) := \{w \mid (q, w, s) \in \mathcal{C}_\mathrm{reach}(M)\} \tag{11.2b}$$

respectively.

Definition 11.7. *A* PDA M *realizes an* LM D *(written* $M \asymp D$*), if* $\mathcal{L}_m(M) = \mathcal{L}_m(D)$ *and* $\mathcal{L}_{um}(M) = \mathcal{L}_{um}(D)$*.*

Recall that M can do silent moves (called λ-transitions), which may modify the stack but do not generate an external symbol and are therefore not visible in the LM D realized by M. This implies that an infinite string of λ-transitions results in a *silent blocking situation* for M even if NbLM(D).

Definition 11.8. *A* PDA M *is* silently blocking *(written* $M \in$ SBA*) if*

$$\exists d \in \mathcal{D}_\mathrm{I}^\omega(M), n \in \mathbf{N} \;.\; \forall k \geq n \;.\; \pi_2(\pi_2(d(n))) = \pi_2(\pi_2(d(k))),$$

where[2] $\pi_2(\pi_2(e, (q, w, s))) = w$*.*

[2]Slightly abusing notation, we also use $\pi_i(d)$ to denote the ith element of the tuple d.

Observe that every LM D with context free marked and unmarked language can be realized by a PDA M which is not silently blocking. We therefore assume in the remainder of this chapter that the specification $D_S \in$ LM is always realized by an automaton which is not silently blocking and we will show in Section 11.4 that the algorithm realizing $\mathrm{F_c}$ does not introduce silent blocking.

Other Automata Models as Subclasses of PDA

Using the construction of PDA in Definition 11.1, it is straightforward to introduce *deterministic pushdown automata* (DPDA) by restricting the next state relation δ to be deterministic. The class of languages generated by DPDA is the class of *deterministic context free languages* (DCFL).

Definition 11.9. *A deterministic pushdown automaton (*DPDA*) is a* PDA M *s.t.*

$$\begin{aligned}
&\forall e_1 = (q, \lambda, \gamma, s, q') \in \delta, e_2 = (q, \sigma, \gamma, s', q'') \in \delta \,.\, e_1 = e_2 \quad \textit{and} \\
&\forall e_1 = (q, \sigma, \gamma, s, q') \in \delta, e_2 = (q, \sigma, \gamma, s', q'') \in \delta \,.\, e_1 = e_2.
\end{aligned} \tag{11.3}$$

Example 11.2. The PDA M_1 depicted in Figure 11.1 is deterministic, i.e., $M_1 \in$ DPDA, as (11.3) holds. This can be verified by checking that (i) whenever there exists an outgoing λ-transition in state q requiring stack-top γ there exists no other outgoing transition in q requiring stack-top γ and (ii) there exist no two transitions requiring the same stack-top γ and label σ in one state q. The marked and unmarked language of M_1 are

$$\begin{aligned}
\mathcal{L}_m(M_1) &= \{a^n b^n \mid n \in \mathbf{N}\} \quad \text{and} \\
\mathcal{L}_{um}(M_1) &= \{a^m b^n \mid m, n \in \mathbf{N}_0, n \leq m\} = \overline{\mathcal{L}_m(M_1)},
\end{aligned}$$

respectively. ◁

Finite automata can be defined as special PDA which do neither have λ-transitions nor a stack.

Definition 11.10. *A* finite automaton *(FA) is a* PDA M

$$\textit{s.t.} \quad \forall (q, \sigma, \gamma, s, q') \in \delta \,.\, (\gamma = s = \square \wedge \sigma \neq \lambda) \tag{11.4}$$

*and a deterministic finite automaton (*DFA*) is a* DPDA *s.t.* (11.4) *holds.*

The class of languages generated by DFA is the class of regular languages (REG). Note that for every language L generated by an FA, there also exists a DFA generating L [see 24, p.22]. However, this is not true for PDA and DPDA, implying that DCFL are a strict subclass of CFL, i.e., DCFL $\subset$ CFL.

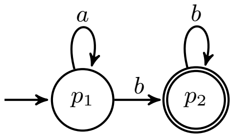

Figure 11.2.: DFA M_2 in Example 11.3.

Example 11.3. The automaton M_2 in Figure 11.2 is a DFA with input alphabet $\Sigma = \{a, b\}$, state set $Q = \{p_1, p_2\}$, marking state set $F = \{p_2\}$ and initial state p_1, where a transition $(p, \sigma, \square, \square, p') \in \delta$ is depicted by an edge from p to p' labeled by σ. The marked and unmarked language of M_2 are given by

$$\begin{aligned} \mathcal{L}_m(M_2) &= \{a^n b b^m \mid n, m \in \mathbf{N}_0\} \quad \text{and} \\ \mathcal{L}_{um}(M_2) &= \{a^n b^m \mid n, m \in \mathbf{N}_0\} = \overline{\mathcal{L}_m(M_2)}, \end{aligned}$$

respectively. ◁

Product Automata

The infimum of two LM, as introduced in Lemma 10.3 and used in Theorem 10.10 to calculate the desired smallest minimally restrictive controller, can be realized by a product automaton. The following standard definition of the product of a DFA and a DPDA is the only one used in the remainder of this chapter and is adapted from [24, p.135].

Definition 11.11. *Let* $M_1 = (Q_1, \Sigma, \{\square\}, \delta_1, q_{10}, \square, F_1) \in$ DFA *and* $M_2 = (Q_2, \Sigma, \Gamma, \delta_2, q_{20}, \square, F_2) \in$ DPDA. *Then the* product automaton *is defined by* $M_\times = M_1 \times M_2 = (Q_\times, \Sigma, \Gamma, \delta_\times, q_{\times 0}, \square, F_\times)$ *with* $Q_\times = Q_1 \times Q_2$,

$$\delta_\times = \left\{ ((p, q), \sigma, \gamma, s, (p', q')) \;\middle|\; \left(\begin{pmatrix} \sigma \in \Sigma \\ \wedge (q, \sigma, \gamma, s, q') \in \delta_2 \\ \wedge (p, \sigma, \square, \square, p') \in \delta_1 \end{pmatrix} \vee \begin{pmatrix} \sigma = \lambda \\ \wedge p = p' \\ \wedge (q, \lambda, \gamma, s, q') \in \delta_2 \end{pmatrix} \right) \right\},$$

$q_{\times 0} = (q_{10}, q_{20})$ *and* $F_\times = F_1 \times F_2$.

As stated in [24, p.135], this product automaton is again a DPDA and its marked and unmarked language are given by

$$\mathcal{L}_m(M_P \times M_O) = \mathcal{L}_m(M_P) \cap \mathcal{L}_m(M_O) \quad \text{and} \tag{11.5a}$$
$$\mathcal{L}_{um}(M_P \times M_O) = \mathcal{L}_{um}(M_P) \cap \mathcal{L}_{um}(M_O). \tag{11.5b}$$

11.2. Manufacturing Example

This section introduces an example taken from [56] to illustrate the algorithm subsequently derived in Section 11.3-11.4 and to outline the practical relevance of the considered problem.

Consider a manufacturing process where two products A and B are manufactured on a machine sequentially. Assume that the sequence and the production rate of both products vary over time (because, e.g., the supply rate of different ingredients necessary to produce A and B is non-constant). The machine always produces two pieces of B at the same time. Therefore, the production of a pair of pieces B is modeled by a string B_1B_2, where B_1 is a controllable event, while B_2 is not controllable. The production of one piece A is represented by the controllable event A. Finally, in a different manufacturing step, which is not modeled here, one piece of A and one piece of B are assembled to form one piece of the final product C.

As A and B are perishable, all produced pieces should be used up (by assembling C) every day. Therefore, the external event r is triggered if the end of the working day is "close". After this event, the controller should balance the number of produced pieces by disabling the appropriate event. At the end of the working day, the external event F is triggered and all pieces which have not been used in the production of C are moved to the trash. A new process is started when the external event S occurs, indicating the beginning of the next working day.

This specification can be realized by the DPDA M_S depicted in Figure 11.3. There, the state q_1 represents the standard functionality, i.e., neither r nor F have occurred and the machine produces pieces of A and B. q_2 represents the balancing mode. The event F leads to q_3, modeling the end of this day's production process. Finally, state q_4 represents the dumping of unused pieces if balancing was aborted by F. In M_S, the stack is used to count the difference between the number of pieces of type A ($\#(A)$) and type B ($\#(B)$) produced, e.g., the stack content "$\bullet \bullet \bullet \square$" models $\#(A) - \#(B) = 3$ while "$\circ \circ \circ \square$" corresponds to $\#(B) - \#(A) = 3$. Therefore, balancing is achieved if the stack is exhausted (i.e., contains only "$\square$") before F occurs.

Recall that the plant can produce either one piece of A or two pieces of B sequentially, which can be modeled by the DFA $M_{P,1}$ depicted in Figure 11.4 (left). The external events r and F are included in $M_{P,1}$ by self-looping them in all states while the external event S resets the automaton to its initial state. However, the occurrence of r, F, and S is not arbitrary and can be modeled by the DFA $M_{P,2}$ as depicted in Figure 11.4 (middle). In $M_{P,2}$, the state p_I represents the standard production process. If r occurs, the machine is prepared to shut down while still producing pieces. The actual shut down is triggered by the event F, leading to the final state p_F, from which the machine is restarted by the external event S. The DFA modeling the overall plant behavior is the product $M_P = M_{P,1} \times M_{P,2}$, depicted in Figure 11.4 (right).

Observe, that in this control problem the external events r, F, S and the event B_2

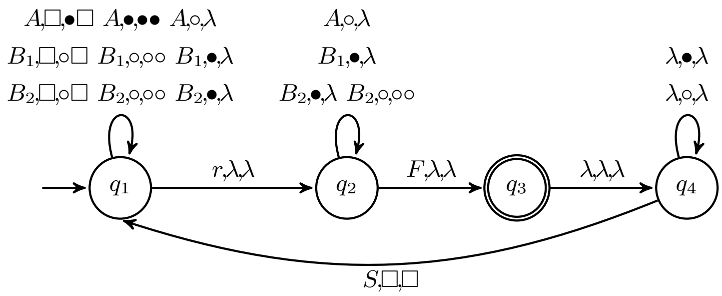

Figure 11.3.: DPDA M_S representing the specification in the manufacturing example of Section 11.2. Here the abbreviation σ,λ,λ := $\{\sigma,\gamma,\gamma | \gamma \in \Gamma\}$ is used to denote transitions not modifying the stack which are enabled for any stack-top.

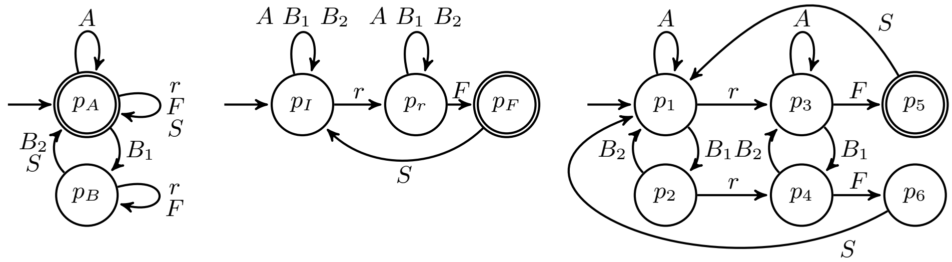

Figure 11.4.: DFA $M_{P,1}$ (left), DFA $M_{P,2}$ (middle) and DFA $M_P = M_{P,1} \times M_{P,2}$ (right) modeling the plant in the manufacturing example of Section 11.2.

cannot be prevented by the controller and are therefore uncontrollable, giving $\Sigma_{\mathsf{uc}} = \{B_2, r, F, S\}$ and $\Sigma_{\mathsf{c}} = \{A, B_1\}$.

Remark 11.1. The maximal mismatch of the number of pieces of type A and B during one day is unpredictable. If the maximal mismatch were known prior to controller design, one can obviously also realize D_S as a DFA coding the current mismatch in the DFA states. However, this might result in an automaton which has a very large number of states and transitions. In this case, as the specification is usually modeled using engineering intuition for the problem to be solved, writing down the specification as a DPDA is more intuitive and less prone to error. Furthermore, controller synthesis using DPDA specifications does not need to be redone if the maximal mismatch changes. Finally, storing a DPDA with finite stack usually requires less memory than storing a language-equivalent DFA. This benefit obviously grows with the maximal mismatch considered.
The latter benefit is also obtained when using discrete event systems with parameters, proposed in [5, 6]. While this modeling framework is similar to the use of DPDA, to the best of the authors' knowledge it can only handle safety specifications w.r.t. a set

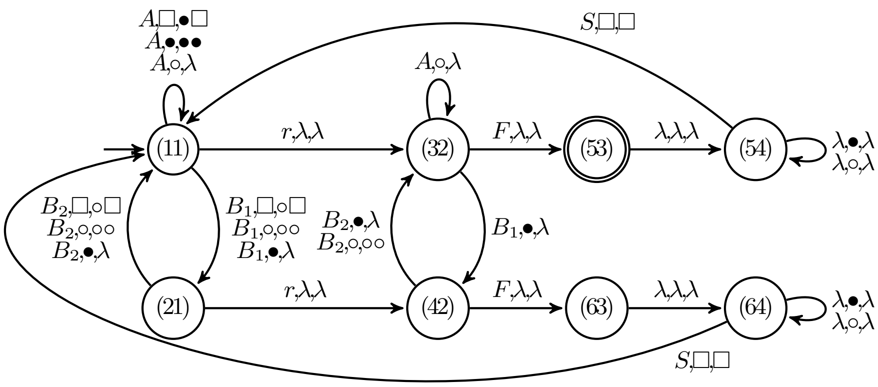

Figure 11.5.: DPDA $M_{P\times S} \asymp D_O = \mathrm{F_{nb}}(\inf(D_P, D_S))$ in the manufacturing example of Section 11.2, where $(ij) := (p_i, q_j)$.

of unsafe *states* $Q_{us} \subseteq Q$. This circumvents the need to handle stack-dependent controllability issues, which is necessary when using an unrestricted class of context free specification languages, as discussed in the remaining part of this section. ◁

Recall from Theorem 10.10 that the iterative calculation $\mathcal{U}$ is initialized with $\mathrm{F_{nb}}(\inf(D_P, D_S))$. Therefore, to obtain the first input to $\mathrm{F_c}$, we construct the product automaton of the DFA $M_P \asymp D_P$ and the DPDA $M_S \asymp D_S$ depicted in Figure 11.4 and Figure 11.3, respectively, using the construction in Definition 11.11. This results in the product automaton $M_{P\times S} = M_P \times M_S$ whose accessible part is depicted in Figure 11.5.

Observe that $M_{P\times S}$ is not silently blocking, i.e.,$M_{P\times S} \notin \mathrm{SBA}$, as the silent self-loops in states (p_5, q_4) and (p_6, q_4) will eventually terminate if the end-of-stack-marker $\square$ is reached. Furthermore, no blocking situations occur in $M_{P\times S}$ as in every state an outgoing transition exists for all stack-tops in $\Gamma = \{\bullet, \circ, \square\}$. As $M_{P\times S} \asymp \inf(D_P, D_S)$ we therefore have $\mathrm{NbLM}(\inf(D_P, D_S))$. This obviously implies

$$M_{P\times S} \asymp D_O = \mathrm{F_{nb}}(\inf(D_P, D_S)) = \inf(D_P, D_S).$$

Hence, the DPDA $M_{P\times S}$ depicted in Figure 11.5 realizes the input to the first call of $\mathrm{F_c}$ in the iterative algorithm $\mathcal{U}$ of Theorem 10.10.

11.3. Building Blocks of the Algorithm Calculating $\mathrm{F_c}$ for DCFL

This section is mostly taken from [56] and derives a chain of automata manipulations for DPDA to realize one call of the function $\mathrm{F_c}$ in the iterative algorithm $\mathcal{U}$ of Theorem 10.10 for DCFL. Given an LM D_O and its DPDA realization $M_O \asymp D_O$, the algorithm has to compute a DPDA M'_O realizing $\mathrm{F_c}(D_O)$, i.e., $M'_O \asymp \mathrm{F_c}(D_O)$. I.e., this algorithm has to find all controllability problems in M_O and delete them in a minimally restrictive way.

Intuitively, a controllability problem occurs if the maximal finite derivations of a string w for M_P and M_O end in the state p and q, respectively, and M_P can generate an uncontrollable event $\mu \in \Sigma_{\mathsf{uc}}$ in p while M_O cannot generate μ in q. Therefore, the first step of the algorithm needs to find pairs (p, q) that can be reached in M_P and M_O, respectively, generating the same string w.

Recall from Theorem 10.10 that $\mathrm{F_c}$ is first called with input $D_O = \mathrm{F_{nb}}(\inf(D_P, D_S))$ in $\mathcal{U}$, i.e., $\mathrm{NbLM}(D_O)$ and $D_O \leq \inf(D_P, D_S)$. Now observe that these properties of D_O are also true for the input of $\mathrm{F_c}$ in all future iterations of $\mathcal{U}$ as $\mathrm{F_{nb}}$ and $\mathrm{F_c}$ are monotone (from Lemma 10.9) and $\mathrm{F_{nb}}$ is always executed prior to $\mathrm{F_c}$ (from Theorem 10.10). We can therefore assume without loss of generality that $M_O \asymp D_O$ always satisfies $\mathcal{L}_m(M_O) \subseteq \mathcal{L}_m(D_P)$ and $\mathcal{L}_{um}(M_O) \subseteq \overline{\mathcal{L}_m(M_O)}$. Due to these properties, we can easily obtain the pairs (p, q) by calculating a product automaton $M_\times = M_P \times M_O$, which has the same marked and unmarked language as M_O. This is shown in the following lemma. Observe that the product construction is necessary in every iteration as the application of $\mathrm{F_{nb}}$ might significantly change the structure of its input (see [60]).

Lemma 11.12. *Let $M_\times = M_P \times M_O$ s.t. $M_P \in \mathrm{DFA}$, $M_O \in \mathrm{DPDA}$, $\mathcal{L}_m(M_O) \subseteq \mathcal{L}_m(M_P)$, $\mathcal{L}_{um}(M_O) \subseteq \overline{\mathcal{L}_m(M_O)}$ and $M_O \notin \mathrm{SBA}$.*
Then (i) $\mathcal{L}_m(M_\times) = \mathcal{L}_m(M_O)$, (ii) $\mathcal{L}_{um}(M_\times) = \mathcal{L}_{um}(M_O)$, (iii) $M_\times \notin \mathrm{SBA}$, and (iv) $\times$ is implementable.

Proof. (i) follows from $\mathcal{L}_m(M_O) \subseteq \mathcal{L}_m(M_P)$ and (11.5). $\mathcal{L}_{um}(M_O) \subseteq \overline{\mathcal{L}_m(M_O)}$ and $\overline{\mathcal{L}_{um}(M_P)} = \mathcal{L}_{um}(M_P)$ implies $\mathcal{L}_{um}(M_O) \subseteq \mathcal{L}_{um}(M_P)$, which, using (11.5), immediately proves (ii). λ-transitions in $M_\times$ are exactly copied from M_O implying (iii) since $M_O \notin \mathrm{SBA}$ and (iv) follows from [24, p.135]. $\square$

Example 11.4. Consider the manufacturing example in Section 11.2 and recall that this control problem does not require the application of $\mathrm{F_{nb}}$ to $\inf(D_P, D_S)$ (as the latter is nonblocking) implying $M_O = M_{P \times S}$ in the first call of $\mathrm{F_c}$. Hence, $M_\times = M_P \times M_O$ is, modulo the labeling of states, equivalent to $M_{P\times S}$ and therefore given by the DPDA in Figure 11.5. $\triangleleft$

Splitting States

Unfortunately, in contrast to the DFA algorithm in [78, Lemma 5.1], it is not possible to remove controllability problems in M_O in a minimally restrictive fashion by deleting states or edges in $M_\times$. This is due to the following observations: (i) it is possible that controllability problems occur in one state for a strict subset of possible stack-tops (e.g., as $B_2 \in \Sigma_{\mathsf{uc}}$, $M_\times = M_{P\times S}$ in Figure 11.5 has a controllability problem in (p_4, q_2) for stack-top $\square$ only). Therefore, removing this state may also delete controllable words. (ii) It is generally not possible to uniquely prevent a certain stack-top symbol in a given state by removing certain pre-transitions as an incoming transition can generate more than one stack-top[3] (e.g., the transition $((p_3, q_2), B_1, \bullet, \lambda, (p_4, q_2))$ in Figure 11.5 generates as stack-top the symbol which is currently in the stack underneath $\bullet$). (iii) Controllability problems are locally not detectable in states $(\tilde{p}, \tilde{q}) \in Q_\lambda$ as it is not possible to determine in $(\tilde{p}, \tilde{q})$ whether an uncontrollable event $\mu \in \Sigma_{\mathsf{uc}}$ generated by M_P in $\tilde{p}$ is also generated by $M_\times$ after a finite sequence of λ-transitions starting in $(\tilde{p}, \tilde{q})$ (e.g., determining whether in Figure 11.5 $S \in \Sigma_{\mathsf{uc}}$ can always be generated in (p_5, q_4) after all finite sequences of λ-transitions starting in (p_5, q_3)). If such a controllability problem occurs, it will be resolved at the final state $(\tilde{p}, \tilde{q}')$ of the λ-transition sequence (e.g. in (p_5, q_4)). However, observe that $(\tilde{p}, \tilde{q}) \in F_\times$ and $(\tilde{p}, \tilde{q}') \notin F_\times$ implies that the word $\tilde{w} \in \mathcal{L}_m(M_\times)$ with $((\tilde{p}, \tilde{q}), \tilde{w}, s), ((\tilde{p}, \tilde{q}'), \tilde{w}, s') \in \mathcal{C}_{\mathrm{reach}}(M_\times)$ (having a controllability problem) is not removed from the marked language by removing $(\tilde{p}, \tilde{q}')$ in $M_\times$ (e.g. as $(p_5, q_3) \in F_\times$ and $(p_5, q_4) \notin F_\times$ in Figure 11.5, all words marked by (p_5, q_3) are not removed from the marked language by removing (p_5, q_4)). To remove a word $w \in \mathcal{L}_m(M_\times)$ with a controllability problem from the marked language, we have to ensure that *only* its maximal derivation $f \in \mathcal{D}_{max}(M_\times, w)$ ends in a marking state.

To overcome these problems we split states and redirect transitions in a particular way, such that deleting certain states deletes *all* words $w \in \mathcal{L}_m(M_\times)$ *(and only those)* having a controllability problem. For this purpose we introduce four new state types: *regular* $(\cdot_r)$ and *special* $(\cdot_s)$ *main* states

$$\mathcal{M}_r(Q) := \{\langle q \rangle_r \mid q \in Q\} \text{ and } \mathcal{M}_s(Q) := \{\langle q \rangle_s \mid q \in Q\}$$

and *regular* $(\cdot_r)$ and *special* $(\cdot_s)$ *auxiliary* states

$$\mathcal{A}_r(Q, \Gamma) := \{\langle q, \gamma \rangle_r \mid q \in Q \wedge \gamma \in \Gamma\} \text{ and } \mathcal{A}_s(Q, \Gamma) := \{\langle q, \gamma \rangle_s \mid q \in Q \wedge \gamma \in \Gamma\},$$

where

$$\mathcal{M}(Q) = \mathcal{M}_r(Q) \cup \mathcal{M}_s(Q) \text{ and } \mathcal{A}(Q, \Gamma) = \mathcal{A}_r(Q, \Gamma) \cup \mathcal{A}_s(Q, \Gamma)$$

are the sets of all main and all auxiliary states, respectively. Hence, every state is split into two entities, i.e., a set of regular and a set of special states, each consisting of one main and $|\Gamma|$ auxiliary states. Using these new states, we define a function splitting states and redirecting transitions.

Definition 11.13. *Let* $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$. *Then the* split automaton $M_{Sp} =$ SPLIT(M) *is defined by* $M_{Sp} = (Q_{Sp}, \Sigma, \Gamma, \delta_{Sp}, q_{Sp\,0}, \Box, F_{Sp})$ *with* $Q_{Sp} = \mathcal{M}(Q) \cup$

[3] This is the reason why the algorithm presented in [20] does not give a minimally restrictive controller.

$\mathcal{A}(Q, \Gamma)$,

$$\begin{aligned}\delta_{Sp} = &\{(\langle p\rangle_r, \lambda, \gamma, \gamma, \langle p, \gamma\rangle_r) | p \in Q, \gamma \in \Gamma\} \\ &\cup \{(\langle p\rangle_s, \lambda, \gamma, \gamma, \langle p, \gamma\rangle_s) | p \in Q, \gamma \in \Gamma\} \\ &\cup \left\{ (\langle p, \gamma\rangle_r, \sigma, \gamma, s, \langle p'\rangle_r) \middle| \begin{pmatrix} (p, \sigma, \gamma, s, p') \in \delta \\ \wedge (p \notin F \vee \sigma \neq \lambda) \end{pmatrix} \right\} \\ &\cup \left\{ (\langle p, \gamma\rangle_r, \lambda, \gamma, s, \langle p'\rangle_s) \middle| (p, \lambda, \gamma, s, p') \in \delta \wedge p \in F \right\} \\ &\cup \left\{ (\langle p, \gamma\rangle_s, \lambda, \gamma, s, \langle p'\rangle_s) \middle| (p, \lambda, \gamma, s, p') \in \delta \right\} \\ &\cup \left\{ (\langle p, \gamma\rangle_s, \sigma, \gamma, s, \langle p'\rangle_r) \middle| (p, \sigma, \gamma, s, p') \in \delta \wedge \sigma \neq \lambda \right\},\end{aligned}$$

$q_{Sp\,0} = \langle q_0\rangle_r$ *and* $F_{Sp} = (\mathcal{A}_r(F, \Gamma) \cup \mathcal{A}_s(Q, \Gamma)) \setminus Q_\lambda(M_{Sp})$.

Example 11.5. Using the automaton $M_\times = M_{P\times S}$ depicted in Figure 11.5, we can construct its split version $M_{Sp} = \text{SPLIT}(M_\times)$ by applying Definition 11.13. To keep the illustration of this automaton reasonably simple, we split each state (p_i, q_j) into a regular and a special entity, denoted by $\{ij\}_r$ and $\{ij\}_s$, respectively, while only depicting the splitting of $\{42\}_r$, $\{54\}_s$ and $\{64\}_r$ into main and auxiliary states. The resulting simplified automaton is shown in Figure 11.6, where also all "obviously useless" entities (i.e., entities that are not connected by a path (ignoring the labels) to the initial state) were removed. Observe that the entity $\{53\}_r$ is not marking (while (p_5, q_3) was marking in $M_\times$) as it has only outgoing λ transitions which are redirected to the *special* entity $\{54\}_s$. Observe that $\langle(p_5, q_4), \bullet\rangle_s, \langle(p_5, q_4), \circ\rangle_s \in Q_\lambda(M_{Sp})$ and $\langle(p_5, q_4), \Box\rangle_s \notin Q_\lambda(M_{Sp})$, implying that the latter is marking while the former are not. ◁

Intuitively, the newly introduced auxiliary states separate outgoing transitions by their required stack-top, as shown in Example 11.5. Therefore, if a controllability problem occurs for one stack-top, we can delete the respective auxiliary state without deleting controllable words.

Furthermore, observe that all main states belong to $Q_\lambda(M_{Sp})$ while, due to determinism, auxiliary states can either belong to the set $Q_\lambda(M_{Sp})$ or do not have outgoing λ-transitions at all. This uniquely defines the subset of states (i.e., $\mathcal{A}(Q, \Gamma) \setminus Q_\lambda(M_{Sp})$) for which controllability can and must efficiently be tested.

The new special states are used to ensure that only states reached by maximal derivations of words $w \in \mathcal{L}_m(M_O)$ are marking. Observe that the construction in SPLIT ensures that final states of maximal derivations always are in the set $\mathcal{A}(Q, \Gamma) \setminus Q_\lambda(M_{Sp})$. Therefore, shifting the marking into those states ensures that uncontrollable words can be removed from the marked language by removing states in the set $\mathcal{A}(Q, \Gamma) \setminus Q_\lambda(M_{Sp})$. Before formally proving this statement we show that SPLIT does not change the marked and unmarked language of its input.

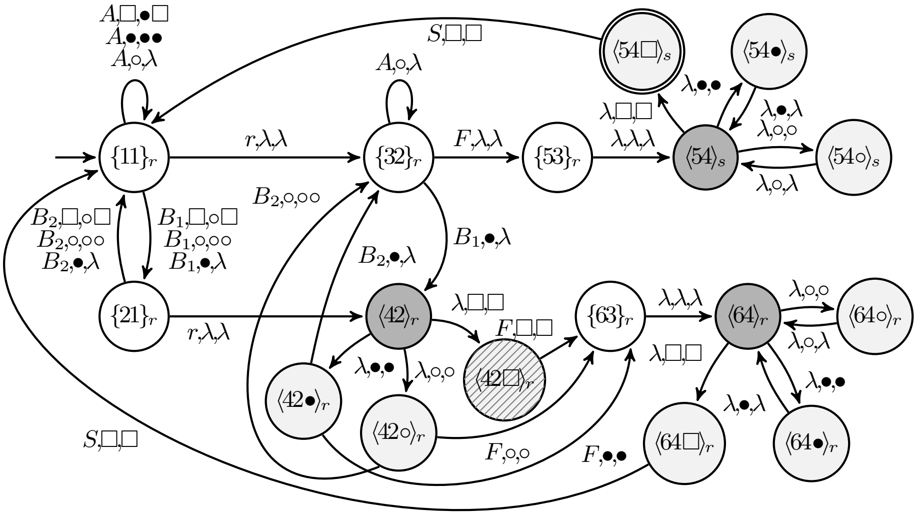

Figure 11.6.: A slightly simplified version of the DPDA $M_{Sp} = \mathrm{SPLIT}(M_{P\times S})$ constructed in Example 11.5, where $\{ij\}_k := \mathcal{A}_k((p_i, q_j)) \cup \mathcal{M}_k((p_i, q_j), \Gamma)$, $\langle ij\rangle_k := \langle (p_i, q_j)\rangle_k$ and $\langle ij\gamma\rangle_k := \langle (p_i, q_j), \gamma\rangle_k$. The introduced main and auxiliary states are marked in dark and light gray, respectively, and uncontrollable states are hatched.

Lemma 11.14. *Let* $M_{Sp} = \mathrm{SPLIT}(M)$ *s.t.* $M \in \mathrm{DPDA}$ *and* $M \notin \mathrm{SBA}$. *Then (i)* $\mathcal{L}_{um}(M_{Sp}) = \mathcal{L}_{um}(M)$, *(ii)* $\mathcal{L}_m(M_{Sp}) = \mathcal{L}_m(M)$, *(iii)* $M_{Sp} \in \mathrm{DPDA}$, *(iv)* $M_{Sp} \notin \mathrm{SBA}$, *and (v)* SPLIT *is implementable.*

Proof. See Section B.1.1 on page 157. □

Lemma 11.15. *Let* $M_{Sp} = \mathrm{SPLIT}(M)$ *s.t.* $\mathcal{L}_{um}(M) \subseteq \overline{\mathcal{L}_m(M)}$. *Then*

$$\forall\, w \in \mathcal{L}_{um}(M_{Sp}), f \in \mathcal{D}_{max}(M_{Sp}, w) \;.$$
$$\pi_1(\pi_2(f(\max(\mathrm{dom(f)})))) \in (\mathcal{A}(Q,\Gamma) \setminus Q_\lambda(M_{Sp}))$$

and $F_{Sp} \subseteq \mathcal{A}(Q,\Gamma) \setminus Q_\lambda(M_{Sp})$, *where* $\pi_1(\pi_2((e,(q,w,s)))) = q$.

Proof. Observe from Definition 11.13 that the second statement immediately follows from the definition of F_{Sp} and it holds that $\mathcal{M}(Q) \subseteq Q_\lambda(M_{Sp})$ and $Q_\lambda(M_{Sp}) \subseteq Q_{Sp} \setminus F_{Sp}$. Therefore it remains to show that maximal derivations cannot end in states $Q_\lambda(M_{Sp})$. Pick $f \in \mathcal{D}_{max}(M_{Sp}, w)$ s.t. $f(\max(\mathrm{dom(f)})) = (e,(q,w,s))$ and assume $q \in Q_\lambda(M_{Sp})$ implying $q \notin F_{Sp}$. As M is nonblocking it follows from Lemma 11.14 (i)-(ii) that M_{Sp} is also nonblocking. Therefore, there exist e', q', s' s.t. $(e,(q,w,s)) \vdash_{M_{Sp}} (e',(q',w,s'))$ implying that f is not a final derivation from Definition 11.3, proving the statement by contradiction. □

Accessibility

Technically, we are now ready to delete all states that have a controllability problem. However, the automaton $M_{Sp} = \mathrm{SPLIT}(M_P \times M_O) \asymp D_O$ might not be accessible, i.e., there may exist states and transitions that are not contained in a reachable configuration. This implies that the subsequently introduced function to remove states with a controllability problem may change the automaton without changing the corresponding language. Hence, we would not be able to decide on the automaton level, whether $\mathrm{F_c}(D_O) = D_O$. We therefore remove all non-accessible parts, adapting the algorithm in [18, Thm.4.1].

Definition 11.16. *Let* $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$*. Then the* accessible part $\mathrm{AC}(M)$ *of* M *is defined by* $M_{Ac} = (Q_{Ac}, \Sigma, \Gamma, \delta_{Ac}, q_0, \square, F_{Ac})$ *s.t.* $Q_{Ac} = Q \setminus \mathrm{Q_{na}}(M)$*,* $F_{Ac} = F \setminus \mathrm{Q_{na}}(M)$ *and* $\delta_{Ac} = \delta \setminus \delta_{\mathrm{na}}(M)$*, where*

$$\mathrm{Q_{na}}(M) := \{q \in Q \setminus \{q_0\} \mid \mathcal{L}_m((Q, \Sigma, \Gamma, \delta, q_0, \square, \{q\})) = \emptyset\}$$

are the non-accessible states and

$$\delta_{\mathrm{na}}(M) := \left\{ e = (q, x, \gamma, s, q') \in \delta \;\middle|\; \begin{array}{l} \forall\, r \notin Q, \delta' = (\delta \cup \{(q, \lambda, \gamma, \gamma, r)\}) \setminus \{e\}\, . \\ \mathcal{L}_m((Q \cup \{r\}, \Sigma, \Gamma, \delta', q_0, \square, \{r\})) = \emptyset \end{array} \right\}$$

are the non-accessible transitions.

Note that in a DPDA, in contrast to A DFA, transitions can be non-accessible even if they connect accessible states, since the required stack-top might not be available at its pre-state.

Example 11.6. Consider the automaton M_{Sp} in Example 11.5, depicted in Figure 11.6. It can be verified by inspection that all states and transitions are accessible, i.e., $M_{Ac} = \mathrm{AC}(M_{Sp}) = M_{Sp}$. This is due to the fact that all main states are accessible with all possible stack-tops $\{\bullet, \circ, \square\}$. This implies that all auxiliary states and their outgoing transitions are also accessible. ◁

Not surprisingly, removing the non-accessible part of a DPDA does not affect its marked and unmarked language.

Lemma 11.17. *Let* $M_{Ac} = \mathrm{AC}(M)$ *s.t.* $M \in \mathrm{DPDA}$ *and* $M \notin \mathrm{SBA}$*.*
Then (i) $\mathcal{L}_m(M_{Ac}) = \mathcal{L}_m(M)$*, (ii)* $\mathcal{L}_{um}(M_{Ac}) = \mathcal{L}_{um}(M)$*, (iii)* $M_{Ac} \in \mathrm{DPDA}$*, (iv)* $M_{Ac} \notin \mathrm{SBA}$*, and (v)* AC *is implementable.*

Proof. Observe that $\mathrm{Q_{na}}(M)$ and $\delta_{\mathrm{na}}(M)$ are not contained in a reachable configuration. Therefore, removing these sets does not change the marked and unmarked language, implying (i) and (ii). Since $M \in \mathrm{DPDA}$ and $M \notin \mathrm{SBA}$, (iii) and (iv) immediately follow as AC only removes states and transitions. (v) follows, as the emptiness of a CFL is decidable [see 24, p.137]. □

Removing Controllability Problems

As the final step of our algorithm, we will identify states in the set $\mathcal{A}(Q,\Gamma) \setminus Q_\lambda(M_{Sp})$ that have a controllability problem and remove them. This is done in analogy to the DFA algorithm in [78].

Definition 11.18. *Let $\Sigma_{\mathsf{uc}} \subseteq \Sigma$, $M_P \in$ DFA, $M_O \in$ DPDA, $M_\times = M_P \times M_O = (Q_\times, \Sigma, \Gamma, \delta_\times, q_{\times 0}, \square, F_\times)$, and $M_{Ac} = \mathrm{AC}(\mathrm{SPLIT}(M_\times)) = (Q_{Ac}, \Sigma, \Gamma, \delta_{Ac}, q_0, \square, F_{Ac})$. Furthermore, let*

$$\mathrm{NCS} := \left\{ \begin{array}{l} \langle (p,q),\gamma\rangle \in ((Q_{Ac} \cap \mathcal{A}(Q_\times,\Gamma)) \setminus Q_\lambda(M_{Ac})) \mid \\ \quad \exists \mu \in \Sigma_{\mathsf{uc}} \, . \begin{pmatrix} \exists p' \, . \, (p,\mu,\square,\square,p') \in \delta_P \\ \wedge \forall s,r \, . \, (\langle (p,q),\gamma\rangle, \mu, \gamma, s, r) \notin \delta_{Ac} \end{pmatrix} \end{array} \right\}$$

be the set of non-controllable auxiliary states.
Then $\mathrm{RNCS}(M_{Ac}) = (Q_R, \Sigma, \Gamma, \delta_R, q_0, \square, F_R)$, where $Q_R = Q_{Ac} \setminus \mathrm{NCS}$, $F_R = F_{Ac} \setminus \mathrm{NCS}$ and $\delta_R = \{(q,x,\gamma,w,q') \in \delta' | q,q' \in Q_R\}$ is the automaton without non-controllable auxiliary states*.*

Example 11.7. Consider the accessible automaton $M_{Ac} = M_{Sp}$ depicted in Figure 11.6 and recall that $\Sigma_{\mathsf{uc}} = \{F, r, S, B_2\}$. While F and r are always enabled in M_{Ac}, the uncontrollable event B_2 is prevented in $\langle (p_4,q_2),\square\rangle_r$, introducing a controllability problem in this state. For S observe that all auxiliary states related to p_5 and p_6 are contained in $Q_\lambda(M_{Ac})$ except for $\langle (p_5,q_4),\square\rangle_s$ and $\langle (p_6,q_4),\square\rangle_r$, where no controllability problem occurs. This implies that all finite sequences of λ-transitions starting in $\{53\}_r$ and $\{63\}_r$ will eventually end in $\langle (p_5,q_4),\square\rangle_s$ and $\langle (p_6,q_4),\square\rangle_r$, respectively, where S is enabled. Therefore, the set of uncontrollable auxiliary states of M_{Ac} is given by $\mathrm{NCS} = \{\langle (p_4,q_2),\square\rangle_r\}$, shown hatched in Figure 11.6. Removing this state and all ingoing and outgoing transitions generates the automaton $M'_O = \mathrm{RNCS}(M_{Ac})$. Observe that M'_O has a blocking situation in $\langle (p_4,q_2)\rangle_r$ for all words w whose derivations generate stack-top $\square$ in $\langle (p_2,q_2)\rangle_r$. This is resolved by the algorithm realizing the function $\mathrm{F_{nb}}$ in [60, 59]. $\triangleleft$

11.4. Effective Computability of $\mathrm{F_c}$ for DCFL

In the previous section three functions to manipulate DPDA, namely SPLIT, AC and RNCS where introduced. We now show in this section that this sequence of automaton manipulations, starting with a nonblocking DPDA, removes all (and only those) marked words having a prefix which is uncontrollable w.r.t. the plant and hence realizes one call of $\mathrm{F_c}$ in the iterative algorithm $\mathcal{U}$ of Theorem 10.10.

Recall from Theorem 10.10 that $\mathrm{F_c}$ is first called with input $D_O = \mathrm{F_{nb}}(\inf(D_P, D_S))$ in $\mathcal{U}$, i.e., $\mathrm{NbLM}(D_O)$ and $D_O \leq \inf(D_P, D_S)$. Now observe that these properties on D_O

are also true for the input of $\mathrm{F_c}$ in all future iterations of $\mathcal{U}$ as $\mathrm{F_{nb}}$ and $\mathrm{F_c}$ are monotone (from Lemma 10.9) and $\mathrm{F_{nb}}$ is always executed prior to $\mathrm{F_c}$ (from Theorem 10.10). We can therefore assume without loss of generality that $M_O \asymp D_O$ always satisfies $\mathrm{NbLM}(D_O)$ and $D_O \leq \inf(D_P, D_S)$in the following theorem.

Theorem 11.19. *Let $D_P, D_S, D_O \in \mathsf{LM}$ s.t. $\mathrm{NbLM}(D_O)$ and $D_O \leq \inf(D_P, D_S)$. Furthermore, let $M_P \in \mathrm{DFA}$, $M_O \in \mathrm{DPDA}$. Then*

$$\mathrm{RNCS}(\mathrm{AC}(\mathrm{SPLIT}(M_P \times M_O))) \asymp \mathrm{F_c}(D_O) \tag{11.6}$$

if $M_P \asymp D_P$ and $M_O \asymp D_O$.

Proof. See Section B.1.2 on page 159. □

Remark 11.2. Observe from steps (C) and (E)(a) in the proof of Theorem 11.19 in Section B.1.2 that unmarked words $w \in \mathcal{L}_{um}(M_O)$ with a controllability problem are still contained in the unmarked language of $M'_O = \mathrm{RNCS}(\mathrm{AC}(\mathrm{SPLIT}(M_P \times M_O)))$. This follows from the construction of SPLIT, which implies that there always exists a non-maximal derivation generating w and ending in a main state $\langle (p, q)\rangle \in Q_\lambda(M_{Ac})$ which is neither marking nor considered in NCS. Hence, this state cannot be removed by RNCS even if w has an uncontrollability problem. This is the reason why we cannot implement $\Psi'(\mathcal{L}_{um}(M_O))$, i.e., the removal of all uncontrollable *unmarked* words, as used by [78] (see Remark 10.2). ◁

Considering the problem statement from the beginning of this chapter, we also need to prove that the considered chain of automaton manipulations introduced in Section 11.3 introduces no silent blocking situations and allows to determine if $\mathrm{F_c}$ returns its input.

Theorem 11.20. *Let $M'_O = \mathrm{RNCS}(\mathrm{AC}(\mathrm{SPLIT}(M_P \times M_O)))$ s.t. $M_P \in \mathrm{DFA}$, $M_O \in \mathrm{DPDA}$ and $M_O \notin \mathrm{SBA}$. Then $M'_O \notin \mathrm{SBA}$.*

Proof. It follows from Lemma 11.12, 11.14 and 11.17 that $\mathrm{AC}(\mathrm{SPLIT}(M_P \times M_O)) \notin \mathrm{SBA}$. Now observe from Definition 11.18 that RNCS does obviously not introduce infinite λ-sequences as this function only removes existing states and transitions. □

Theorem 11.21. *Let $D_O, D'_O \in \mathsf{LM}$ s.t. $D'_O = \mathrm{F_c}(D_O)$. Then it is decidable whether $D_O = D'_O$.*

Proof. Pick $M_P \in \mathrm{DFA}$, $M_O \in \mathrm{DPDA}$ s.t. $M_P \asymp D_P$ and $M_O \asymp D_O$ and let $M'_O = \mathrm{RNCS}(\mathrm{AC}(\mathrm{SPLIT}(M_P \times M_O)))$. Then it follows from Theorem 11.19 that $M'_O \asymp D'_O$. Now let $M_{Ac} = \mathrm{AC}(\mathrm{SPLIT}(M_P \times M_O))$ and observe that it follows from Lemma 11.12, Lemma 11.14 and Lemma 11.17, that $\mathcal{L}_{um}(M_{Ac}) = \mathcal{L}_{um}(M_O)$ and $\mathcal{L}_m(M_{Ac}) = \mathcal{L}_m(M_O)$, implying $M_{Ac} \asymp D_O$. Now observe from Definition 11.18 that $M_{Ac} = M'_O$ (implying $D_O = D'_O$) iff $\mathrm{NCS} = \emptyset$, which is obviously decidable. □

Theorem 11.19 is the main result of Part II and shows that there exists an implementable algorithm to realize $\mathrm{F_c}$ for DCFL using their DPDA realizations. This algorithm has polynomial runtime as each of its components has polynomial runtime. In particular, the operations $\times$, SPLIT, and RNCS have quadratic runtime and the operation AC from [24, p.137] can be implemented with cubic runtime (see [60, 59]). We believe that significant complexity reductions for real world problems can be achieved by using distributed or hierarchical versions of the monolithic case treated in this work.

Conclusion

In this thesis we have built two bridges between results obtained by the control and the computer science community in abstraction-based controller synthesis.

In Part I we have compared finite state machine abstractions resulting from strongest asynchronous l-complete approximation (SAlCA) and quotient based abstraction (QBA). For this purpose we have first derived an asynchronous version of the existing abstraction technique of strongest l-complete approximations (SlCA). SlCA were introduced in [36] for (behaviorally) time invariant systems using a stronger version of the original l-completeness property defined in [76]. To resolve the resulting inconsistencies, and also to address a wider system class, the procedure suggested in [36] can be adapted in a straightforward way using the original l-completeness notion from [76], capturing also time variant systems. This, not surprisingly, leads to realizations with time dependent next state relations. To address this, we have proposed a new approximation technique, called strongest asynchronous l-complete approximation (SAlCA). This abstraction was shown to always be realizable by a state machine (SM) with time independent next state relation. For time invariant systems it produces the same approximation as [36], however, the mentioned inconsistencies are resolved.

As QBA are not restricted to (behaviorally) time invariant systems this generalization can be viewed as the first step to formally relate QBA and SAlCA. To obtain a common ground for this comparison we have furthermore introduced the notion of ϕ-dynamical systems, a modeling framework for systems with distinct (possibly continuous) internal time axis and (usually discrete) external time axis. By generalizing the asynchronous state property to ϕ-dynamical systems we have shown that, whenever this property holds, the external symbolic dynamics of a ϕ-dynamical system can be modeled by an asynchronous state space dynamical system (ASSDS) with unique discrete time axis $T = \mathbb{N}_0$. As we have shown that the step by step evolution of an ASSDS can be realized by an SM and both QBA and SAlCA are constructible from SM directly, this observation results in a unified setting to construct SAlCA and QBA.

To bridge the remaining differences between SAlCA and QBA, we have introduced a new parameter $m \in [0, l]$ to realize SAlCA by different state machines. We have shown that the choice $m = 0$ corresponds to relating states in the original state machine $\mathcal{Q}$ to their *strict l-long past* of external symbols, reproducing the standard realization of SAlCA. On the other hand, choosing $m = l$ corresponds to relating states in the original state machine $\mathcal{Q}$ to their *l-long future* of external symbols. We have shown that the construction of realizations for SAlCA with $m = l$ is closely related to the construction of QBA, if the latter is obtained from a partition resulting from l steps of the usual repartitioning algorithm. The latter was formalized by incorporating the refinement step into the construction of QBA resulting in a chain of different realizations with increasing precision.

Even if the latter observation renders both methods conceptually similar, they are generally incomparable if the system to be abstracted is equipped with a predefined set of output symbols. Only in the special case where the original system is future unique for l time steps both abstractions are identical up to a trivial renaming of states. Interestingly, a slightly weaker condition than future uniqueness implies that the realization of QBA resulting from the lth step of the repartitioning algorithm realizes the behavior of the SAlCA. This implies that there exist situations where a QBA realizes the SAlCA but does not coincide with a realization of SAlCA using l and m.

If the output symbol set is not predefined it can be arbitrarily chosen prior to the construction of SAlCA and QBA, as usually done when constructing QBA. In this scenario no further refinement of the obtained abstractions is necessary resulting in identical abstract state machines when using SAlCA and QBA and choosing $l = m = 1$.

In Part II we have described two steps in the generalization of SCT to situations where the plant language is regular but the specification language is deterministic context free. In this case, the SCP cannot be solved using the iterative functions derived in [78]. We have therefore discussed a slightly modified fixed point calculation $\mathcal{U}$ using the two basic functions $\mathrm{F_c}$ and $\mathrm{F_{nb}}$. This calculation is computable for the considered SCP, if, for the class of DCFL, (i) $\mathrm{F_c}$ is computable, (ii) $\mathrm{F_{nb}}$ is computable, (iii) it is decidable if $\mathrm{F_c}$ returns its input and (iv) $\mathcal{U}$ terminates.

Thereafter, we have presented an implementable algorithm realizing the function $\mathrm{F_c}$ to calculate the largest controllable marked sublanguage of a given DCFL. This algorithm consists of a sequence of automaton manipulations which, starting with a nonblocking DPDA, removes all (and only those) marked words having a prefix which is uncontrollable w.r.t. the plant. We have shown that this sequence of automaton manipulations realizes $\mathrm{F_c}$, does not introduce silent blocking and allows to decide if $\mathrm{F_c}$ returns its input. This implies that (i) and (iii) are true.

Concerning (ii), an algorithm realizing $\mathrm{F_{nb}}$ for DCFL was suggested in [60, 59] which was not discussed in this thesis. However, the overall algorithm solving the SCP for deterministic context free specification languages is available as a plug-in in `libFAUDES` [31]. Finally, characterizing the class of SCP for which (iv) can be guaranteed is left for future work. We have, however, not encountered termination problems in any realistic example.

A. Appendix Part I

A.1. Proofs for Chapter 6

In this section we provide detailed proofs for the theorems in Chapter 6. To simplify subsequent proofs we first translate the conditions for a transition in $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ which were given in Definition 6.3 in terms of transitions of $\mathcal{Q}$ into conditions of the domino game.

Lemma A.1. *Given* (6.1) *and* $\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}$ *as in Definition 6.3 it holds for all* $w \in W$ *and* $\widehat{x}, \widehat{x}' \in \widehat{X}^{\mathcal{I}^l_m}$ *that*

$$
\begin{aligned}
&(\widehat{x}, w, \widehat{x}') \in \widehat{\delta}^{\mathcal{I}^l_m} \\
&\Leftrightarrow \begin{pmatrix} \widehat{x}'|_{[0,l-m-1]} = \left(\widehat{x}|_{[0,l-m-1]} \cdot \pi_V(w)\right)|_{[1,l-m]} \\ \wedge \widehat{x}'|_{[l-m,l-2]} = \widehat{x}|_{[l-m+1,l-1]} \\ \wedge \widehat{x}|_{[0,l-m-1]} \cdot \pi_V(w) \cdot \widehat{x}'|_{[l-m,l-1]} \in D_{l+1} \end{pmatrix} && \text{(A.1a)} \\
&\Leftrightarrow \exists \zeta \in D_{l+1} \,.\, \begin{pmatrix} \zeta|_{[0,l-1]} = \widehat{x} \\ \wedge \zeta|_{[1,l]} = \widehat{x}' \\ \wedge \pi_V(w) = \zeta(l-m) \end{pmatrix} && \text{(A.1b)}
\end{aligned}
$$

Proof. "$\Rightarrow$":
• Observe from (6.7c) that $(\widehat{x}, w, \widehat{x}') \in \widehat{\delta}^{\mathcal{I}^l_m}$ iff the first two lines of the conjunction in (6.7c) are fulfilled and there exist $x, x' \in X$ s.t.

$$\exists (\nu, \xi) \in {}^{\diamond}\mathcal{B}_{SV}, k \,.\, \begin{pmatrix} \xi(k) = x \\ \wedge \widehat{x} = \nu|_{[k+m-l,k+m-1]} \end{pmatrix}, \tag{A.2a}$$

$$\exists (\nu', \xi') \in {}^{\diamond}\mathcal{B}_{SV}, k' \,.\, \begin{pmatrix} \xi'(k') = x' \\ \wedge \widehat{x}' = \nu'|_{[k'+m-l,k'+m-1]} \end{pmatrix}, \tag{A.2b}$$

$$\exists (\nu'', \xi'') \in {}^{\diamond}\mathcal{B}_{SV}, k'' \,.\, \begin{pmatrix} \xi''(k'') = x \\ \wedge \nu''(k'') = v \\ \wedge \xi''(k''+1) = x' \end{pmatrix}, \tag{A.2c}$$

Here (A.2a) and (A.2b) follow from (6.5) and (A.2c) follows from (2.20b).
• Now observe from (A.2) that $\xi(k) = x = \xi''(k'')$ and $\xi'(k') = x' = \xi''(k''+1)$. Using

(2.25) we therefore obtain

$$(\tilde{\nu},\tilde{\xi}) = (\nu,\xi) \wedge_{k''}^{k} (\nu'',\xi'') \wedge_{k'}^{k''+1} (\nu',\xi') \in {}^{\diamond}\mathcal{B}_{SV} \tag{A.2d}$$

giving

$$\begin{aligned}\tilde{\nu}|_{[k+m-l,k+m]} &= \nu|_{[k+m-l,k-1]} \cdot \nu''(k'') \cdot \nu'|_{[k',k'+m-1]} \\ &= \widehat{x}|_{[0,l-m-1]} \cdot v \cdot \widehat{x}'|_{[l-m,l-1]},\end{aligned}$$

and therefore, as $\tilde{\nu}|_{[k+m-l,k+m]} \in {}^{\diamond}\mathcal{B}_V|_{[k+m-l,k+m]}$ (using (6.1d)) we have $\widehat{x}|_{[0,l-m-1]} \cdot \pi_V(u,y) \cdot \widehat{x}'|_{[l-m,l-1]} \in D_{l+1}$.

• Now let $\zeta = \tilde{\nu}|_{[k+m-l,k+m]} \in D_{l+1}$ and observe that $v = \zeta(l-m)$. With this choice of ζ the first two lines of the conjunction in (6.7c) immediately imply $\zeta|_{[0,l-1]} = \widehat{x}$ and $\zeta|_{[1,l]} = \widehat{x}'$.

"$\Leftarrow$":

• Pick $\zeta \in D_{l+1}$ and $w, \widehat{x}$ and $\widehat{x}'$ s.t. the right side of (A.1b) holds.

• It is easy to see that the first two lines of the conjunction in (A.1a) and (6.7c) hold with this choice and $\zeta = \widehat{x}|_{[0,l-m-1]} \cdot v \cdot \widehat{x}'|_{[l-m,l-1]} \in D_{l+1}$.

• Now using (6.1d) there exist $(\tilde{\nu},\tilde{\xi})$ and $\tilde{k} \in \mathbb{N}_0$ s.t. $\zeta = \tilde{\nu}|_{[\tilde{k}+m-l,\tilde{k}+m-1]}$.

• Now we can choose all signals in (A.2a) and (A.2b) equivalent to $(\tilde{\nu},\tilde{\xi})$ and $x = \tilde{\xi}(\tilde{k})$ as well as $x' = \tilde{\xi}(\tilde{k}+1)$, giving $\widehat{x} \in \mathrm{E}^{\mathcal{I}_m^l}(x)$, $\widehat{x}' \in \mathrm{E}^{\mathcal{I}_m^l}(x')$ and $(x,w,x') \in \delta$, hence the last line of the conjunction in (6.7c) holds. $\square$

A.1.1. Proof of Theorem 6.4

First observe that $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ is live and reachable by definition. Using Definition 2.22 we therefore only prove both directions of the behavior inclusion separately:

1.) Show $\widehat{\mathcal{B}}_V^{l\uparrow} \subseteq \pi_V(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}))$:

Pick $\widehat{\nu} \in {}^{\diamond}\widehat{\mathcal{B}}_V^{l\uparrow}$, $\widehat{\omega}$ s.t. $\pi_V(\widehat{\omega}) = \widehat{\nu}|_{[0,\infty)}$ and $\widehat{\xi}$ s.t. $\forall k \in \mathbb{N}_0 \;.\; \widehat{\xi}(k) = \widehat{\nu}|_{[k-l+m,k+m-1]}$. Then we prove both lines in (2.20b) separately:

• Show $\widehat{\xi}(0) \in \widehat{X}_0^{\mathcal{I}_m^l}$:

Using the same argument as in the proof of Proposition 6.1 (hence, dominos can only have diamonds when they are obtained from the initial behavior) we know that $\widehat{\nu}|_{[m-l-1,m-1]} \in {}^{\diamond}\mathcal{B}_V|_{[m-l-1,m-1]}$ for $m \in [0,l]$. Hence there exists $(\nu',\xi') \in \mathcal{B}_{SV}$ s.t. $\nu'|_{[m-l-1,m-1]} = \widehat{\nu}|_{[m-l-1,m-1]}$. Using (6.5) this implies that $\widehat{\nu}|_{[m-l,m-1]} \in \mathrm{E}^{\mathcal{I}_m^l}(\xi'(0))$ with $\xi'(0) \in X_0$, hence $\widehat{\xi}(0) \in \widehat{X}_0^{\mathcal{I}_m^l}$ (from (6.7b)).

• Show $\forall k \in \mathbb{N}_0 \;.\; (\widehat{\xi}(k), \widehat{\omega}(k), \widehat{\xi}(k+1)) \in \widehat{\delta}^{\mathcal{I}_m^l}$:

As (6.2) implies $\forall k \in \mathbb{N}_0 \;.\; \widehat{\nu}|_{[k-l+m,k+m]} \in D_{l+1}$, this follows immediately from the choice of $\widehat{\xi}$ and (A.1b).

2.) Show $\pi_V(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m})) \subseteq \widehat{\mathcal{B}}_V^{l\uparrow}$:
Pick $(\widehat{\omega}, \widehat{\xi}) \in \mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m})$ and $\widehat{\nu} \in (V \cup \{\diamond\})^{\mathbb{N}_0}$ s.t. $\pi_V(\widehat{\omega}) = \widehat{\nu}|_{[0,\infty)}$ and $\forall k < 0 \,.\, \widehat{\nu}(k) = \diamond$. To show $\widehat{\nu} \in {}^\diamond\widehat{\mathcal{B}}_V^{l\uparrow}$, observe that the first line of (6.2) holds by construction and the second line of (6.2) follows directly from (A.1b) and the second line of (2.20b), if we pick $\widehat{\xi}$ accordingly.

A.1.2. Proof of Theorem 6.5

• First observe from (6.1c), (6.5) and (6.7a) that

$$\widehat{X}^{\mathcal{I}^l_m} = \bigcup_{k\in\mathbb{N}_0} {}^\diamond\mathcal{B}_V|_{[k-l+m,k+m-1]} \subseteq \{\diamond\}^l \cup D_l \tag{A.3}$$

where equality only holds for $m = 0$, hence $\widehat{X}^{\mathcal{I}^l_0} = \{\diamond\}^l \cup D_l$. Furthermore, it can be observed from (6.5) and (6.7b) that

$$\widehat{X}_0^{\mathcal{I}^l_m} = {}^\diamond\mathcal{B}_V|_{[-l+m,m-1]}, \text{ hence } \widehat{X}_0^{\mathcal{I}^l_0} = {}^\diamond\mathcal{B}_V|_{[-l,-1]} = \{\diamond\}^l. \tag{A.4}$$

• It is therefore easy to see that $\widehat{x} \in \widehat{X}_0^l = \{\lambda\}$ implies ${}^\diamond\widehat{x} = \diamond^l \in \widehat{X}_0^{\mathcal{I}^l_0} = \{\diamond\}^l$ and vice versa, hence (5.6a) holds for $\mathcal{R}$ and $\mathcal{R}^{-1}$.
• Now recall from the proof of Proposition 6.1 that for the choice of $V = W$ it holds that $\overline{\mathcal{B}} = \mathcal{B}_V$. Using this and (2.42c) we obtain

$$\begin{aligned}
&(\widehat{x}, w, \widehat{x}') \in \widehat{\delta}^{l\uparrow} \\
\Leftrightarrow &\begin{pmatrix} \widehat{x}' = \widehat{x} \cdot w \\ \wedge |\widehat{x}|_L < l \\ \wedge \widehat{x} \cdot w \in \mathcal{B}_V|_{[0,|\widehat{x}|_L]} \end{pmatrix} \vee \begin{pmatrix} \widehat{x}' = (\widehat{x} \cdot w)|_{[1,l-1]} \\ \wedge |\widehat{x}|_L = l \\ \wedge \widehat{x} \cdot w \in \bigcup_{k'\in\mathbb{N}_0} \mathcal{B}_V|_{[k',k'+l]} = D_{l+1} \end{pmatrix} \\
\Leftrightarrow &\begin{pmatrix} \widehat{x}' = \widehat{x} \cdot w \\ \wedge \exists r > 0 \,.\, \diamond^r \cdot \widehat{x} \cdot w \in \mathcal{B}_V|_{[-r,l-r]} \end{pmatrix} \vee \begin{pmatrix} \widehat{x}' = (\widehat{x} \cdot w)|_{[1,l-1]} \\ \wedge |\widehat{x}|_L = l \\ \wedge \widehat{x} \cdot w \in D_{l+1} \end{pmatrix}
\end{aligned}$$

• Now observe that $|\widehat{x}|_L = l$ implies ${}^\diamond\widehat{x} = \widehat{x}$ and $|\widehat{x}|_L < l$ with ${}^\diamond\widehat{x} = \diamond^r \cdot \widehat{x} \in \bigcup_{k'\in\mathbb{N}_0} \mathcal{B}_V|_{[k',k'+l]} = D_{l+1}$ implies ${}^\diamond\widehat{x} \in \mathcal{B}_V|_{[-r,l-r-1]}$. We therefore obtain

$$(\widehat{x}, w, \widehat{x}') \in \widehat{\delta}^{l\uparrow} \Leftrightarrow \begin{pmatrix} {}^\diamond\widehat{x}' = ({}^\diamond\widehat{x} \cdot w)|_{[1,l-1]} \\ \wedge {}^\diamond\widehat{x} \cdot w \in D_{l+1} \end{pmatrix} \Leftrightarrow ({}^\diamond\widehat{x}, w, {}^\diamond\widehat{x}') \in \widehat{\delta}^{\mathcal{I}^l_0}$$

where the last equality follows from (A.1b). Using this, (5.6b) obviously holds for $\mathcal{R}$ and $\mathcal{R}^{-1}$. □

A.1.3. Proof of Theorem 6.7

We prove the statement by applying Definition 5.7 and Theorem 5.8. Therefore, we first construct the $\phi_{\mathbf{i}}$-extensions of the involved models. Using Figure 6.1 and Figure 5.4 gives the following ASSϕDS.

$$\begin{aligned}
[\mathcal{Q}]^{\phi_{\mathbf{i}}} &= (T, T, W \times X, W, \phi_{\mathbf{i}}(\mathcal{B}_f(\mathcal{Q}))) \\
\left[\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}\right]^{\phi_{\mathbf{i}}} &= (T, T, W \times \widehat{X}^{\mathcal{I}^l_m}, W, \phi_{\mathbf{i}}(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m})) \\
\breve{\Sigma}^{\phi}_S &= (\breve{T}, T, \breve{W} \times \breve{X}, W, \phi_S) \\
\left[\widehat{\Sigma}^{l\uparrow}_S\right]^{\phi_{\mathbf{i}}} &= (T, T, W \times \widehat{X}^{\mathcal{I}^l_m}, W, \phi_{\mathbf{i}}(\mathcal{B}_f(\mathcal{Q})))
\end{aligned}$$

First, recall that $\mathcal{Q}$ is constructed from $\breve{\Sigma}^{\phi}_S$ as depicted in Figure 6.4 implying that the external reachable state space $\breve{X}^{\phi}$ of $\breve{\Sigma}^{\phi}_S$ coincides with the state space X of $\mathcal{Q}$. As $\left[\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}\right]^{\phi_{\mathbf{i}}}$ and $\left[\widehat{\Sigma}^{l\uparrow}_S\right]^{\phi_{\mathbf{i}}}$ also share the same (externally reachable) state space, (5.2a) always holds for all implications in (6.9). Therefore we only prove that the various implications hold for (5.2b).

1.) Show $\mathcal{R} \in \mathfrak{S}_V([\mathcal{Q}]^{\phi_{\mathbf{i}}}, \left[\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}\right]^{\phi_{\mathbf{i}}}) \Rightarrow \mathcal{R} \in \mathfrak{S}_V(\breve{\Sigma}^{\phi}_S, \left[\widehat{\Sigma}^{l\uparrow}_S\right]^{\phi_{\mathbf{i}}})$:

- Pick $(x, \widehat{x}) \in \mathcal{R}$, $\left((\omega, \breve{\xi}), \omega, \theta\right) \in \phi_S$, $k_1 \in \mathbb{N}_0$ $\;s.t.\;$ $\breve{\xi}(\theta^{-1}(k_1)) = x$.
- First observe that (4.7) implies the existence of ξ s.t.

$$\forall k \in T \,.\, \xi(k) = \breve{\xi}(\theta^{-1}(k)) \quad \text{and} \quad (\omega, \xi) \in \mathcal{B}_S.$$

- As $\mathcal{Q}$ is induced by Σ_S we know from Lemma 2.20 and Lemma 2.4 (i) that $\mathcal{B}_f(\mathcal{Q}) = \overline{\mathcal{B}_S} \supseteq \mathcal{B}_S$, hence $(\omega, \xi) \in \mathcal{B}_f(\mathcal{Q})$.
- Therefore, we know from (5.5) that $((\omega, \xi), \omega, \theta_{\mathbf{i}}) \in \phi_{\mathbf{i}}(\mathcal{B}_f(\mathcal{Q}))$ and $\xi(k_1) = x$.
- Now recall that $\mathcal{R} \in \mathfrak{S}_V([\mathcal{Q}]^{\phi_{\mathbf{i}}}, \left[\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}\right]^{\phi_{\mathbf{i}}})$ and $(x, \widehat{x}) \in \mathcal{R}$. Hence, the above observation implies the existence of $\left((\widehat{\omega}, \widehat{\xi}), \widehat{\omega}, \theta_{\mathbf{i}}\right) \in \phi_{\mathbf{i}}(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}))$ s.t. the right side of (5.2b) holds.
- As $\pi_V(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m})) = \widehat{\mathcal{B}}^{l\uparrow}_V$ (from Theorem 6.4), this immediately implies that there exits $\widehat{\omega}' \in W^{\mathbb{N}_0}$ s.t. $\left((\widehat{\omega}', \widehat{\xi}), \widehat{\omega}', \theta_{\mathbf{i}}\right) \in \phi_{\mathbf{i}}(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}))$ and $\pi_V(\widehat{\omega}) = \pi_V(\widehat{\omega}')$.
- With this it immediately follows that the right side of (5.2b) holds for this choice of signals, which proves the statement.

2.) Show $\mathcal{R} \in \mathfrak{S}_V(\breve{\Sigma}^{\phi}_S, \left[\widehat{\Sigma}^{l\uparrow}_S\right]^{\phi_{\mathbf{i}}}) \Rightarrow \mathcal{R} \in \mathfrak{S}_V(\mathcal{Q}, \widehat{\mathcal{Q}}^{\mathcal{I}^l_m})$:

- Pick $(x, \widehat{x}) \in \mathcal{R}, w \in W, x' \in X_1$ $\;s.t.\;$ $(x, w, x') \in \delta$.
- As $\mathcal{Q}$ is induced by Σ_S, using (2.21) we can pic $(\omega, \xi) \in \mathcal{B}_S$ and $k_1 \in \mathbb{N}_0$ s.t. $x' = \xi(k_1+1)$ and $w = \omega(k_1)$.

- Now (4.7) implies the existence of $\breve{\xi}$ s.t.

$$\forall k \in \mathbb{N}_0 \,.\, \xi(k) = \breve{\xi}({\theta_1}^{-1}(k)) \quad \text{and} \quad \Big((\omega, \breve{\xi}), \omega, \theta\Big) \in \phi_S.$$

- Now recall that $\mathcal{R} \in \mathfrak{S}_V(\breve{\Sigma}_S^{\phi}, \left[\widehat{\Sigma}_S^{l\uparrow}\right]^{\phi_{\mathbf{i}}})$ and $(x, \widehat{x}) \in \mathcal{R}$. Hence, the above observation implies the existence of $\Big((\widehat{\omega}, \widehat{\xi}), \widehat{\nu}, \theta_{\mathbf{i}}\Big) \in \phi_{\mathbf{i}}(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}))$ s.t. the right side of (5.2b) holds.
- As $\widehat{\Sigma}_S^{l\uparrow}$ is realized by $\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}$ w.r.t. V (from Theorem 6.4) this implies that we can pick $\widehat{x}' = \widehat{\xi}(k_2 + 1)$ and w' s.t. $\pi_V(w') = \pi_V(\widehat{\omega}(k_2))$ s.t. the right side of (5.6b) holds, what proves the statement.

3.) Show $\mathcal{R} \in \mathfrak{S}_V(\left[\widehat{\Sigma}_S^{l\uparrow}\right]^{\phi_{\mathbf{i}}}, \breve{\Sigma}_S^{\phi}) \Rightarrow \mathcal{R} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}, \mathcal{Q})$:
- Pick $(\widehat{x}, x) \in \mathcal{R}, w \in W, \widehat{x}' \in \widehat{X}^{\mathcal{I}_m^l} \quad s.t. \quad (\widehat{x}, w, \widehat{x}') \in \widehat{\delta}^{\mathcal{I}_m^l}$
- As the behavior of $\widehat{\Sigma}_S^{l\uparrow}$ is given by $\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l})$ from Proposition 6.6 we can pick $(\widehat{\omega}, \widehat{\xi}) \in \mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l})$ and k_1 s.t. $\widehat{x}' = \widehat{\xi}(k_1 + 1)$ and $w = \widehat{\omega}(k_1)$.
- Now (5.5) implies $\Big((\widehat{\omega}, \widehat{\xi}), \widehat{\omega}, \theta_{\mathbf{i}}\Big) \in \phi_{\mathbf{i}}(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}))$.
- Now recall that $\mathcal{R} \in \mathfrak{S}_V(\left[\widehat{\Sigma}_S^{l\uparrow}\right]^{\phi_{\mathbf{i}}}, \breve{\Sigma}_S^{\phi})$ and $(\widehat{x}, x) \in \mathcal{R}$. Hence, the above observation implies the existence of $\Big((\omega, \breve{\xi}), \omega, \theta\Big) \in \phi_S$ s.t. the right side of (5.2b) holds.
- As $\mathcal{Q}$ is induced by $\breve{\Sigma}_S^{\phi}$ this implies that we can pick $x' = \xi(k_2+1)$ and $w' = \omega(k_2)$ s.t. the right side of (5.6b) holds, what proves the statement.

4.) Show Σ_S is complete $\Rightarrow \left(\mathcal{R} \in \mathfrak{S}_V(\left[\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}\right]^{\phi_{\mathbf{i}}}, [\mathcal{Q}]^{\phi_{\mathbf{i}}}) \Rightarrow \mathcal{R} \in \mathfrak{S}_V(\left[\widehat{\Sigma}_S^{l\uparrow}\right]^{\phi_{\mathbf{i}}}, \breve{\Sigma}_S^{\phi})\right)$:
- Pick $(\widehat{x}, x) \in \mathcal{R}, \ \Big((\widehat{\omega}, \widehat{\xi}), \widehat{\omega}, \theta_{\mathbf{i}}\Big) \in \phi_{\mathbf{i}}(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}))\ ,\ k_1 \in \mathbb{N}_0 \quad s.t. \quad \widehat{\xi}(k_1) = \widehat{x}$.
- Using Proposition 6.6 this implies $(\widehat{\omega}, \widehat{\xi}) \in \mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l})$.
- Therefore, we know from (5.5) that $\Big((\widehat{\omega}, \widehat{\xi}), \widehat{\omega}, \theta_{\mathbf{i}}\Big) \in \phi_{\mathbf{i}}(\mathcal{B}_f(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}))$ and $\widehat{\xi}(k_1) = \widehat{x}$.
- Now recall that $\mathcal{R} \in \mathfrak{S}_V(\left[\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}\right]^{\phi_{\mathbf{i}}}, [\mathcal{Q}]^{\phi_{\mathbf{i}}})$ and $(\widehat{x}, x) \in \mathcal{R}$. Hence, the above observation implies that the existence of $((\omega, \xi), \omega, \theta_{\mathbf{i}}) \in \phi_{\mathbf{i}}(\mathcal{B}_f(\mathcal{Q}))$ s.t. the right side of (5.2b) holds.
- As Σ_S is complete and $\mathcal{Q}$ is induced by Σ_S we know that $\mathcal{B}_S = \mathcal{B}_f(\mathcal{Q})$, hence $(\omega, \xi) \in \mathcal{B}_S$.
- Now (4.7) implies the existence of $\breve{\xi}$ s.t.

$$\forall k \in \mathbb{N}_0 \,.\, \xi(k) = \breve{\xi}({\theta_1}^{-1}(k)) \quad \text{and} \quad \Big((\omega, \breve{\xi}), \omega, \theta\Big) \in \phi_S.$$

- With this it immediately follows that the right side of (5.2b) holds for this choice of signals, which proves the statement. $\square$

A.1.4. Proof of Theorem 6.10

We show both statements separately.

(i) $\mathcal{R} \in \mathfrak{S}_W(\mathcal{Q}, \widehat{\mathcal{Q}}^{\mathcal{I}^l_m}) \Leftrightarrow \mathcal{Q}$ is future unique w.r.t. $\mathcal{I}^l_m$.

• We first show that (5.6a) always holds for $\mathcal{R}$:
Pick $x \in X_0$. Then it follows from (2.23) and (6.5) that there exists $\zeta \in D_l$ s.t. $\zeta \in \mathrm{E}^{\mathcal{I}^l_m}(x)$. Using (6.7b) we get $\zeta \in \widehat{X}_0^{\mathcal{I}^l_m}$, which proves the statement.

• It remains to show that (5.6b) holds for $\mathcal{R}$ and W iff $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}^l_m$. We show both directions separately:

"$\Leftarrow$" : We first assume that $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}^l_m$ and prove the implication in (5.6b).
– Pick $(x, \widehat{x}) \in \mathcal{R}$, i.e., $\widehat{x} \in \mathrm{E}^{\mathcal{I}^l_m}(x)$ and w, x' s.t. $(x, w, x') \in \delta$.
– It follows from (6.5) and (2.20b) that (A.2a) and (A.2c) holds, hence

$$(\tilde{\nu}, \tilde{\xi}) = (\nu, \xi) \wedge^k_{k''} (\nu'', \xi'') \in \mathcal{B}_{SV}, \tag{A.5}$$

$$\begin{aligned} \text{where} \quad & \tilde{\xi}(k) = x && \text{(A.6a)}\\ & \tilde{\xi}(k+1) = x' && \text{(A.6b)}\\ & \tilde{\nu}|_{[k+m-l,k-1]} = \nu|_{[k+m-l,k-1]} && \text{(A.6c)}\\ & \tilde{\nu}(k) = \pi_V(w) = v && \text{(A.6d)} \end{aligned}$$

– Now pick $\widehat{x}' = \tilde{\nu}|_{[k+m-l+1,k+m]}$ and observe that (A.6b) implies $\widehat{x}' \in \mathrm{E}^{\mathcal{I}^l_m}(x')$.
– Furthermore, (A.6a) implies $\tilde{\nu}|_{[k+m-l,k+m-1]} \in \mathrm{E}^{\mathcal{I}^l_m}(x)$ and using (6.6) and (A.6c) therefore yields $\widehat{x} = \nu|_{[k+m-l,k+m-1]} = \tilde{\nu}|_{[k+m-l,k+m-1]}$, hence $\widehat{x}|_{[1,l-1]} = \widehat{x}'|_{[0,l-2]}$.
– With this, using (A.6d) immediately shows that (6.7c) holds, hence $(\widehat{x}, w, \widehat{x}') \in \widehat{\delta}^{\mathcal{I}^l_m}$.

"$\Rightarrow$" :
– Observe that (5.6b) holds for $\mathcal{R}$ and W iff

$$\forall x, x', \widehat{x}, w \;.\; \begin{pmatrix} \widehat{x} \in \mathrm{E}^{\mathcal{I}^l_m}(x) \\ \wedge (x, w, x') \in \delta \end{pmatrix} \Rightarrow \left(\exists \widehat{x}' \in \mathrm{E}^{\mathcal{I}^l_m}(x') \;.\; (\widehat{x}, w, \widehat{x}') \in \widehat{\delta}^{\mathcal{I}^l_m}\right) \tag{A.7}$$

– Recall that $\mathcal{Q}$ is always future unique w.r.t. $\mathcal{I}^l_m$ if $m = 0$. Therefore, we assume $m > 0$ and show that (6.6) holds if (A.7) holds.

– Pick $x \in X$ and $\zeta, \zeta' \in \mathrm{E}^{\mathcal{I}^l_m}(x)$.
– Using (6.5), there exist $(\nu, \xi) \in \mathcal{B}_{SV}$ and k s.t. $x = \xi(k)$ and $\zeta' = \nu|_{[k+m-l,k+m-1]}$.
– Now pick $x' = \xi(k+1)$ and w s.t. $\pi_V(w) = \nu(k) = \zeta'(l-m)$ and observe that $(x, w, x') \in \delta$ (hence we can apply (A.7) with $\widehat{x} = \zeta$) and

$$\tilde{\zeta} = \zeta'|_{[1,l-1]} \cdot \nu(k+m) \in \mathrm{E}^{\mathcal{I}^l_m}(x'). \tag{A.8}$$

– First observe that the right side of (A.7) implies $\widehat{x}(l-m) = \zeta(l-m) = \pi_V(w)$ (from (6.7c) and $m > 0$), hence $\zeta(l-m) = \zeta'(l-m)$.
– Furthermore, we know that there exists $\zeta'' \in \mathrm{E}^{\mathcal{I}^l_m}(x')$ s.t. $(\zeta, w, \zeta'') \in \widehat{\delta}^{\mathcal{I}^l_m}$, hence (from (6.7c)) $\zeta|_{[1,l-1]} = \zeta''|_{[0,l-2]}$.
– Using (A.8) and the same reasoning as before (substituting x by x' and ζ, ζ' by $\tilde{\zeta}, \zeta''$) we immediately obtain $\zeta''(l-m) = \tilde{\zeta}(l-m) = \zeta'(l-m+1)$.
– As $\zeta|_{[1,l-1]} = \zeta''|_{[0,l-2]}$ we therefore have $\zeta(l-m+1) = \zeta'(l-m+1)$.
– Applying this process iteratively therefore yields $\zeta|_{[l-m,l-1]} = \zeta'|_{[l-m,l-1]}$, what proves the statement.

(ii) $\mathcal{R}^{-1} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}, \mathcal{Q}) \Leftrightarrow \mathcal{Q}$ is state-based async. l-compl.

• First observe that (5.6a) always holds for $\mathcal{R}^{-1}$, as we can pick $\widehat{x} \in \widehat{X}_0^{\mathcal{I}^l_m}$ and (6.7b) implies the existence of $x \in X_0$ s.t. $\zeta \in \mathrm{E}^{\mathcal{I}^l_m}(x)$.

• It remains to show that (5.6b) holds for $\mathcal{R}^{-1}$ and Y iff $\mathcal{Q}$ is state-based asynchronously l-complete w.r.t. $\mathcal{I}^l_m$. We show both directions separately:

"$\Leftarrow$" :
– Pick $(\widehat{x}, x) \in \mathcal{R}^{-1}$, i.e., $\widehat{x} \in \mathrm{E}^{\mathcal{I}^l_m}(x)$ and $w, \widehat{x}'$ s.t. $(\widehat{x}, w, \widehat{x}') \in \widehat{\delta}^{\mathcal{I}^l_m}$. Then it follows from (A.1b) that there exists $\zeta \in D_{l+1}$ s.t. $\zeta|_{[0,l-1]} = \widehat{x}$, $\zeta|_{[1,l]} = \widehat{x}'$ and $\pi_Y(w) = \zeta(l-m)$. Using (6.11) this implies that $\zeta \in \mathrm{E}^{[m-l,m]}(x)$, hence there exists $(\nu, \xi) \in \mathcal{B}_{SV}$ and $k \in \mathbb{N}_0$ s.t. $x = \xi(k)$ and $\zeta = \nu|_{[k+m-l,k+m]}$.
– Now pick $x' = \xi(k+1)$ and observe that $\widehat{x}' \in \mathrm{E}^{\mathcal{I}^l_m}(x')$, hence $(\widehat{x}', x') \in \mathcal{R}^{-1}$. Furthermore, (2.20b) implies the existence of w' s.t. $(x, w', x') \in \delta$ and $\pi_W(w') = v$, what proves the statement.

"$\Rightarrow$" :
– Pick $x \in X$ and $\zeta \in D_{l+1}$ s.t. $\zeta|_{[0,l-1]} \in \mathrm{E}^{\mathcal{I}^l_m}(x)$. Furthermore define $\widehat{x} = \zeta|_{[0,l-1]}$ and $\widehat{x}' = \zeta|_{[1,l]}$ and $v = \zeta(l-m)$ and observe from (A.1b) that there exists w s.t. $(\widehat{x}, w, \widehat{x}') \in \widehat{\delta}^{\mathcal{I}^l_m}$ and $\pi_V(w) = v$ and observe that $(\widehat{x}, x) \in \mathcal{R}^{-1}$.
– As $\mathcal{R}^{-1} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}, \mathcal{Q})$ we know that there exist x' and w' s.t. $(x, w', x') \in \delta$, $\pi_W(w') = v$ and $\widehat{x}' \in \mathrm{E}^{\mathcal{I}^l_m}(x')$.

– With this we know that (A.2) holds, hence $\zeta = \tilde{\nu}|_{[k+m-l,k+m]}$ and $x = \tilde{\xi}(k)$ and therefore $\zeta \in \mathrm{E}^{[m-l,m]}(x)$.

A.1.5. Proof of Theorem 6.12

To simplify the proof of Theorem 6.12 (ii) we first show that the necessary and sufficient condition in Theorem 6.12 (ii) is equivalent to a particular condition of the domino-game.

Lemma A.2. *Given* (6.1) *let*

$$\forall \zeta \in (\{\diamond\} \cup V)^{l+2} \; . \left(\begin{pmatrix} \zeta|_{[0,l]} \in \{\diamond\}^l \cup D_{l+1} \\ \wedge \zeta|_{[1,l+1]} \in D_{l+1} \end{pmatrix} \Rightarrow \zeta \in D_{l+2} \right). \tag{A.9}$$

Then

$$(\text{A.9}) \Leftrightarrow \widehat{\mathcal{B}}_V^{l\uparrow} = \widehat{\mathcal{B}}_V^{l+1\uparrow}.$$

Proof. We prove both directions separately.

"$\Rightarrow$":
– Recall from Lemma 2.31 (i) that $\widehat{\mathcal{B}}_V^{l+1\uparrow} \subseteq \widehat{\mathcal{B}}_V^{l\uparrow}$ always holds. We therefore only prove that (A.9) implies $\widehat{\mathcal{B}}_V^{l\uparrow} \subseteq \widehat{\mathcal{B}}_V^{l+1\uparrow}$.
– Now pick $\nu \in {}^{\diamond}\widehat{\mathcal{B}}_V^{l\uparrow}$ and observe that the second line in (6.2) and (A.9) implies $\forall k{\in}\mathbb{N}_0 \; . \; \nu|_{[k-l-1,k]}{\in}D_{l+2}$. Using (6.2) again and observing that the first line in (6.2) is independent of l gives $\nu \in {}^{\diamond}\widehat{\mathcal{B}}_V^{l+1\uparrow}$, hence $\widehat{\mathcal{B}}_V^{l\uparrow} \subseteq \widehat{\mathcal{B}}_V^{l+1\uparrow}$.

"$\Leftarrow$":
– Observe that (A.9) always holds if $\zeta|_{[0,l]} \in \{\diamond\}^l$. We therefore pick $\zeta \in V^{l+2}$ s.t. $\zeta|_{[0,l]} \in D_{l+1}$ and $\zeta|_{[1,l+1]} \in D_{l+1}$.
– Now recall from the proof of Proposition 6.1 that $\pi_V(\overline{\mathcal{B}}) = \mathcal{B}_V$ and from Lemma 2.31 (ii) and Proposition 2.32 that $\bigcup_{k\in\mathbb{N}_0} \overline{\mathcal{B}}|_{[k,k+l]} = \bigcup_{k\in\mathbb{N}_0} \widehat{\mathcal{B}}_V^{l\uparrow}|_{[k,k+l]}$. This immediately yields $D_{l+1} = \bigcup_{k\in\mathbb{N}_0} {}^{\diamond}\widehat{\mathcal{B}}_V^{l\uparrow}|_{[k-l,k]}$.
– This implies the existence of $\nu, \nu' \in {}^{\diamond}\widehat{\mathcal{B}}_V^{l\uparrow}$ and $k, k' \in \mathbb{N}_0$ s.t. $\nu|_{[k-l,k]} = \zeta|_{[0,l]}$ and $\nu'|_{[k'-l,k']} = \zeta|_{[1,l+1]}$.
– Picking $\nu'' = \nu \wedge_{k'}^{k} \nu$ it is easily verified that $\nu'' \in {}^{\diamond}\widehat{\mathcal{B}}_V^{l\uparrow}$ and $\nu''|_{[k-l,k+1]} = \zeta$.
– As $\widehat{\mathcal{B}}_V^{l\uparrow} = \widehat{\mathcal{B}}_V^{l+1\uparrow}$ we know that $\nu'' \in \widehat{\mathcal{B}}_V^{l+1\uparrow}$ and therefore using (6.2) gives $\nu''|_{[k-l,k+1]}{\in}$ D_{l+2}, what proves the statement . ∎

Proof of Theorem 6.12

We show both statements separately.

(i) Show $\mathcal{R} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}_m^{l+1}}, \widehat{\mathcal{Q}}^{\mathcal{I}_m^l})$.

• Show (5.6a):
– Let $\widehat{x}_{l+1} \in \widehat{X}_0^{\mathcal{I}_m^{l+1}}$ and pick $\widehat{x}_l = \widehat{x}_{l+1}|_{[1,l]}$.
– It follows from (6.7b) that there exists an $x \in X_0$ s.t. $\widehat{x}_{l+1} \in \mathrm{E}^{\mathcal{I}_m^{l+1}}(x)$.
– Now it can be easily observed from (6.5) that $\widehat{x}_l \in \mathrm{E}^{\mathcal{I}_m^l}(x)$, hence $\widehat{x}_l \in \widehat{X}_0^{\mathcal{I}_m^l}$ (from (6.7b)) and $(\widehat{x}_{l+1}, \widehat{x}_l) \in \mathcal{R}$ (from (6.13)).

• Show (5.6b):
– Pick $(\widehat{x}_{l+1}, \widehat{x}_l) \in \mathcal{R}$, w, v and $\widehat{x}'_{l+1}$, $\widehat{x}'_l$ s.t. $(\widehat{x}_{l+1}, w, \widehat{x}'_{l+1}) \in \widehat{\delta}^{\mathcal{I}_m^{l+1}}$, $\widehat{x}'_l = \widehat{x}'_{l+1}|_{[1,l]}$ and $v = \pi_V(w)$.
– With this choice it immediately follows that $(\widehat{x}'_{l+1}, \widehat{x}'_l) \in \mathcal{R}$ and it remains to show that there exist w' s.t. $(\widehat{x}_l, w', \widehat{x}'_l) \in \widehat{\delta}^{\mathcal{I}_m^l}$, and $v = \pi_V(w')$.
– Using Lemma A.1 we have

$$\begin{aligned}
&\widehat{x}'_{l+1}|_{[0,l-m]} = \left(\widehat{x}_{l+1}|_{[0,l-m]} \cdot v\right)|_{[1,l+1-m]}\\
&\widehat{x}'_{l+1}|_{[l+1-m,l-1]} = \widehat{x}_{l+1}|_{[l-m+2,l]}\\
&\widehat{x}_{l+1}|_{[0,l-m]} \cdot v \cdot \widehat{x}'_{l+1}|_{[l+1-m,l]} \in D_{l+2}
\end{aligned}$$

– Now using $\widehat{x}_l = \widehat{x}_{l+1}|_{[1,l]}$ (from (6.13)) and $\widehat{x}'_l = \widehat{x}'_{l+1}|_{[1,l]}$ (from above) yields

$$\begin{aligned}
&\widehat{x}'_l|_{[0,l-m-1]} = \left(\widehat{x}_l|_{[0,l-m-1]} \cdot v\right)|_{[1,l-m]}\\
&\widehat{x}'_l|_{[l-m,l-2]} = \widehat{x}_l|_{[l-m+1,l-1]}\\
&\widehat{x}_l|_{[0,l-m-1]} \cdot v \cdot \widehat{x}'_l|_{[l-m,l-1]} \in D_{l+1}.
\end{aligned}$$

By using Lemma A.1 again this proves the statement .

(ii) Show $\mathcal{R}^{-1} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}_m^l}, \widehat{\mathcal{Q}}^{\mathcal{I}_m^{l+1}}) \Leftrightarrow \widehat{\mathcal{B}}_V^{l\uparrow} = \widehat{\mathcal{B}}_V^{l+1\uparrow}$

• We first show that (5.6a) always holds for $\mathcal{R}^{-1}$:
– Let $\widehat{x}_l \in \widehat{X}_0^{\mathcal{I}_m^l}$.
– It follows from (6.7b) that there exists an $x \in X_0$ s.t. $\widehat{x}_l \in \mathrm{E}^{\mathcal{I}_m^l}(x)$.
– Using (6.5) there exists $(\nu, \xi) \in \mathcal{B}_{SV}$ s.t. $\xi(0) = x$ and $\widehat{x}_l = \nu|_{[m-l,m-1]}$.
– Now pick $\widehat{x}_{l+1} = \nu|_{[m-l-1,m-1]}$ and observe that $\widehat{x}_{l+1} \in \mathrm{E}^{\mathcal{I}_m^{l+1}}(x)$, hence $\widehat{x}_{m+1} \in \widehat{X}_0^{\mathcal{I}_m^{l+1}}$ (from (6.7b)) and $(\widehat{x}_l, \widehat{x}_{l+1}) \in \mathcal{R}^{-1}$ (from (6.13)).

• It remains to show that (5.6a) holds for $\mathcal{R}^{-1}$ iff (A.9) holds. We show both directions separately:

"$\Leftarrow$:"
- Pick $(\widehat{x}_l, \widehat{x}_{l+1}) \in \mathcal{R}^{-1}$, w, v and $\widehat{x}'_{l+1}$, $\widehat{x}'_l$ s.t. $(\widehat{x}_l, w, \widehat{x}'_l) \in \widehat{\delta}^{\mathcal{I}^l_m}$, $\widehat{x}'_{l+1} = \widehat{x}_l(0) \cdot \widehat{x}'_l$ (i.e., $(\widehat{x}'_{l+1}, \widehat{x}'_l) \in \mathcal{R}$) and $v = \pi_V(w)$.
- Now using (A.1b) implies the existence of $\zeta \in D_{l+1}$ s.t. $\zeta|_{[0,l-1]} = \widehat{x}_l$, $\zeta|_{[1,l]} = \widehat{x}'_l$ and $v = \zeta(l-m)$, hence $\widehat{x}'_{l+1} = \zeta \in D_{l+1}$. Furthermore, recall that $\widehat{x}_{l+1} \in \widehat{X}^{\mathcal{I}^{l+1}_m} \subseteq \{\diamond\}^l \cup D_{l+1}$ (from (A.3)). Therefore, we can apply Lemma A.2 and obtain $\zeta' \in D_{l+2}$ s.t. $\zeta'|_{[0,l]} = \widehat{x}_{l+1}$, $\zeta'|_{[1,l+1]} = \widehat{x}'_{l+1}$ and $v = \zeta'(l+1-m)$.
- Using (A.1b) again this implies the existence of w' s.t. $v = \pi_V(w')$ and $(\widehat{x}_{l+1}, w', \widehat{x}'_{l+1}) \in \widehat{\delta}^{\mathcal{I}^{l+1}_m}$, what proves the statement.

"$\Rightarrow$:"
- Pick ζ s.t. $\zeta|_{[0,l]} \in \{\diamond\}^l \cup D_{l+1}$ and $\zeta|_{[1,l+1]} \in D_{l+1}$. Furthermore, define $\widehat{x}_l = \zeta|_{[1,l]}$ and $\widehat{x}'_l = \zeta|_{[2,l]}$ and $v = \zeta(l-m+1)$.
- With this choice (A.1b) implies the existence of w s.t. $v = \pi_V(w)$ and $(\widehat{x}_l, w, \widehat{x}'_l) \in \widehat{\delta}^{\mathcal{I}^l_m}$.
- Now observe that $(\widehat{x}_l, \zeta|_{[0,l]}) \in \mathcal{R}^{-1}$ and recall that $\mathcal{R}^{-1} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}, \widehat{\mathcal{Q}}^{\mathcal{I}^{l+1}_m})$, hence we know that there exists ζ', w' s.t. $(\zeta|_{[0,l]}, w', \zeta') \in \widehat{\delta}^{\mathcal{I}^{l+1}_m}$, $\pi_V(w') = v = \zeta(l-m+1)$ and $\widehat{x}'_l = \zeta'|_{[1,l]}$, hence $\zeta' = \zeta|_{[1,l+1]}$.
- Now using (A.1b) for $l+1$ implies $\zeta \in D_{l+2}$, what proves the statement.

A.1.6. Proof of Theorem 6.13

To simplify the proof of Theorem 6.13 (ii) we first show that the necessary and sufficient conditions in Theorem 6.13 (ii) are equivalent to a simpler condition of the domino-game.

Lemma A.3. *Given* (6.1) *let*

$$\forall \zeta, \zeta' \in D_{l+1} \,.\, \left(\zeta|_{[0,l-1]} = \zeta'|_{[0,l-1]} \Rightarrow \zeta = \zeta'\right). \tag{A.10}$$

Then

$$\text{(A.10)} \Leftrightarrow \begin{pmatrix} \mathcal{Q} \textit{ is future unique w.r.t. } \mathcal{I}^l_{m+1} \\ \wedge \mathcal{Q} \textit{ is state-based async. } l\textit{-complete w.r.t. } \mathcal{I}^l_m \end{pmatrix}.$$

Proof. We show all statements separately:

- Show (A.10) implies future uniqueness of $\mathcal{Q}$ w.r.t. $\mathcal{I}^l_{m+1}$:
– Pick $x \in X$ and $\zeta, \zeta' \in \mathrm{E}^{\mathcal{I}^l_{m+1}}(x)$.
– It follows from (6.5) that there exist $(\nu, \xi), (\nu', \xi') \in \mathcal{B}_{SV}$ and $k, k' \in \mathbb{N}_0$ s.t.

$$\begin{aligned} &\xi(k) = \xi(k') = x, \\ &\nu|_{[k-l+m+1,k+m]} = \zeta \text{ and} \\ &\nu'|_{[k'-l+m+1,k'+m]} = \zeta' \end{aligned}$$

– Using (2.25) we can therefore construct $(\nu'', \xi'') = (\nu, \xi) \wedge_{k'}^{k} (\nu', \xi')$.
– Now pick $\tilde{\zeta} = \nu|_{[k-l,k]}$ and $\tilde{\zeta}' = \nu''|_{[k-l,k]}$ and observe

$$\tilde{\zeta}'|_{[0,l-1]} = \nu''|_{[k-l,k-1]} = \nu|_{[k-l,k-1]} = \tilde{\zeta}|_{[0,l-1]}. \tag{A.11}$$

– Using (A.10) we therefore have $\tilde{\zeta} = \tilde{\zeta}'$, hence $\nu(k) = \zeta(l-m+1) = \zeta'(l-m+1) = \nu''(k)$.
– Now we can pick $\tilde{\zeta} = \nu|_{[k-l+1,k+1]}$ and $\tilde{\zeta}' = \nu''|_{[k-l+1,k+1]}$ and (by reusing the above argument) obtain $\nu(k) = \zeta(l-m+2) = \zeta'(l-m+2) = \nu''(k)$.
– Iteratively applying the above reasoning therefore yields $\zeta|_{[l-m+1,l]} = \zeta'|_{[l-m+1,l]}$, what proves the statement.

• Show (A.10) implies that $\mathcal{Q}$ is state-based asynchronously l-complete w.r.t. $\mathcal{I}_m^l$:
– Pick $x \in X$, $\zeta \in D_{l+1}$ s.t. $\zeta|_{[0,l-1]} \in \mathrm{E}^{\mathcal{I}_m^l}(x)$.
– It follows from (6.5) that there exist $(\nu, \xi) \in \mathcal{B}_{SV}$ and $k \in \mathbb{N}_0$ s.t.

$$\begin{aligned} &\xi(k) = \xi(k') = x \text{ and} \\ &\nu|_{[k-l+m,k+m-1]} = \zeta|_{[0,l-1]} \end{aligned}$$

– Now pick $\zeta' = \nu|_{[k-l+m,k+m]}$ and observe that $\zeta' \in \mathrm{E}^{[m-l,m]}(x)$, $\zeta|_{[0,l-1]} = \zeta'|_{[0,l-1]}$ and $\zeta' \in D_{l+1}$.
– Using (A.10) we have $\zeta = \zeta'$ and therefore $\zeta \in \mathrm{E}^{[m-l,m]}(x)$, what proves the statement.

• Show that future uniqueness of $\mathcal{Q}$ w.r.t. $\mathcal{I}_{m+1}^l$ and state-based asynchronously l-completeness of $\mathcal{Q}$ w.r.t. $\mathcal{I}_m^l$ implies (A.10):
– Pick $\zeta, \zeta' \in D_{l+1}$ s.t. $\zeta|_{[0,l-1]} = \zeta'|_{[0,l-1]}$.
– Observe that this implies $\zeta|_{[0,l-1]} \in D_l$. Hence, there exists $x \in X$ s.t. $\zeta|_{[0,l-1]} \in \mathrm{E}^{\mathcal{I}_m^l}(x) = \mathrm{E}^{[m-l,m-1]}(x)$.
– Using that $\mathcal{Q}$ is state-based asynchronously l-complete w.r.t. $\mathcal{I}_m^l$ we know that $\zeta, \zeta' \in \mathrm{E}^{[m-l,m]}(x)$.
– Using (6.5) this implies that $\zeta|_{[1,l]}, \zeta'|_{[1,l]} \in \mathrm{E}^{\mathcal{I}_{m+1}^l}(x)$.
– As $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_{m+1}^l$ this implies $\zeta|_{[1,l]} = \zeta'|_{[1,l]}$, hence $\zeta = \zeta'$, what proves the statement. $\square$

Proof of Theorem 6.13

(i) Show $\mathcal{R} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}_{m+1}^l}, \widehat{\mathcal{Q}}^{\mathcal{I}_m^l})$.

• Show that (5.6a) holds for $\mathcal{R}$:
– Let $\widehat{x}_{m+1} \in \widehat{X}_0^{\mathcal{I}_{m+1}^l}$ and pick $\widehat{x}_m = \diamond \cdot \widehat{x}_{m+1}|_{[0,l-2]}$, implying $\widehat{x}_{m+1}|_{[0,l-2]} = \widehat{x}_m|_{[1,l-1]}$.
– Furthermore, it follows from (6.7b) that there exists an $x \in X_0$ s.t. $\widehat{x}_{m+1} \in \mathrm{E}^{\mathcal{I}_{m+1}^l}(x)$.

– Now it can be easily observed from (6.5) that $\widehat{x}_m \in \mathrm{E}^{\mathcal{I}^l_m}(x)$, hence $\widehat{x}_m \in \widehat{X}_0^{\mathcal{I}^l_m}$ (from (6.7b)) and $(\widehat{x}_{m+1}, \widehat{x}_m) \in \mathcal{R}$ (from (6.13)).

• Show that (5.6b) holds for $\mathcal{R}$:
– Pick $(\widehat{x}_{m+1}, \widehat{x}_m) \in \mathcal{R}$, w, v and $\widehat{x}'_{m+1}$ s.t. $(\widehat{x}_{m+1}, w, \widehat{x}'_{m+1}) \in \widehat{\delta}^{\mathcal{I}^l_{m+1}}$ and $v = \pi_V(w)$.
– Using (6.7c) this implies $\widehat{x}_{m+1}|_{[1,l-1]} = \widehat{x}'_{m+1}|_{[0,l-2]}$ and $\widehat{x}_{m+1}(l-m-1) = v$.
– Now pick $\widehat{x}'_m = \widehat{x}_{m+1}$ and observe that the first two lines in (A.1a) hold and $(\widehat{x}_m, \widehat{x}'_m) \in \mathcal{R}$. Therefore (6.14) and (6.5) implies

$$\exists(\nu,\xi)\in\mathcal{B}_{SV}, k \;.\; \begin{pmatrix} \xi(k) = x \\ \wedge \widehat{x}_m = \nu|_{[k+m-l,k+m-1]} \end{pmatrix} \quad \text{and} \tag{A.12a}$$

$$\exists(\nu',\xi')\in\mathcal{B}_{SV}, k' \;.\; \begin{pmatrix} \xi'(k') = x \\ \wedge \widehat{x}'_m = \nu'|_{[k'+m+1-l,k'+m]} \end{pmatrix}. \tag{A.12b}$$

– As $\xi(k) = \xi'(k') = x$ in (A.12), using (2.25) gives

$$\begin{aligned} (\tilde{\nu}, \tilde{\xi}) &= (\nu, \xi) \wedge^k_{k'} (\nu', \xi') \in \mathcal{B}_{SV}, \quad \text{where} \\ \tilde{\nu}|_{[k+m-l,k+m]} &= \nu|_{[k+m-l,k-1]} \cdot \nu'|_{[k',k'+m]} \\ &= \widehat{x}_m|_{[0,l-m-1]} \cdot \widehat{x}'_m|_{[l-m-1,l-1]}. \end{aligned} \tag{A.13}$$

– As $\widehat{x}'_m(l-m-1) = \widehat{x}_{m+1}(l-m-1) = v$ this implies

$$\widehat{x}_m|_{[0,l-m-1]} \cdot v \cdot \widehat{x}'_m|_{[l-m,l-1]} \in D_{l+1}.$$

– Now (A.1a) implies the existence of w' s.t. $\pi_V(w') = v$ and $(\widehat{x}_m, w', \widehat{x}'_m) \in \widehat{\delta}^{\mathcal{I}^l_m}$.
– It remains to show that $(\widehat{x}'_{m+1}, \widehat{x}'_m) \in \mathcal{R}$.
· First observe that $\widehat{x}'_{m+1}|_{[0,l-2]} = \widehat{x}_{m+1}|_{[1,l-1]} = \widehat{x}'_m|_{[1,l-1]}$ from above, hence the first line in (6.14) holds.
· Now $(\widehat{x}_{m+1}, w, \widehat{x}'_{m+1}) \in \widehat{\delta}^{\mathcal{I}^l_{m+1}}$ and (A.1b) implies the existence of

$$\zeta \in D_{l+1} \; s.t. \; \widehat{x}_{m+1} = \zeta|_{[0,l-1]} \text{ and } \widehat{x}'_{m+1} = \zeta|_{[1,l]}.$$

· We can therefore pick $(\tilde{\nu}, \tilde{\xi}) \in \mathcal{B}_{SV}$, $\tilde{x}$ and $\tilde{k}$ s.t.

$$\tilde{\nu}|_{[\tilde{k}+(m+1)-l,\tilde{k}+(m+1)]} = \zeta \text{ and } \tilde{x} = \tilde{\xi}(k+1),$$

implying $\widehat{x}'_{m+1} \in \mathrm{E}^{\mathcal{I}^l_{m+1}}(\tilde{x})$ and $\widehat{x}'_m = \widehat{x}_{m+1} \in \mathrm{E}^{\mathcal{I}^l_m}(\tilde{x})$, hence the second line in (6.14) also holds.

(ii) Show $\mathcal{R}^{-1} \in \mathfrak{S}_V(\widehat{\mathcal{Q}}^{\mathcal{I}^l_m}, \widehat{\mathcal{Q}}^{\mathcal{I}^l_{m+1}}) \Leftrightarrow$ (A.10)

• We first show that (5.6a) always holds for $\mathcal{R}^{-1}$:
– Let $\widehat{x}_m \in \widehat{X}_0^{\mathcal{I}_m^l}$ and observe that it follows from (6.7b) that there exists an $x \in X_0$ s.t. $\widehat{x}_m \in \mathrm{E}^{\mathcal{I}_m^l}(x)$.
– Using (6.5) there exists $(\nu, \xi) \in \mathcal{B}_{SV}$ s.t. $\xi(0) = x$ and $\widehat{x}_m = \nu|_{[m-l,m-1]}$.
– Now pick $\widehat{x}_{m+1} = \nu|_{[m+1-l,m]}$ and observe that $\widehat{x}_{m+1} \in \mathrm{E}^{\mathcal{I}_{m+1}^l}(x)$, hence $\widehat{x}_{m+1} \in \widehat{X}_0^{\mathcal{I}_{m+1}^l}$ (from (6.7b)) and $(\widehat{x}_m, \widehat{x}_{m+1}) \in \mathcal{R}^{-1}$ (from (6.14)).

• It remains to show that (5.6b) holds for $\mathcal{R}^{-1}$ iff (A.10) in Lemma A.3 holds. We show both statements separately:

"$\Leftarrow$":
– Pick $(\widehat{x}_m, \widehat{x}_{m+1}) \in \mathcal{R}^{-1}$, w, v and $\widehat{x}'_m$ s.t. $(\widehat{x}_m, w, \widehat{x}'_m) \in \widehat{\delta}^{\mathcal{I}_m^l}$ and $v = \pi_V(w)$.
– Using (A.1b) this implies the existence of $\zeta \in D_{l+1}$ s.t. $\widehat{x}_m = \zeta|_{[0,l-1]}$, $\widehat{x}'_m = \zeta|_{[1,l]}$ and $v = \zeta(l-m)$.
– Now let $\zeta' = \widehat{x}_m|_{[0,0]} \cdot \widehat{x}_{m+1}$ and observe that $\zeta|_{[0,l-1]} = \zeta'|_{[0,l-1]}$ and therefore $\zeta = \zeta'$ (from (A.10)), hence $\widehat{x}_{m+1} = \widehat{x}'_m$.
– As $\zeta' \in D_{l+1}$ we can pick $(\tilde{\nu}, \tilde{\xi}) \in \mathcal{B}_{SV}$, $\tilde{x}$ and $\tilde{k}$ s.t. $\tilde{\nu}|_{[\tilde{k}+m-l,\tilde{k}+m]} = \zeta$.
– Now pick $\zeta'' = \tilde{\nu}|_{[\tilde{k}+(m+1)-l,\tilde{k}+(m+1)]}$ and observe that $\widehat{x}_{m+1} = \zeta''|_{[0,l-1]}$ and $\zeta''(l-m-1) = v$. Therefore choosing $\widehat{x}'_{m+1} = \zeta''|_{[1,l]}$ and using (A.1b) implies the existence of w' s.t. $(\widehat{x}_{m+1}, w', \widehat{x}'_{m+1}) \in \widehat{\delta}^{\mathcal{I}_{m+1}^l}$ and $\pi_V(w') = v$.
– Furthermore, observe that choosing $\tilde{x} = \tilde{\xi}(\tilde{k}+1)$ implies $\widehat{x}'_{m+1} \in \mathrm{E}^{\mathcal{I}_{m+1}^l}(\tilde{x})$ and $\widehat{x}'_m = \widehat{x}_{m+1} \in \mathrm{E}^{\mathcal{I}_m^l}(\tilde{x})$, hence $(\widehat{x}'_m, \widehat{x}'_{m+1}) \in \mathcal{R}^{-1}$.

"$\Rightarrow$":
– Pick $\zeta, \zeta' \in D_{l+1}$ s.t. $\zeta|_{[0,l-1]} = \zeta'|_{[0,l-1]}$ and pick $\widehat{x}_m = \zeta|_{[0,l-1]} = \zeta'|_{[0,l-1]}$, $\widehat{x}'_m = \zeta|_{[1,l]}$, $\widehat{x}_{m+1} = \zeta'|_{[1,l]}$ and $v = \zeta(l-m)$.
– Using (A.1b) this implies the existence of w s.t. $(\widehat{x}_m, w, \widehat{x}'_m) \in \widehat{\delta}^{\mathcal{I}_m^l}$ and $v = \pi_V(w)$.
– Using the same reasoning as before it furthermore holds that $(\widehat{x}_m, \widehat{x}_{m+1}) \in \mathcal{R}^{-1}$ as $\zeta' \in D_{l+1}$.
– As $\mathcal{R}^{-1} \in \mathfrak{S}_V(\widehat{Q}^{\mathcal{I}_m^l}, \widehat{Q}^{\mathcal{I}_{m+1}^l})$ we know that there exist $w', \widehat{x}'_{m+1}$ s.t. $(\widehat{x}_{m+1}, w', \widehat{x}'_{m+1}) \in \widehat{\delta}^{\mathcal{I}_m^l}$, $v = \pi_V(w')$ and $(\widehat{x}'_m, \widehat{x}'_{m+1}) \in \mathcal{R}^{-1}$.
– Now let $l = 1$ implying $m = 0$ (as $m < l$). Then (6.7c) implies $v = \widehat{x}_{m+1}(1)$ and therefore $\zeta(1) = v = \widehat{x}_{m+1}(1) = \zeta'(1)$ implying $\zeta = \zeta'$.
– Now let $l > 1$ and observe that $\widehat{x}_{m+1}|_{[1,l-1]} = \widehat{x}'_{m+1}|_{[0,l-2]} = \widehat{x}'_m|_{[1,l-1]}$ (from (6.7c) and (6.14)). Therefore $\zeta(l) = \widehat{x}'_m(l-1) = \widehat{x}_{m+1}(l-1) = \zeta(l)'$ holds, giving $\zeta = \zeta'$.

A.2. Proofs for Chapter 7

In this section we provide detailed proofs for the claims made in Chapter 7.

A.2.1. Proof of Theorem 7.6

Recall from Definition 7.3 that $\phi^l : X \to \widehat{Y}^l$ and therefore applying (3.2b) to δ^l in (7.7) yields $\mathrm{H}_{\delta^l}(x) = \{\phi^l(x)\}$ (as $\mathrm{H}_{\delta^l} : X \to 2^{\widehat{Y}^l}$). Hence we have

$$\forall x \in X \;.\; \left(\widehat{y} \in \mathrm{H}_{\delta^l}(x) \Leftrightarrow \widehat{y} = \phi^l(x)\right). \tag{A.14}$$

Furthermore, recall from Definition 7.3 that $\widehat{\mathcal{Q}}^{l\triangledown}$ is obtained from $\mathcal{Q}^l$ via Theorem 3.8. Hence, applying the latter to $\mathcal{Q}^l$ instead of $\mathcal{Q}$ means that we have to substitute Y by $\widehat{Y}^l$, δ by δ^l and H_δ by H_{δ^l} (using (A.14)) in Equation 3.5, giving

$$\widehat{X}^{l\triangledown} = \left\{\widehat{y} \in \widehat{Y}^l \,\middle|\, \exists x \in X \;.\; \widehat{y} = \phi^l(x)\right\} = \widehat{Y}^l = \widehat{X}^{l\blacktriangledown}, \tag{A.15a}$$

$$\widehat{X}_0^{l\triangledown} = \left\{\widehat{y} \in \widehat{Y}^l \,\middle|\, \exists x \in X_0 \;.\; \widehat{y} = \phi^l(x)\right\} = \widehat{X}_0^{l\blacktriangledown}, \quad \text{and}$$

$$\widehat{\delta}^{l\triangledown} = \left\{ (\widehat{x}, u, \widehat{y}, \widehat{x}') \,\middle|\, \exists x, x' \in X \;.\; \begin{pmatrix} \widehat{x} = \phi^l(x) \\ \wedge \widehat{x}' = \phi^l(x') \\ \wedge (x, u, \widehat{y}, x') \in \delta^l \end{pmatrix} \right\}. \tag{A.15b}$$

The equality in (A.15a) holds as $\widehat{Y}^l$ is defined to be the set of equivalence classes of Φ^l. Now substituting (7.7) in (A.15b) and using $\mathrm{H}_{\widehat{\delta}^\triangledown}(\widehat{x}) = \{\widehat{x}\}$ (from (3.5)) yields

$$\widehat{\delta}^{l\triangledown} = \left\{ (\widehat{x}, u, \widehat{x}, \widehat{x}') \,\middle|\, \exists x, x' \in X, y \in Y \;.\; \begin{pmatrix} \widehat{x} = \phi^l(x) \\ \wedge \widehat{x}' = \phi^l(x') \\ \wedge (x, u, y, x') \in \delta \end{pmatrix} \right\}$$

With (7.9c), this imediatelly gives

$$\widehat{\delta}^{l\triangledown} = \left\{ (\widehat{x}, u, \widehat{x}, \widehat{x}') \,\middle|\, \exists y \in Y \;.\; (\widehat{x}, u, y, \widehat{x}') \in \widehat{\delta}^{l\blacktriangledown} \right\},$$

what proves the statement. □

A.2.2. Proof of Theorem 7.7

(i) Show $\mathcal{R} \in \mathfrak{S}_{U\times Y}(\mathcal{Q}, \widehat{\mathcal{Q}}^{l\blacktriangledown})$:
- Pick $x \in X_0 \subseteq X$. As Φ^l is a partition of X (see [11, Prop.3.9(i)]), ϕ^l is the natural projection map of Φ^l and $\widehat{Y}^l$ is a set of equivalence classes of Φ^l, there

must exist $\widehat{x} \in \widehat{Y}^l$ s.t. $\widehat{x} = \phi^l(x)$. Hence (5.6a) holds
- Pick $(x, \widehat{x}) \in \mathcal{R}$, i.e. $\widehat{x} = \phi^l(x)$ and $(x, u, y, x') \in \delta$. If we now choose $\widehat{x}' = \phi^l(x')$ we obtain $(x', \widehat{x}') \in \mathcal{R}$ and $(\widehat{x}, u, y, \widehat{x}') \in \widehat{\delta}^{l\blacktriangledown}$ (from (7.9c)). Hence (5.6b) holds.

(ii) Show, $\mathcal{R}^{-1} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{l\blacktriangledown}, \mathcal{Q}) \Rightarrow \Phi^l$ is a fixed-point of (7.4):
- Observe that (5.6b) holds for $\mathcal{R}^{-1}$ and $V = Y$ iff

$$\forall\, \widehat{x}, \widehat{x}', u, y, x \,.\, \left(\begin{pmatrix} \widehat{x} = \phi^l(x) \\ \wedge(\widehat{x}, u, y, \widehat{x}') \in \widehat{\delta}^{l\blacktriangledown} \end{pmatrix} \Rightarrow \exists x', u' \,.\, \begin{pmatrix} \widehat{x}' = \phi^l(x') \\ \wedge(x, u', y, x') \in \delta \end{pmatrix} \right) \tag{A.16}$$

- Fix $\widehat{x}, \widehat{x}', u, y$ and recall from (7.3) and (7.9c) that

$$(\widehat{x}, u, y, \widehat{x}') \in \widehat{\delta}^{l\blacktriangledown} \Leftrightarrow \exists \tilde{x}, \tilde{x}' \in X \,.\, \begin{pmatrix} \widehat{x} = \phi^l(\tilde{x}) \\ \wedge \widehat{x}' = \phi^l(\tilde{x}') \\ \wedge(\tilde{x}, u, y, \tilde{x}') \in \delta \end{pmatrix}$$

- Now let $Z := \left(\phi^l\right)^{-1}(\widehat{x})$ and $Z' := \left(\phi^l\right)^{-1}(\widehat{x}')$ and observe that (A.16) implies that for *all* choices of $x \in Z$ there exists u' and $x' \in Z'$ s.t. $(x, u', y, x') \in \delta$. Hence, using (7.3), (A.16) implies that

$$Z \cap \mathrm{T}_\delta^{-1}(Z') \neq \emptyset \Rightarrow Z \subseteq \mathrm{T}_\delta^{-1}(Z'),$$

i.e. Φ^l is a fixed point of (7.4).

(ii) Show, $\begin{pmatrix} \text{(3.2d) holds for } \mathcal{Q} \\ \wedge \Phi^l \text{ is a fixed-point of (7.4)} \end{pmatrix} \Rightarrow \mathcal{R}^{-1} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{l\blacktriangledown}, \mathcal{Q})$
- (5.6a) always holds for $\mathcal{R}^{-1}$, as we can pick $\widehat{x} \in \widehat{X}_0^{l\blacktriangledown}$ and obtain from (7.9b) that there exists $x \in X_0$ s.t. $\widehat{x} = \phi^l(x)$.
- Pick $\widehat{x}, \widehat{x}', u, y, x$ s.t. the left side of (A.16) holds and let $Z := \left(\phi^l\right)^{-1}(\widehat{x})$ and $Z' := \left(\phi^l\right)^{-1}(\widehat{x}')$.
- Then (7.9c) implies the existence of $\tilde{x} \in Z$ and $\tilde{x}' \in Z'$ s.t. $(\tilde{x}, u, y, \tilde{x}') \in \delta$, hence $Z \cap \mathrm{T}_\delta^{-1}(Z') \neq \emptyset$.
- As Φ^l is a fixed-point of (7.4), we therefore know that $Z \subseteq \mathrm{T}_\delta^{-1}(Z')$, hence there exists u', y' and $x' \in Z'$ s.t. $(x, u, y', x') \in \delta$ (with x from the left side of (A.16)).
- However, the right side of (A.16) only holds if we can choose $y' = y$, what follows from (3.2d) since $\mathcal{Q}$ has separable dynamics. $\square$

A.2.3. Proof of Theorem 7.8

(i) Show $\mathcal{R} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{l+1\blacktriangledown}, \widehat{\mathcal{Q}}^{l\blacktriangledown})$:

- Show (5.6a) holds for $\mathcal{R}$: Pick $\widehat{x}_{l+1} \in \widehat{X}_0^{l+1\blacktriangledown}$ and $Z_{l+1} = \left(\phi^{l+1}\right)^{-1}(\widehat{x}_{l+1})$ and observe from (7.9b) that there exists $x \in X_0$ s.t. $x \in Z_{l+1}$. Using (7.6) there exists $Z_l \in \Phi^l$ s.t. $Z_l \supseteq Z_{l+1}$, hence $x \in Z_l$. Now choosing $\widehat{x}_l$ as the equivalence class of Z_l (i.e. $Z_l = \left(\phi^l\right)^{-1}(\widehat{x}_l)$), yields $\widehat{x}_l = \phi^l(x)$, giving $\widehat{x}_l \in \widehat{X}_0^{l\blacktriangledown}$ and $(\widehat{x}_{l+1}, \widehat{x}_l) \in \mathcal{R}$.
- Show (5.6b) holds for $\mathcal{R}$:

– Pick $(\widehat{x}_{l+1}, \widehat{x}_l) \in \mathcal{R}$, $Z_{l+1} = \left(\phi^{l+1}\right)^{-1}(\widehat{x}_{l+1})$ and $Z_l = \left(\phi^l\right)^{-1}(\widehat{x}_l)$, , i.e., $Z_{l+1} \subseteq Z_l$. Furthermore, pick $u, y, \widehat{x}'_{l+1}, \widehat{x}'_l$ s.t. $(\widehat{x}_{l+1}, u, y, \widehat{x}'_{l+1}) \in \widehat{\delta}^{l+1\blacktriangledown}$ and $Z'_{l+1} = \left(\phi^{l+1}\right)^{-1}(\widehat{x}'_{l+1})$ and $Z'_l = \left(\phi^l\right)^{-1}(\widehat{x}'_l)$ s.t. $Z'_{l+1} \subseteq Z'_l$, implying that $(\widehat{x}'_{l+1}, \widehat{x}'_l) \in \mathcal{R}$.

– Recall from (7.3) and (7.9c) that

$$(\widehat{x}_{l+1}, u, y, \widehat{x}'_{l+1}) \in \widehat{\delta}^{l+1\blacktriangledown} \Leftrightarrow \exists x, x' \in X \,.\, \begin{pmatrix} \widehat{x}_{l+1} = \phi^{l+1}(x) \\ \wedge \widehat{x}'_{l+1} = \phi^{l+1}(x') \\ \wedge (x, u, y, x') \in \delta \end{pmatrix}$$

– As $Z_{l+1} \subseteq Z_l$ and $Z'_{l+1} \subseteq Z'_l$ we obviously have $\widehat{x}_l = \phi^l(x)$ and $\widehat{x}'_l = \phi^l(x')$, implying $(\widehat{x}_l, u, y, \widehat{x}'_l) \in \widehat{\delta}^{l\blacktriangledown}$, what proves the statement.

(ii) Show $\left(\mathcal{R}^{-1} \in \mathfrak{S}_{U\times Y}(\widehat{\mathcal{Q}}^{l\blacktriangledown}, \widehat{\mathcal{Q}}^{l+1\blacktriangledown}) \Leftrightarrow \Phi^l \text{ fixed-point of (7.4)}\right)$:

- We first prove that (5.6b) holds for $\mathcal{R}^{-1}$ iff Φ^l is a fixed-point of (7.4).

– Observe that (5.6b) holds for $\mathcal{R}^{-1}$ and $V = U \times Y$ iff

$$\forall\, \widehat{x}_{l+1}, \widehat{x}_l, \widehat{x}'_l, u, y \,.\, \begin{pmatrix} \left(\phi^{l+1}\right)^{-1}(\widehat{x}_{l+1}) \subseteq \left(\phi^l\right)^{-1}(\widehat{x}_l) \\ \wedge (\widehat{x}_l, u, y, \widehat{x}'_l) \in \widehat{\delta}^{l\blacktriangledown} \end{pmatrix} \Rightarrow \exists \widehat{x}'_{l+1} \,.\, \begin{pmatrix} \left(\phi^{l+1}\right)^{-1}(\widehat{x}'_{l+1}) \subseteq \left(\phi^l\right)^{-1}(\widehat{x}'_l) \\ \wedge (\widehat{x}_{l+1}, u, y, \widehat{x}'_{l+1}) \in \widehat{\delta}^{l+1\blacktriangledown} \end{pmatrix} \tag{A.17}$$

– "$\Leftarrow$": as Φ^l is a fixed-point of (7.4) we know that

$$\forall Z_l \in \Phi^l, Z_{l+1} \in \Phi^{l+1} \,.\, (Z_{l+1} \subseteq Z_l \Rightarrow Z_{l+1} = Z_l) \tag{A.18}$$

Hence, whenever the left side in (A.17) holds, we know that $\left(\phi^{l+1}\right)^{-1}(\widehat{x}_{l+1}) = \left(\phi^l\right)^{-1}(\widehat{x}_l)$ and we have to pick $\widehat{x}'_{l+1}$ s.t. $\left(\phi^{l+1}\right)^{-1}(\widehat{x}'_{l+1}) = \left(\phi^l\right)^{-1}(\widehat{x}'_l)$ and obtain that the right side of (A.17) trivially holds from (7.9c).

– "$\Rightarrow$": Fix $\widehat{x}_l, \widehat{x}'_l, u, y$ and define $Z_l = \left(\phi^l\right)^{-1}(\widehat{x}_l)$ and $Z'_l = \left(\phi^l\right)^{-1}(\widehat{x}'_l)$. With this, (A.17) implies that

$$\forall Z_{l+1} \subseteq Z_l \,.\, \left(Z_l \cap \mathrm{T}_\delta^{-1}(Z'_l) \neq \emptyset \Rightarrow Z_{l+1} \subseteq \mathrm{T}_\delta^{-1}(Z'_l)\right) \tag{A.19}$$

as $\mathrm{T}_\delta^{-1}(Z'_{l+1}) \subseteq \mathrm{T}_\delta^{-1}(Z'_l)$ with $Z'_{l+1} = \left(\phi^{l+1}\right)^{-1}(\widehat{x}'_{l+1})$.
– Now observe that from (7.6) and the fact that Φ^l and Φ^{l+1} are partitions follows $Z_l = \bigcup_{Z_{l+1} \subseteq Z_l} Z_{l+1}$. Hence, (A.19) implies

$$Z_l \cap \mathrm{T}_\delta^{-1}(Z'_l) \neq \emptyset \Rightarrow Z_l \subseteq \mathrm{T}_\delta^{-1}(Z'_l),$$

i.e., Φ^l is a fixed point of (7.4).
• Finally, we show that (5.6b) holds for $\mathcal{R}^{-1}$ if Φ^l is a fixed-point of (7.4):
– Pick $\widehat{x}_l \in \widehat{X}_0^{l\blacktriangledown}$ and observe from (7.9b) that there exists $x \in X_0$ s.t. $\widehat{x}_l = \phi^l(x)$.
– Now pick any $\widehat{x}_{l+1}$ s.t. $\left(\phi^{l+1}\right)^{-1}(\widehat{x}_{l+1}) \subseteq \left(\phi^l\right)^{-1}(\widehat{x}_l)$.
– As Φ^l is a fixed-point of (7.4) we know that (A.18) holds, hence $\left(\phi^l\right)^{-1}(\widehat{x}_l) = \left(\phi^{l+1}\right)^{-1}(\widehat{x}_{l+1})$ and therefore $\widehat{x}_{l+1} = \phi^{l+1}(x)$, implying $\widehat{x}_{l+1} \in \widehat{X}_0^{l+1\blacktriangledown}$. □

A.3. Proofs for Chapter 8

In this section we provide detailed proofs for the claims made in Chapter 8.

A.3.1. Proof of Proposition 8.1

We prove this statement by induction.
• $l = 1$: Recall that $\mathcal{I}_1^1 = [0, 0]$. Therefore (2.20b), (6.5) and $V = Y$ implies

$$\begin{aligned}
&\mathrm{E}^{\mathcal{I}_1^1}(x) \\
&= \left\{ y \in (Y \cup \{\diamond\}) \middle| \exists (\mu, \gamma, \xi) \in {}^\diamond\mathcal{B}_f(\mathcal{Q}), k \in \mathbb{N}_0 \;.\; \begin{pmatrix} \xi(k) = x \\ \wedge y = \gamma|_{[k,k]} \end{pmatrix} . \right\} \\
&= \left\{ y \in Y \middle| \exists (\mu, \gamma, \xi) \in {}^\diamond\mathcal{B}_f(\mathcal{Q}), k \in \mathbb{N}_0 \;.\; \begin{pmatrix} \xi(k) = x \\ \wedge \gamma(k) = y \\ \wedge (\xi(k), \mu(k), \gamma(k), \xi(k+1) \in \delta) \end{pmatrix} \right\} \\
&= \left\{ y \in Y \middle| \exists u \in U, x' \in X \;.\; (x, u, y, x') \in \delta \right\} \\
&= \mathrm{H}_\delta(x)
\end{aligned} \tag{A.20}$$

where the second last line holds as $\mathcal{Q}$ is reachable and live, as this implies that all transitions in δ are part of an infinite derivation. The rest follows from (7.4a).
• $(l-1) \to l$: Given that

$$\Phi^{l-1} = \left\{ \left(\mathrm{E}^{\mathcal{I}_{l-1}^{l-1}}\right)^{-1}(\Upsilon) \middle| \Upsilon \in 2^{(Y)^{l-1}} \right\} \tag{A.21}$$

holds with $\mathcal{I}_{l-1}^{l-1} = [0, l-2]$, we show (8.2).

– First observe that

$$\left\{ \left(\mathrm{E}^{\mathcal{I}_l^l}\right)^{-1}(\Upsilon) \middle| \Upsilon \in 2^{(Y)^l} \right\} = \left\{ \left\{ x \in X \middle| \Upsilon = \mathrm{E}^{\mathcal{I}_l^l}(x) \right\} \middle| \Upsilon \in 2^{(Y)^l} \right\}$$
$$= \left\{ Z \in 2^X \middle| \forall x, x' \in Z \; . \; \mathrm{E}^{\mathcal{I}_l^l}(x) = \mathrm{E}^{\mathcal{I}_l^l}(x') \right\} \tag{A.22}$$

We therefore have to show that the right side of (A.22) and the right side of (7.5b) in Lemma 7.2 coincide for a fixed $Z \in 2^X$.

– Using (6.5) and $V = Y$ we can rewrite $\mathrm{E}^{\mathcal{I}_l^l}$ for all $\zeta \in (Y)^l$ as follows.

$$\begin{aligned}
&\zeta \in \mathrm{E}^{[0,l-1]}(x)\\
&\Leftrightarrow \exists (\mu,\gamma,\xi) \in \mathcal{B}_f(\mathcal{Q}), k \in \mathbb{N}_0 \; . \; \begin{pmatrix} \xi(k) = x \\ \wedge \zeta = \gamma|_{[k,k+l-1]} \end{pmatrix}\\
&\Leftrightarrow \exists (\mu,\gamma,\xi) \in \mathcal{B}_f(\mathcal{Q}), k \in \mathbb{N}_0 \; . \; \begin{pmatrix} \xi(k) = x \\ \wedge (\xi(k), \mu(k), \zeta(0), \xi(k+1)) \in \delta \\ \wedge \zeta|_{[1,l-1]} = \gamma|_{[k+1,k+l-1]} \end{pmatrix}\\
&\Leftrightarrow \exists (\mu,\gamma,\xi) \in \mathcal{B}_f(\mathcal{Q}), k \in \mathbb{N}_0 \; . \; \begin{pmatrix} \xi(k) = x \\ \wedge (\xi(k), \mu(k), \zeta(0), \xi(k+1)) \in \delta \\ \wedge \zeta|_{[1,l-1]} \in \mathrm{E}^{[0,l-2]}(\xi(k+1)) \end{pmatrix}\\
&\Leftrightarrow \exists \tilde{x} \in X, u \in U \; . \; \begin{pmatrix} (x, u, \zeta(0), \tilde{x}) \in \delta \\ \wedge \zeta|_{[1,l-1]} \in \mathrm{E}^{[0,l-2]}(\tilde{x}) \end{pmatrix}
\end{aligned}$$

where the last line follows from $\mathcal{Q}$ being reachable and live.

– Using this equivalence we can rewrite the right side of (A.22) for fixed $x, x' \in Z$ as

$$\begin{aligned}
&\mathrm{E}^{[0,l-1]}(x) = \mathrm{E}^{[0,l-1]}(x')\\
&\Leftrightarrow \forall \zeta \in (Y)^l \; . \; \left(\zeta \in \mathrm{E}^{[0,l-1]}(x) \Rightarrow \zeta \in \mathrm{E}^{[0,l-1]}(x') \right)\\
&\Leftrightarrow \forall \zeta \in (Y)^l \; . \; \begin{array}{l} \exists \tilde{x} \in X, u \in U \; . \; \begin{pmatrix} (x, u, \zeta(0), \tilde{x}) \in \delta \\ \wedge \zeta|_{[1,l]} \in \mathrm{E}^{[0,l-2]}(\tilde{x}) \end{pmatrix} \Rightarrow \\ \quad \exists \tilde{x}' \in X, u' \in U \; . \; \begin{pmatrix} (x', u', \zeta(0), \tilde{x}') \in \delta \\ \wedge \zeta|_{[1,l]} \in \mathrm{E}^{[0,l-2]}(\tilde{x}') \end{pmatrix} \end{array}
\end{aligned}$$

$$\Leftrightarrow \forall y \in Y, \Upsilon \in 2^{(Y)^{l-1}} \;.\; \begin{array}{l} \exists \tilde{x} \in X, u \in U \;.\; \begin{pmatrix} (x,u,y,\tilde{x}) \in \delta \\ \wedge \Upsilon = \mathrm{E}^{[0,l-2]}(\tilde{x}) \end{pmatrix} \Rightarrow \\ \exists \tilde{x}' \in X, u' \in U \;.\; \begin{pmatrix} (x',u',y,\tilde{x}') \in \delta \\ \wedge \Upsilon = \mathrm{E}^{[0,l-2]}(\tilde{x}') \end{pmatrix} \end{array}$$

$$\Leftrightarrow \forall y \in Y, \Upsilon \in 2^{(Y)^{l-1}} \;.\; \begin{array}{l} \exists \tilde{x} \in \left(\mathrm{E}^{[0,l-2]}\right)\overset{-1}{(\Upsilon)}, u \in U \;.\; (x,u,y,\tilde{x}) \in \delta \Rightarrow \\ \exists \tilde{x}' \in \left(\mathrm{E}^{[0,l-2]}\right)\overset{-1}{(\Upsilon)}, u' \in U \;.\; (x',u',y,\tilde{x}') \in \delta \end{array}$$

– Now it follows from (8.1a) (in particular (3.2d)) that

$$\begin{aligned} &\mathrm{E}^{[0,l-1]}(x) = \mathrm{E}^{[0,l-1]}(x') \\ \Leftrightarrow &\begin{pmatrix} \mathrm{H}_\delta(x) = \mathrm{H}_\delta(x') \\ \wedge \forall Z^{l-1} \in \Phi^{l-1} \;.\; \left(\{x\} \cap \mathrm{T}_\delta^{-1}(Z^{l-1}) \neq \emptyset \Rightarrow \{x'\} \cap \mathrm{T}_\delta^{-1}(Z^{l-1}) \neq \emptyset\right) \end{pmatrix} \end{aligned}$$

– Substituting the previous statement into the right side of (A.22) yields

$$\begin{aligned} &\forall x, x' \in Z \;.\; \mathrm{E}^{[0,l-1]}(x) = \mathrm{E}^{[0,l-1]}(x') \\ \Leftrightarrow &\begin{pmatrix} \forall x, x' \in Z \;.\; \mathrm{H}_\delta(x) = \mathrm{H}_\delta(x') \\ \wedge \forall Z' \in \Phi^{l-1} \;.\; \left(\left(Z \cap \mathrm{T}_\delta^{-1}(Z') \neq \emptyset\right) \Rightarrow \left(Z \subseteq \mathrm{T}_\delta^{-1}(Z')\right)\right) \end{pmatrix} \\ \Leftrightarrow &\begin{pmatrix} \forall x, x' \in Z \;.\; \mathrm{H}_\delta(x) = \mathrm{H}_\delta(x') \\ \wedge Z \in \Phi^{l} \end{pmatrix} \end{aligned}$$

where the last line follows from applying (7.5b).
– Now it follows from (7.5a) in Lemma 7.2 that $Z \in \Phi^l$ implies $\forall x, x' \in Z \;.\; \mathrm{H}_\delta(x) = \mathrm{H}_\delta(x')$. We therefore obtain that

$$Z \in \left\{ \left(\mathrm{E}^{[0,l-1]}\right)\overset{-1}{(\Upsilon)} \middle| \Upsilon \in 2^{(Y)^l} \right\} \Leftrightarrow Z \in \Phi^l$$

what proves the statement. □

A.3.2. Proof of Theorem 8.3

To simplify the subsequent proof of Theorem 8.3 we first derive two lemmas which are a direct consequence of Proposition 8.1 which was proven in the previous section.

Lemma A.4. *Given (8.1) it holds that*

$$\forall \Upsilon \in 2^{(Y)^l}, r < l \;.\; \left(\mathrm{E}^{\mathcal{I}_l^l}\right)\overset{-1}{(\Upsilon)} \subseteq \left(\mathrm{E}^{\mathcal{I}_{l-r}^{l-r}}\right)\overset{-1}{(\Upsilon|_{[0,l-r-1]})}. \tag{A.23}$$

Proof. - Pick $x \in \left(\mathrm{E}^{\mathcal{I}_l^l}\right)^{-1}(\Upsilon)$. Using (6.5) this implies that for all $\zeta \in \Upsilon$ there exists $(\nu, \xi) \in \mathcal{B}_{SV}$, $k \in \mathbb{N}_0$ s.t. $x = \xi(k)$ and $\zeta = \nu|_{[k,k+l-1]}$.
- Now observe, that for every choice of ζ, it holds that $\zeta|_{[0,l-r-1]} \in \Upsilon|_{[0,l-r-1]}$. Using the same choice of signals (ν, ξ) and k this immediately implies that $x \in \left(\mathrm{E}^{\mathcal{I}_{l-r}^{l-r}}\right)^{-1}(\Upsilon|_{[0,l-r-1]})$, what proves the statement. □

Lemma A.5. *Given* (8.1) *and* $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ *as in Definition 7.5, it holds that*

$$\forall \widehat{x}, \widehat{x}', u, y \,.\, (\widehat{x}, u, y, \widehat{x}') \in \widehat{\delta}^{l\blacktriangledown} \Rightarrow \begin{pmatrix} \widehat{x}'|_{[0,l-2]} \subseteq \widehat{x}|_{[1,l-1]} \\ \wedge y \in \widehat{x}|_{[0,0]} \end{pmatrix}. \tag{A.24}$$

Proof. - Pick $\widehat{x}, \widehat{x}', u, y$ s.t. $(\widehat{x}, u, y, \widehat{x}') \in \widehat{\delta}^{l\blacktriangledown}$ and define $Z =: \left(\mathrm{E}^{\mathcal{I}_l^l}\right)^{-1}(\widehat{x})$ and $Z' =: \left(\mathrm{E}^{\mathcal{I}_l^l}\right)^{-1}(\widehat{x}')$.
- Using (7.9c), this implies that there exists $x \in Z$ s.t. $y \in \mathrm{H}_\delta(x)$. Furthermore, using (A.23) we know that $x \in \left(\mathrm{E}^{\mathcal{I}_1^1}\right)^{-1}(\widehat{x}|_{[0,0]})$. With this (A.20) implies $y \in \widehat{x}|_{[0,0]}$.
- Now let $\tilde{Z}' = \left(\mathrm{E}^{\mathcal{I}_{l-1}^{l-1}}\right)^{-1}(\widehat{x}'|_{[0,l-2]})$ and recall from (A.23) that $Z' \subseteq \tilde{Z}'$ implying $\mathrm{T}_\delta^{-1}(Z') \subseteq \mathrm{T}_\delta^{-1}(\tilde{Z}')$.
- Using (7.9c) and (7.3) we know that $(\widehat{x}, u, y, \widehat{x}') \in \widehat{\delta}^{l\blacktriangledown}$ implies $Z \cap \mathrm{T}_\delta^{-1}(Z') \neq \emptyset$. Hence, $Z \cap \mathrm{T}_\delta^{-1}(\tilde{Z}') \neq \emptyset$ and therefore (using (7.5b) in Lemma 7.2) $Z \subseteq \mathrm{T}_\delta^{-1}(\tilde{Z}')$.
- This implies that

$$\forall x \in Z \,.\, \exists x' \in \mathrm{T}_\delta(x) \,.\, \widehat{x}'|_{[0,l-2]} = \mathrm{E}^{[0,l-2]}(x') \tag{A.25}$$

- Now recall from (6.5) that

$$\widehat{x}|_{[1,l-1]} = \bigcup_{x \in \mathrm{T}_\delta(Z)} \mathrm{E}^{[0,l-2]}(x)$$

implying $\widehat{x}'|_{[0,l-2]} \subseteq \widehat{x}|_{[1,l-1]}$.

□

Proof of Theorem 8.3

- Pick $\gamma \in \pi_Y(\mathcal{B}_f(\widehat{\mathcal{Q}}^{l\blacktriangledown}))$.
- Using (2.20b) we know that there exist $(\mu, \widehat{\xi})$ s.t. $\widehat{\xi}(0) \in \widehat{X}_0^{l\blacktriangledown}$ and

$$\forall k \in \mathbb{N}_0 \,.\, (\widehat{\xi}(k), \mu(k), \gamma(k), \widehat{\xi}(k+1)) \in \widehat{\delta}^{l\blacktriangledown}.$$

- Using Lemma A.5 we know that for all $k \in \mathbb{N}_0$ it holds that

$$\widehat{\xi}(k+1)|_{[0,l-2]} \subseteq \widehat{\xi}(k)|_{[1,l-1]} \quad \text{and} \quad \gamma(k) \in \widehat{\xi}(k)|_{[0,0]}.$$

Applying these equations iteratively yields $\gamma|_{[k,k+l-1]} \in \widehat{\xi}(k)$.
- Furthermore, we can use (7.9c) and (2.20b) to know that

$$\exists (\gamma', \xi') {\in} \mathcal{B}_{SV}, k' \,.\, \begin{pmatrix} \gamma'(k') {=} \gamma(k) \\ \wedge \widehat{\xi}(k) {=} \mathrm{E}^{\mathcal{I}^l_l}(\xi'(k')) \\ \wedge \widehat{\xi}(k+1) {=} \mathrm{E}^{\mathcal{I}^l_l}(\xi'(k'+1)) \end{pmatrix},$$

- Using these signals, $\gamma|_{[k,k+l-1]} \in \widehat{\xi}(k)$ implies that

$$\exists (\gamma'', \xi'') {\in} \mathcal{B}_{SV}, k'' \,.\, \begin{pmatrix} \gamma''|_{[k'',k''+l-1]} = \gamma|_{[k,k+l-1]} \\ \wedge \xi''(k'') = \xi'(k') \end{pmatrix} \text{ and}$$

$$\exists (\gamma''', \xi''') {\in} \mathcal{B}_{SV}, k''' \,.\, \begin{pmatrix} \gamma'''|_{[k''',k'''+l-1]} = \gamma|_{[k+1,k+l]} \\ \wedge \xi'''(k''') = \xi'(k'+1) \end{pmatrix}$$

Using (2.25) we now obtain

$$\tilde{\gamma} = \gamma'' \wedge^{k''}_{k'} \gamma' \wedge^{k'+1}_{k'''} \gamma''' \in \pi_Y(\mathcal{B}_f(\mathcal{Q})) = \mathcal{B}_V \tag{A.26}$$

where $\gamma|_{[k,k+l]} = \tilde{\gamma}|_{[k',k'+l]}$ and therefore $\gamma|_{[k,k+l]} \in D_{l+1}$.
- Now observe from (7.9b) and (6.5) that $\widehat{\xi}(0) \in \widehat{X}_0^{l\blacktriangledown}$ implies $\widehat{\xi}(0) \subseteq \pi_Y(\mathcal{B}_f(\mathcal{Q}))|_{[0,l-1]}$ and therefore $\gamma|_{[0,l-1]} \in \mathcal{B}_V|_{[0,l-1]}$.
- Combining that $\forall k \in \mathbb{N}_0 \,.\, \gamma|_{[k,k+l]} \in D_{l+1}$ and $\gamma|_{[0,l-1]} \in \mathcal{B}_V|_{[0,l-1]}$ we obtain $\gamma \in \widehat{\mathcal{B}}_Y^{l\uparrow}$ (from (6.2)). $\square$

A.3.3. Proof of Theorem 8.5

We prove both statements separately.

(i) Show $\mathcal{R} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{\mathcal{I}^l_l}, \widehat{\mathcal{Q}}^{l\blacktriangledown}) \Leftrightarrow \widehat{\mathcal{Q}}^{l\blacktriangledown}$ is domino consistent.

• We first show that (5.6a) always holds for $\mathcal{R}$:
– Pick $\zeta \in \widehat{X}_0^{\mathcal{I}^l_l}$. Then it follows from (6.7b) that there exists $x \in X_0$ s.t. $\zeta \in \mathrm{E}^{\mathcal{I}^l_l}(x)$.
– Now Pick $\Upsilon = \mathrm{E}^{\mathcal{I}^l_l}(x)$ and observe that $\Upsilon \in \widehat{X}_0^{l\blacktriangledown}$, which proves the statement.

• Now we need to show that (5.6b) holds for $\mathcal{R}$ and $V = Y$ iff $\widehat{\mathcal{Q}}^{l\blacktriangledown}$ is domino consistent. We show both directions separately:

"$\Rightarrow$"
– Observe that (5.6b) holds for $\mathcal{R}$ and $U \times Y$ iff

$$\forall\, \zeta, \zeta', \Upsilon, u, y \;.\; \left(\begin{array}{c}\zeta \in \Upsilon \\ \wedge(\zeta, u, y, \zeta') \in \widehat{\delta}^{\mathcal{I}^l_l}\end{array}\right) \Rightarrow \exists \Upsilon', u' \;.\; \left(\begin{array}{c}\zeta' \in \Upsilon' \\ \wedge(\Upsilon, u', y, \Upsilon') \in \widehat{\delta}^{l\blacktriangledown}\end{array}\right) \tag{A.27}$$

– Now pick $\tilde{\zeta} \in D_{l+1}$ and $\Upsilon \in 2^{(Y)^l}$ s.t. $\zeta = \tilde{\zeta}|_{[0,l-1]} \in \Upsilon$ and pick $\zeta' = \tilde{\zeta}|_{[1,l]}$ as well as $y = \tilde{\zeta}(0)$. Then it follows from (A.1b) in Lemma A.1 with $m = l$ that there exits u s.t. $(\zeta, u, y, \zeta') \in \widehat{\delta}^{\mathcal{I}^l_l}$, hence the left side of (A.27) holds.
– Now we know from the right side of (A.27) and (7.9c) that there exists $x \in \left(\mathrm{E}^{\mathcal{I}^l_l}\right)^{-1}(\Upsilon)$, u' and x' s.t. $(x, u, y, x') \in \delta$ and $\zeta' \in \mathrm{E}^{\mathcal{I}^l_l}(x')$.
– Now it follows immediately from (6.5) that $y \cdot \zeta' = \tilde{\zeta} \in \mathrm{E}^{[0,l]}(x)$, what proves the statement.
"$\Leftarrow$"
– Pick $(\zeta, \Upsilon) \in \mathcal{R}$, i.e., $\zeta \in \Upsilon$ and u, y, ζ' s.t. $(\zeta, u, y, \zeta') \in \widehat{\delta}^{\mathcal{I}^l_l}$.
– Now it follows from Lemma A.1 that $y \cdot \zeta' \in D_{l+1}$.
– Using (8.4) therefore implies the existence of $x \in \left(\mathrm{E}^{\mathcal{I}^l_l}\right)^{-1}(\Upsilon)$ s.t. $y \cdot \zeta \in \mathrm{E}^{[0,l]}(x)$.
– Using (6.5) this implies that there exists $(\mu', \gamma', \xi') \in \mathcal{B}_V$ and $k' \in \mathbb{N}_0$ s.t.

$$\Upsilon = \mathrm{E}^{\mathcal{I}^l_l}(\xi'(k')) \text{ and } \zeta' \in \mathrm{E}^{\mathcal{I}^l_l}(\xi'(k'+1))$$

– Now pick $\Upsilon' = \mathrm{E}^{\mathcal{I}^l_l}(\xi'(k'+1))$ and observe that $\zeta' \in \Upsilon'$.
– Moreover, using (7.9c) with $x = \xi'(k')$, $x' = \xi'(k'+1)$ and $u' = \mu'(k')$ immediately implies $(\Upsilon, u', y, \Upsilon') \in \widehat{\delta}^{l\blacktriangledown}$, what proves the statement.

(ii) $\mathcal{R}^{-1} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{l\blacktriangledown}, \widehat{\mathcal{Q}}^{\mathcal{I}^l_l}) \Leftrightarrow \mathcal{Q}$ is future unique w.r.t. $\mathcal{I}^l_l$.
"$\Rightarrow$"
– Let $\mathcal{R}^{\mathcal{I}^l_l}$ and $\mathcal{R}^{\triangledown}$ be equivalent to the relations in (6.12) and (7.8) (with $m = l$), respectively. Then it follows from Theorem 6.10 and Theorem 7.7 that

$$\mathcal{R}^{\triangledown} \in \mathfrak{S}_Y(\mathcal{Q}, \widehat{\mathcal{Q}}^{l\blacktriangledown}), \quad \text{and} \tag{A.28a}$$

$$\mathcal{R}^{\mathcal{I}^l_l} \in \mathfrak{S}_Y(\mathcal{Q}, \widehat{\mathcal{Q}}^{\mathcal{I}^l_m}) \Leftrightarrow \mathcal{Q} \text{ is future unique w.r.t. } \mathcal{I}^l_l. \tag{A.28b}$$

– Now take $\mathcal{R}$ as in (8.5) observe that

$$\begin{aligned}\mathcal{R}^{\triangledown} \circ \mathcal{R}^{-1} &= \left\{ (x,\widehat{x}) \in X \times \widehat{X}^{\mathcal{I}_l^l} \middle| \exists \Upsilon \in \widehat{X}^{l\blacktriangledown} \, . \, \begin{pmatrix} (x,\Upsilon) \in \mathcal{R}^{\triangledown} \\ \wedge (\Upsilon,\widehat{x}) \in \mathcal{R}^{-1} \end{pmatrix} \right\} \\ &= \left\{ (x,\widehat{x}) \in X \times \widehat{X}^{\mathcal{I}_l^l} \middle| \exists \Upsilon \in \widehat{X}^{l\blacktriangledown} \, . \, \begin{pmatrix} \Upsilon = \mathrm{E}^{\mathcal{I}_l^l}(x) \\ \wedge \widehat{x} \in \Upsilon \end{pmatrix} \right\} \\ &= \left\{ (x,\widehat{x}) \in X \times \widehat{X}^{\mathcal{I}_l^l} \middle| \widehat{x} \in \mathrm{E}^{\mathcal{I}_l^l}(x) \right\} \\ &= \mathcal{R}^{\mathcal{I}_l^l}\end{aligned}$$

– Now combining (A.28) with

$$\begin{pmatrix} \mathcal{R}^{\triangledown} \in \mathfrak{S}_Y(\mathcal{Q}, \widehat{\mathcal{Q}}^{l\blacktriangledown}) \\ \wedge \mathcal{R}^{-1} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{l\blacktriangledown}, \widehat{\mathcal{Q}}^{\mathcal{I}_l^l}) \end{pmatrix} \Rightarrow \mathcal{R}^{\triangledown} \circ \mathcal{R}^{-1} \in \mathfrak{S}_Y(\mathcal{Q}, \widehat{\mathcal{Q}}^{\mathcal{I}_l^l})$$

gives

$$\mathcal{R}^{-1} \in \mathfrak{S}_Y(\widehat{\mathcal{Q}}^{l\blacktriangledown}, \widehat{\mathcal{Q}}^{\mathcal{I}_l^l}) \Rightarrow \mathcal{Q} \text{ is future unique w.r.t. } \mathcal{I}_l^l.$$

"$\Leftarrow$"

– We know that $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_l^l$. Using (6.6), this implies that that for all $x \in X$ holds

$$\mathrm{E}^{\mathcal{I}_l^l}(x) \neq \emptyset \Rightarrow |\mathrm{E}^{\mathcal{I}_l^l}(x)| = 1. \tag{A.29}$$

Using (7.9a) this immediately implies $|\Upsilon| = 1$ for all $\Upsilon \in \widehat{X}^{l\blacktriangledown}$. Therefore (8.5) becomes

$$\mathcal{R} = \left\{ (\zeta, \Upsilon) \in \widehat{X}^{\mathcal{I}_l^l} \times \widehat{X}^{l\blacktriangledown} \middle| \Upsilon = \{\zeta\} \right\}. \tag{A.30}$$

– Show that (5.6a) holds for $\mathcal{R}^{-1}$:
∗ Pick $\Upsilon \in \widehat{X}_0^{l\blacktriangledown}$. Then it follows from (7.9b) that there exists $x \in X_0$ s.t. $\Upsilon = \mathrm{E}^{\mathcal{I}_l^l}(x)$.
∗ Now pick $\zeta \in \Upsilon$ and observe that $\zeta \in \mathrm{E}^{\mathcal{I}_l^l}(x)$, i.e., $\zeta \in \widehat{X}_0^{\mathcal{I}_l^l}$ (from (6.7b)).
– Show that (5.6b) holds for $\mathcal{R}^{-1}$:
∗ Pick $(\Upsilon, \zeta) \in \mathcal{R}^{-1}$, i.e., $\Upsilon = \{\zeta\}$ and u, y, Υ', ζ' s.t. $(\Upsilon, u, y, \Upsilon') \in \widehat{\delta}^{l\blacktriangledown}$ and $\Upsilon' = \{\zeta'\}$.
∗ Using (A.30) this immediately implies that $(\Upsilon', \zeta') \in \mathcal{R}^{-1}$.
∗ Now (7.9c) implies the existence of x, x' s.t. $\{\zeta\} = \mathrm{E}^{\mathcal{I}_l^l}(x)$, $\{\zeta'\} = \mathrm{E}^{\mathcal{I}_l^l}(x')$ and $(x, u, y, x') \in \delta$. Using (6.7c) this immediately implies $(\zeta, u, y, \zeta') \in \widehat{\delta}^{\mathcal{I}_l^l}$, what proves the statement. □

A.3.4. Proof of Proposition 8.8

First observe that (i) follows from Corollary 8.6, (ii) follows from (A.30) in the proof of Theorem 8.5 and (iv) follows from (iii) using Theorem 6.10 and Theorem 7.7. Hence, we only prove (iii).
Let $\mathcal{R}$ as in (8.8) and observe that

$$\begin{aligned}
\mathcal{R}^{l\Uparrow} \circ \mathcal{R} &= \left\{ (x, \Upsilon) \in X \times \widehat{X}^{l\blacktriangledown} \middle| \exists \zeta \in \widehat{X}^{\mathcal{I}_l^l} \,.\, \begin{pmatrix} (x, \zeta) \in \mathcal{R}^{l\Uparrow} \\ \wedge (\zeta, \Upsilon) \in \mathcal{R} \end{pmatrix} \right\} \\
&= \left\{ (x, \Upsilon) \in X \times \widehat{X}^{l\blacktriangledown} \middle| \exists \zeta \in \widehat{X}^{\mathcal{I}_l^l} \,.\, \begin{pmatrix} \zeta \in \mathrm{E}^{\mathcal{I}_l^l}(x) \\ \wedge \Upsilon = \{\zeta\} \end{pmatrix} \right\} \\
&= \left\{ (x, \Upsilon) \in X \times \widehat{X}^{l\blacktriangledown} \middle| \Upsilon = \mathrm{E}^{\mathcal{I}_l^l}(x) \right\} \\
&= \mathcal{R}^{l\blacktriangledown}
\end{aligned}$$

where the second last equality results from (A.29) as $\mathcal{Q}$ is future unique w.r.t. $\mathcal{I}_l^l$. Using the same chain of equalities equivalently yields $\mathcal{R}^{-1} \circ \left(\mathcal{R}^{l\Uparrow}\right)^{-1} = \left(\mathcal{R}^{l\blacktriangledown}\right)^{-1}$. Similarily, we obtain

$$\begin{aligned}
\mathcal{R}^{l\blacktriangledown} \circ \mathcal{R}^{-1} &= \left\{ (x, \zeta) \in X \times \widehat{X}^{\mathcal{I}_l^l} \middle| \exists \Upsilon \in \widehat{X}^{l\blacktriangledown} \,.\, \begin{pmatrix} (x, \Upsilon) \in \mathcal{R}^{l\blacktriangledown} \\ \wedge (\Upsilon, \zeta) \in \mathcal{R}^{-1} \end{pmatrix} \right\} \\
&= \left\{ (x, \zeta) \in X \times \widehat{X}^{\mathcal{I}_l^l} \middle| \exists \Upsilon \in \widehat{X}^{l\blacktriangledown} \,.\, \begin{pmatrix} \Upsilon = \mathrm{E}^{\mathcal{I}_l^l}(x) \\ \wedge \Upsilon = \{\zeta\} \end{pmatrix} \right\} \\
&= \left\{ (x, \zeta) \in X \times \widehat{X}^{\mathcal{I}_l^l} \middle| \zeta \in \mathrm{E}^{\mathcal{I}_l^l}(x) \right\} \\
&= \mathcal{R}^{l\Uparrow}
\end{aligned}$$

Using the same chain of equalities equivalently yields $\mathcal{R} \circ \left(\mathcal{R}^{l\blacktriangledown}\right)^{-1} = \left(\mathcal{R}^{l\Uparrow}\right)^{-1}$.
With this observations (iii) follows immediately from the transitivity of simulation relations proven in Theorem 5.3.

B. Appendix Part II

B.1. Proofs for Chapter 11

In this section we provide detailed proofs for the claims made in Chapter 11.

B.1.1. Proof of Lemma 11.14

To simplify notation, we collect all states q, reachable by a finite sequence of λ-transitions from a configuration $(\tilde{q}, w, \tilde{s})$ with $\tilde{q} \in F$ in the set

$$Q_{rlt}(M, w) := \left\{ q \in Q \;\middle|\; \begin{array}{l} \exists\, f \in \mathcal{D}_{\mathrm{I}}(M), n \in \mathbf{N}, \tilde{q} \in F \;. \\ \quad \left(\begin{array}{l} f(n) = (\tilde{e}, (\tilde{q}, w, \tilde{s})) \\ \wedge \exists n' > n \;.\; f(n') = (e, (q, w, s)) \end{array} \right) \end{array} \right\}. \tag{B.1}$$

(i) $\mathcal{L}_{um}(M_{Sp}) = \mathcal{L}_{um}(M)$: Let $q \in Q$, $w \in \Sigma^*$, $\gamma \in \Gamma$, $s \in \Gamma^*$ and $(q, w, \gamma \cdot s)$ be an arbitrary configuration reachable by the initial derivation $f \in \mathcal{D}_{\mathrm{I}}(M)$, implying $w \in \mathcal{L}_{um}(M)$. Observe that the state q is mapped to the states $\{\langle q \rangle_r, \langle q, \gamma \rangle_r\}$ if $q \notin Q_{rlt}(M, w)$ and to $\{\langle q \rangle_s, \langle q, \gamma \rangle_s\}$ if $q \in Q_{rlt}(M, w)$[1]. Using this, we can show that the single-step relation

$$(e, (q, w, \gamma \cdot s)) \vdash_M ((q, \sigma, \gamma, s', q'), (q', w \cdot \sigma, s' \cdot s)) \tag{B.2}$$

is mimicked by a sequence of two single-step relations in M_{Sp} in four different ways

(a) $(q \notin Q_{rlt}(M, w) \wedge (\sigma \neq \lambda \vee q \notin F))$:

$$\begin{array}{l} (\breve{e}, (\langle q \rangle_r, w, \gamma \cdot s)) \vdash_{M_{Sp}} ((\langle q \rangle_r, \lambda, \gamma, \gamma, \langle q, \gamma \rangle_r), (\langle q, \gamma \rangle_r, w, \gamma \cdot s)) \\ \quad \vdash_{M_{Sp}} ((\langle q, \gamma \rangle_r, \sigma, \gamma, s', \langle q' \rangle_r), (\langle q' \rangle_r, w \cdot \sigma, s' \cdot s)) \end{array}$$

(b) $(q \in F \wedge \sigma = \lambda)$:

$$\begin{array}{l} (\breve{e}, (\langle q \rangle_r, w, \gamma \cdot s)) \vdash_{M_{Sp}} ((\langle q \rangle_r, \lambda, \gamma, \gamma, \langle q, \gamma \rangle_r), (\langle q, \gamma \rangle_r, w, \gamma \cdot s)) \\ \quad \vdash_{M_{Sp}} ((\langle q, \gamma \rangle_r, \sigma, \gamma, s', \langle q' \rangle_s), (\langle q' \rangle_s, w \cdot \sigma, s' \cdot s)) \end{array}$$

[1] Observe that reaching q with two different initial derivations $f, f' \in \mathcal{D}_{\mathrm{I}}(M)$ might result in different mappings as the condition in (B.1) might only hold for one out of the two.

(c) $(q \in Q_{rlt}(M,w) \wedge \sigma = \lambda)$:

$$(\breve{e}, (\langle q\rangle_s, w, \gamma\cdot s)) \vdash_{M_{Sp}} ((\langle q\rangle_s, \lambda, \gamma, \gamma, \langle q,\gamma\rangle_s), (\langle q,\gamma\rangle_s, w, \gamma\cdot s)) \vdash_{M_{Sp}} ((\langle q,\gamma\rangle_s, \sigma, \gamma, s', \langle q'\rangle_s), (\langle q'\rangle_s, w\cdot\sigma, s'\cdot s))$$

(d) $(q \in Q_{rlt}(M,w) \wedge \sigma \neq \lambda)$:

$$(\breve{e}, (\langle q\rangle_s, w, \gamma\cdot s)) \vdash_{M_{Sp}} ((\langle q\rangle_s, \lambda, \gamma, \gamma, \langle q,\gamma\rangle_s), (\langle q,\gamma\rangle_s, w, \gamma\cdot s)) \vdash_{M_{Sp}} ((\langle q,\gamma\rangle_s, \sigma, \gamma, s', \langle q'\rangle_r), (\langle q'\rangle_r, w\cdot\sigma, s'\cdot s))$$

Therefore, using induction, we can always construct a derivation $f' \in \mathcal{D}_{\mathrm{I}}(M_{Sp})$, $n' \in \mathbf{N}$ s.t. $f'(n') = (e', (q', w, \gamma\cdot s))$ giving $w \in \mathcal{L}_{um}(M_{Sp})$ and therefore, $\mathcal{L}_{um}(M_{Sp}) \subseteq \mathcal{L}_{um}(M)$. Using the fact that the relations in case (a)-(d) only occur, iff there exists a matching relation (B.2) in M and derivations in M_{Sp} can only be concatenated iff this is possible in M, the construction of single-step relations in M matching single-step relations in M_{Sp} gives the same cases as before, implying $\mathcal{L}_{um}(M) \subseteq \mathcal{L}_{um}(M_{Sp})$.
(ii) Show $\mathcal{L}_m(M_{Sp}) = \mathcal{L}_m(M)$: Let $w \in \mathcal{L}_m(M)$ and $f \in \mathcal{D}_{\mathrm{I}}(M)$, $n \in \mathbf{N}$, $q \in F$ s.t. $f(n) = (e, (q, w, \gamma\cdot s))$ is the starting configuration in (B.2). If $\sigma \neq \lambda$ in (B.2), we know from part (i) and cases (a) and (d) that there exists a derivation $f' \in \mathcal{D}_{\mathrm{I}}(M_{Sp})$, $n' \in \mathbf{N}$ s.t. $f'(n') = (\breve{e}, (\langle q,\gamma\rangle_r, w, \gamma\cdot s))$ with $\langle q,\gamma\rangle_r \in F_{Sp}$ (as $q \in F$ and $\langle q,\gamma\rangle_r \notin Q_\lambda(M_{Sp})$), giving $w \in \mathcal{L}_m(M_{Sp})$. Now let $\sigma = \lambda$. Since $M \notin$ SBA, there exists $\sigma' \neq \lambda$ and a finite chain of singe-step relations, s.t.

$$(\breve{e}, (q, w, \gamma\cdot s)) \vdash_M^* (\tilde{e}, (\tilde{q}, w, \tilde{\gamma}\cdot\tilde{s})) \vdash_M ((\tilde{q}, \sigma', \tilde{\gamma}, \tilde{s}', \tilde{q}'), (\tilde{q}', w\cdot\sigma', \tilde{s}'\cdot\tilde{s})) \tag{B.3a}$$
$$\text{or} \quad (\breve{e}, (q, w, \gamma\cdot s)) \vdash_M^* (\hat{e}, (\hat{q}, w, \hat{\gamma}\cdot\hat{s})) \not\vdash_M . \tag{B.3b}$$

Then we can combine case (b),(c) and (d) from part (i) to mimic (B.3a) by

$$(\breve{e}, (\langle q\rangle, w, \gamma\cdot s)) \vdash_{M_{Sp}}^* (\tilde{h}, (\langle \tilde{q},\tilde{\gamma}\rangle_s, w, \tilde{\gamma}\cdot\tilde{s})) \vdash_{M_{Sp}} ((\langle \tilde{q},\tilde{\gamma}\rangle_s, \sigma', \tilde{\gamma}, \tilde{s}', \langle\tilde{q}'\rangle_r), (\langle\tilde{q}'\rangle_r, w\sigma', \tilde{s}'\cdot\tilde{s}))$$

and (B.3b) by the finite chain $(\perp, (\langle q\rangle, w, \gamma\cdot s)) \vdash_{M_{Sp}}^* (\hat{h}, (\langle\hat{q},\hat{\gamma}\rangle_s, w, \hat{\gamma}\cdot\hat{s})) \not\vdash_{M_{Sp}}$.
Now observe that all states in $\mathcal{A}_s(Q,\Gamma) \cap Q_\lambda(Q_{Sp})$ can *only* have outgoing λ-transitions, due to the determinism of Q that is preserved in Q_{Sp} (from (iii)). Therefore, we have $\langle\tilde{q},\tilde{\gamma}\rangle_s, \langle\hat{q},\hat{\gamma}\rangle_s \notin Q_\lambda(M_{Sp})$, hence $\langle\tilde{q},\tilde{\gamma}\rangle_s, \langle\hat{q},\hat{\gamma}\rangle_s \in F_{Sp}$ by definition. This implies that there exists $f' \in \mathcal{D}_{\mathrm{I}}(M_{Sp})$ and $n'' \in \mathbf{N}$ with $f'(n') = (h, (p, w, r))$ s.t. $p \in F_{Sp}$, implying $w \in \mathcal{L}_m(M_{Sp})$, hence $\mathcal{L}_m(M) \subseteq \mathcal{L}_m(M_{Sp})$. For the proof of $\mathcal{L}_m(M_{Sp}) \subseteq \mathcal{L}_m(M)$ observe that if w is accepted by a state $\langle q,\gamma\rangle_r \in F_{Sp}$, it follows from (i) and the construction of F_{Sp} that w is accepted by $q \in F$ in M. Now let w be accepted by a state $\langle q,\gamma\rangle_s \in F_{Sp}$. Then it follows from case (b) in the proof of (i) that there exists $p \in F$ accepting w,

since otherwise, $\langle q, \gamma \rangle_s \in F_{Sp}$ is not reachable by w, hence $w \in \mathcal{L}_m(M)$.
Since we only split states, redirect existing transitions and add unique λ-transitions, **(iii)** and **(v)** follow immediately from the construction. Furthermore, observe from cases (a)-(d) in the proof of statement (i) that every added λ-transition (from main to auxiliary states) is preceded and followed by a transition from M. Hence **(iv)** follows from $M \notin$ SBA.

B.1.2. Proof of Theorem 11.19

Let us define $M_{Ac} = \mathrm{AC}(\mathrm{SPLIT}(M_P \times M_O))$ and $M'_O = \mathrm{RNCS}(M_{Ac})$.
Observe that the premises imply $\mathcal{L}_m(M_O) \subseteq \mathcal{L}_m(M_P)$ and $\mathcal{L}_{um}(M_O) \subseteq \overline{\mathcal{L}_m(M_O)}$ (from Definition 10.4, 10.1 and 11.7). Furthermore, recall from Definition 11.7 and Definition 10.8 that (11.6) is true iff

$$\mathcal{L}_{um}(M'_O) = \Psi(\mathcal{L}_{um}(M_O)) \quad \text{and} \quad \mathcal{L}_m(M'_O) = \Omega(\mathcal{L}_m(M_O)), \tag{B.4}$$

which remains to be shown using the following steps:

(A) From Lemma 11.12, 11.14 and 11.17, follows that $\mathcal{L}_{um}(M_{Ac}) = \mathcal{L}_{um}(M_O)$ and $\mathcal{L}_m(M_{Ac}) = \mathcal{L}_m(M_O)$.

(B) Pick $w \in \mathcal{L}_{um}(M_O)$ with $\neg\,\mathrm{ContS}(\mathcal{L}_{um}(M_O), L_{\mathrm{Pum}}, w)$. Using Lemma 11.15, we can fix $f_w \in \mathcal{D}_{max,M_{Ac}}(w)$ and its final state $\langle (p,q), \gamma \rangle \in ((Q_{Ac} \cap \mathcal{A}(Q_\times, \Gamma)) \setminus Q_\lambda(M_{Ac}))$. Using Definition 11.11 this implies that M_P only accepts w in p. Now Definition 10.4, (11.2b) and (A) imply that there exists $\mu \in \Sigma_{\mathsf{uc}}$ s.t. $(p, \mu, \square, \square, p') \in \delta_P$ and $\forall s, r \;.\; (\langle (p,q), \gamma \rangle, \mu, \gamma, s, r) \notin \delta_{Ac}$, giving that $\langle (p,q), \gamma \rangle \in \mathrm{NCS}$.

(C) It follows from the construction in Definition 11.13 that there always exists a non-maximal derivation generating w and ending in the main state $\langle (p,q) \rangle \in Q_\lambda(M_{Ac})$ which is neither marking nor considered in NCS.

(D) Definition 11.18 implies that $w \in \mathcal{L}_{um}(M'_O)$ or $w \in \mathcal{L}_m(M'_O)$ iff there exists a derivation $f' \in \mathcal{D}(M_{Ac})$ with final configuration (r, w, s) s.t. $\forall n \in \mathrm{dom(f')}$ holds that $\pi_2(f'(n)) \notin \mathrm{NCS}$. Now observe that for $w \in \mathcal{L}_m(M'_O)$ f' needs to be maximal while for $w \in \mathcal{L}_{um}(M'_O)$ there always exists a non-maximal derivation (from (C)). Now $M_{Ac} \in$ DPDA implies that for all prefixes $w' \sqsubset w$ (resp. $w' \sqsubseteq w$) holds that their maximal derivations are a prefix of f' (where $w' \sqsubseteq w$ only holds if f' is maximal). Therefore, using (B) implies

$$w \in \mathcal{L}_{um}(M'_O) \Rightarrow \big(\forall w' \sqsubset w \;.\; \mathrm{ContS}(\mathcal{L}_{um}(M_O), L_{\mathrm{Pum}}, w')\big) \quad \text{and} \tag{B.5a}$$

$$w \in \mathcal{L}_m(M'_O) \Rightarrow \big(\forall w' \sqsubseteq w \;.\; \mathrm{ContS}(\mathcal{L}_{um}(M_O), L_{\mathrm{Pum}}, w')\big)\,. \tag{B.5b}$$

(E) Now assume $\neg\,\mathrm{ContS}(\mathcal{L}_{um}(M_O), L_{\mathrm{Pum}}, w)$ and all $w' \sqsubset w$ are controllable. From (B) follows that $\langle (p,q), \gamma \rangle \in \mathrm{NCS}$ is removed. This has two consequences:

(a) we still have $w \in \mathcal{L}_{um}(M'_O)$ from (C), i.e.,

$$\big(\forall w' \sqsubset w \,.\, \mathrm{ContS}(\mathcal{L}_{um}(M_O), L_{\mathrm{Pum}}, w')\big) \Rightarrow w \in \mathcal{L}_{um}(M'_O), \tag{B.6a}$$

and

(b) if $w \in \mathcal{L}_m(M_O)$ we have $w \notin \mathcal{L}_m(M'_O)$ from (B) and Lemma 11.15, i.e., w is only contained in $\mathcal{L}_m(M_O)$ if $\mathrm{ContS}(\mathcal{L}_{um}(M_O), L_{\mathrm{Pum}}, w)$ giving

$$\big(\forall w' \sqsubseteq w \,.\, \mathrm{ContS}(\mathcal{L}_{um}(M_O), L_{\mathrm{Pum}}, w')\big) \Rightarrow w \in \mathcal{L}_m(M'_O). \tag{B.6b}$$

(F) Finally, combining (B.5) and (B.6) obviously yields (B.4), which proves the statement.

B.2. Counterexample

In this appendix, the algorithm presented in [20, p.827] and [19, p.64] is applied to an example. This example appeared previously in [55] and [56]. It will be shown that this represents a counterexample to [20, Theorem 3.5] and [19, Theorem 5.2.5] since the final DPDA does not realize the supremal controllable sublanguage of the given prefix closed deterministic context free specification language (which is required to be a subset of the given prefix closed regular plant language). Therefore we claim that the problem of automatically calculating a supremal controllable sublanguage of a DCFL was not solved by Griffin in [19, 20].
The algorithm is initialized with a DFA G and a DPDA M realizing the plant and the specification, respectively, s.t. $\mathcal{L}_m(G) = \mathcal{L}_{um}(G)$, $\mathcal{L}_m(M) = \mathcal{L}_{um}(M)$ and $\mathcal{L}_m(M) \subseteq \mathcal{L}_m(G)$. Observe that the DFA G and the DPDA M depicted in Fig. B.1 satisfy these requirements since their languages are given by

$$\mathcal{L}_m(G) = \mathcal{L}_{um}(G) = \{a^n, a^n b, a^n bu | n \in \mathbf{N}\} \quad \text{and}$$
$$\mathcal{L}_m(M) = \mathcal{L}_{um}(M) = \left\{ \begin{array}{l} a^n, a^m b, a^k bu \mid \\ \quad n, m, k \in \mathbf{N}, m > 0, k > 1 \end{array} \right\},$$

where $\Sigma_{\mathsf{c}} = \{a, b\}$ and $\Sigma_{\mathsf{uc}} = \{u\}$. Using these automata, the construction follows seven steps.

1. Construct M', depicted in Fig. B.2, by making M scan its entire input, using the algorithm in [24, Lemma10.3.].
2. Construct a DPDA M'' that accepts the complement of $\mathcal{L}_m(M)$ using the algorithm in [24, Thm.10.1]. For the considered example M'' is identical to M' in Fig. B.2 but with exchanged non-marking and marking states, i.e., $F'' = \{q_d\}$.
3. Construct a DPDA M''' that accepts $\mathrm{L}^{\mathrm{c}}_{\mathrm{m}}(M) \cap \mathcal{L}_m(G)$, i.e., calculate the cross product of G and M'' using the algorithm in [24, Thm.6.5].

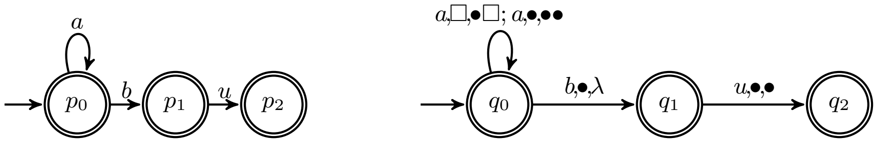

Figure B.1.: DFA G (left) and DPDA M (right) realizing plant and specification, respectively.

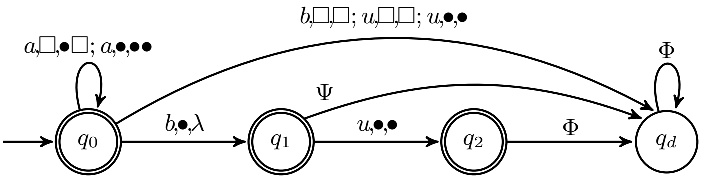

Figure B.2.: DPDA M', with $\Psi = \Phi \setminus u{,}\bullet{,}\bullet$ and $\Phi = \{a{,}\square{,}\square;\, a{,}\bullet{,}\bullet;\, b{,}\square{,}\square;\, b{,}\bullet{,}\bullet;\, u{,}\square{,}\square;\, u{,}\bullet{,}\bullet\}$.

4. Construct M_1, depicted in Fig. B.3, as the accessible part of M''', using the algorithm in [18, Thm.4.1].
5. Construct a predicting machine to observe so called μ-reverse paths using the algorithm in [24, p.240]. For the considered example, the construction simply defines an additional stack symbol μ_γ, $\forall \gamma \in \Gamma$ s.t.

$$\mu_\gamma := \left\{ q \in Q_1 \backslash F_1 \middle| \exists q' \in F_1, v \in \Sigma_{\mathsf{uc}}^* \,.\, (\bot, (q, w, \gamma \cdot s)) \vdash_{M_1}^* (\bot, (q', w \cdot v, s')) \right\}$$

   which denotes the set of unmarked states in M_1 from which a derivation starting with stack-top γ, generating a sequence of uncontrollable symbols $v \in \Sigma_{\mathsf{uc}}^*$ and reaching a marking state q' (i.e., a so called μ-reverse path), exists. For M_1, depicted in Fig. B.3, this gives $\mu_\square = \{(p_1, q_1)\}$ and $\mu_\bullet = \emptyset$. The predicting machine M_1^μ, depicted in Figure B.4, is then identical to M_1 but uses pairs $[\gamma, \mu_\gamma]$ as stack.
6. Construct M_2, depicted in Fig. B.5, by deleting all transitions

$$\delta_{cp} := \left\{ (q, \sigma, \gamma, \gamma' \cdot s, q') \in \delta_M \middle| \begin{pmatrix} (q, \sigma, \gamma, [\gamma', \mu_{\gamma'}] \cdot s, q') \in \delta_{M_1^\mu} \\ \wedge \sigma \in \Sigma_{\mathsf{c}} \wedge q' \in \mu_{\gamma'} \end{pmatrix} \right\}$$

   in M which produce a stack-top in q' which enables a μ-reverse path starting in q'. For M and M_1^μ, depicted in Fig. B.1 and B.4, respectively, observe that $e =$

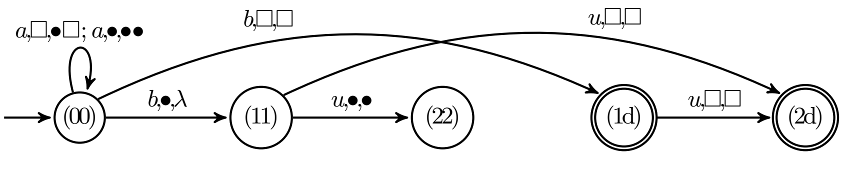

Figure B.3.: DPDA M_1, with $(ij) := (p_i, q_j)$

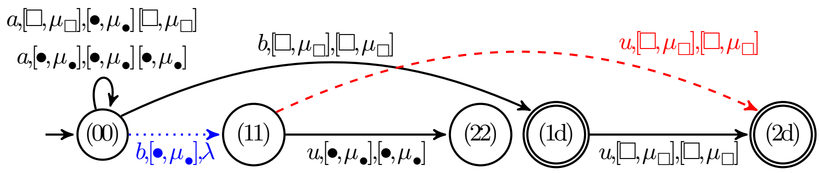

Figure B.4.: DPDA M_1^{μ} with $\mu_\square = \{(p_1, q_1)\}$ and $\mu_\bullet = \emptyset$. The set of μ-reverse paths is depicted in red (dashed) while the set of edges in δ_{cp} is depicted in blue (dotted).

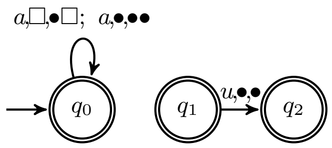

Figure B.5.: DPDA M_2

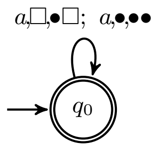

Figure B.6.: DPDA M_3

$((p_0, q_0), b, [\bullet, \mu_\bullet], \lambda, (p_1, q_1)) \in \delta_{M_1^{\mu}}$ is the only ingoing transition to (p_1, q_1) (where the only μ-reverse path starts for stack-top $\square$, since $\mu_\square = \{(p_1, q_1)\}$) and, since M_1 is trim, eventually leads to the stack-top $\square$ in (p_1, q_1). Using the corresponding transition to e in M, this gives $\delta_{cp} = \{(q_0, b, \bullet, \lambda, q_1)\}$. By deleting δ_{cp} in M, we obtain M_2, depicted in Fig. B.5.

7. Construct M_3, depicted in Fig. B.6, as the accessible part of M_2, using the algorithm in [18, Thm.4.1]. If δ_{cp} in step 6 is empty, the algorithm terminates. Otherwise, the algorithm is restarted with $M = M_3$.

Obviously, M_3 does not have further controllability problems. Therefore, the algorithm would redo steps 1-6 and then return M_3.

Now observe that the specification language $\mathcal{L}_m(M)$ restricts the plant language $\mathcal{L}_m(G)$ such that u cannot occur after exactly one a. This generates a controllability problem for the word ab only. Using (10.4) in Section 10.2, the supremal controllable sublanguage of $\mathcal{L}_m(M)$ for this example is given by

$$\begin{aligned}\mathcal{K} &= \{w \in \mathcal{L}_m(M) \mid \forall w' \sqsubseteq w \,.\, w' \neq ab\} = \mathcal{L}_m(M) \setminus \{ab\} \\ &= \left\{a^n, a^m b, a^k bu \middle| n, m, k \in \mathbf{N}, m, k > 1\right\}\end{aligned} \tag{B.7}$$

implying that $\mathcal{L}_m(M_3) = \{a^n \mid n \in \mathbf{N}\}$ is a strict subset of $\mathcal{K}$ which is an obvious contradiction to [20, Thm.3.5]. Furthermore, $\mathcal{K}$ in (B.7) cannot be realized using the state and transition structure of M and only deleting existing transitions.

The automatic synthesis of a DPDA realizing $\mathcal{K}$ for this example is provided as an example within our pushdown-plug-in for `libFAUDES` [31].

Bibliography

[1] R. Alur and D. Dill. Automata for modeling real-time systems. In *Automata, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer Berlin Heidelberg, 1990.

[2] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971 –984, jul. 2000. ISSN 0018-9219.

[3] P. J. Antsaklis, J. A. Stiver, and M. Lemmon. Hybrid system modeling and autonomous control systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 366–392. Springer Berlin Heidelberg, 1993.

[4] C. Belta and L. C. G. J. M. Habets. Controlling a class of nonlinear systems on rectangles. *IEEE Transactions on Automatic Control*, 51(11):1749–1759, Nov 2006.

[5] Y.-L. Chen and F. Lin. Modeling of discrete event systems using finite state machines with parameters. In *Control Applications, 2000. Proceedings of the 2000 IEEE International Conference on*, pages 941–946, 2000. doi: 10.1109/CCA.2000.897591.

[6] Y.-L. Chen and F. Lin. Safety control of discrete event systems using finite state machines with parameters. In *American Control Conference, 2001. Proceedings of the 2001*, volume 2, pages 975–980 vol.2, 2001. doi: 10.1109/ACC.2001.945847.

[7] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, September 1956.

[8] J. M. Davoren and P. Tabuada. On simulations and bisimulations of general flow systems. In *Hybrid Systems: Computation and Control*, pages 145–158. Springer, 2007.

[9] J. M. Davoren, V. Coulthard, N. Markey, and T. Moor. Non-deterministic temporal logics for general flow systems. In *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 280–295. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-21259-1.

[10] R. Ehlers, S. Lafortune, S. Tripakis, and M. Vardi. Reactive synthesis vs. supervisory control: Bridging the gap. Technical Report 162, University of California at Berkeley, 2013. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-162.pdf.

[11] J.-C. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13(2–3):219 – 236, 1990. ISSN 0167-6423.

[12] D. Franke, T. Moor, and J. Raisch. Discrete supervisory control of switched linear systems. *at-Automatisierungstechnik Methoden und Anwendungen der Steuerungs-, Regelungs-und Informationstechnik*, 48(9/2000):460, 2000.

[13] A. Girard and G. J. Pappas. Approximation metrics for discrete and continuous systems. *Automatic Control, IEEE Transactions on*, 52(5):782 –798, may 2007.

[14] A. Girard, A. A. Julius, and G. J. Pappas. Approximate simulation relations for hybrid systems. *Discrete Event Dynamic Systems*, 18(2):163–179, 2008.

[15] A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, 55(1):116–126, 2010.

[16] A. Giua and F. DiCesare. Blocking and controllability of petri nets in supervisory control. *IEEE Transactions on Automatic Control*, 39(4):818–823, 1994.

[17] A. Giua and F. DiCesare. Decidability and closure properties of weak petri net languages in supervisory control. *IEEE Transactions on Automatic Control*, 40(5): 906–910, 1995.

[18] C. Griffin. A note on deciding controllability in pushdown systems. *IEEE Transactions on Automatic Control*, 51(2):334 – 337, feb. 2006.

[19] C. Griffin. *Decidability and optimality in pushdown control systems: A new approach to discrete event control*. PhD thesis, The Pensylvania State University, 2007.

[20] C. Griffin. A note on the properties of the supremal controllable sublanguage in pushdown systems. *IEEE Transactions on Automatic Control*, 53(3):826 –829, apr. 2008. ISSN 0018-9286.

[21] L. C. G. J. M. Habets, P. J. Collins, and J. H. van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*, 51(6):938–948, June 2006.

[22] E. Haghverdi, P. Tabuada, and G. J. Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theoretical Computer Science*, 342(2-3):229 – 261, 2005.

[23] L. E. Holloway, B. H. Krogh, and A. Giua. A survey of petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems*, 7:151–190, 1997.

[24] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, languages and computation*. Addison-Wesley Publishing company, 1979.

[25] M. Iordache and P. Antsaklis. Supervision based on place invariants: A survey. *Discrete Event Dynamic Systems*, 16:451–492, 2006.

[26] A. A. Julius. *On interconnection and equivalence of continuous and discrete systems: a behavioral perspective*. PhD thesis, University of Twente, 2005.

[27] A. A. Julius and A. J. van der Schaft. Bisimulation as congruence in the behavioral setting. *Proc. 44th IEEE Conf. on Decision and Control, and the European Control Conference*, pages 814–819, Dec. 12-15 2005.

[28] T. Kailath. *Linear systems*, volume 1. Prentice-Hall Englewood Cliffs, NJ, 1980.

[29] G. Lafferriere, G. J. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals and Systems*, 13(1):1–21, 2000.

[30] E. Le Corronc, A. Girard, and G. Gössler. Mode sequences as symbolic states in abstractions of incrementally stable switched systems. In *52nd IEEE Conference on Conference on Decision and Control-2013*, 2013.

[31] libFAUDES. Software library for discrete event systems., 2006-2013. URL `http://www.rt.eei.uni-erlangen.de/FGdes/faudes/`.

[32] J. Lunze. Qualitative modelling of linear dynamical systems with quantized state measurements. *Automatica*, 30(3):417 – 431, 1994.

[33] T. Masopust. A note on controllability of deterministic context-free systems. *Automatica*, 48(8):1934–1937, 2012.

[34] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989. ISBN 0-13-115007-3.

[35] T. Moor. Approximationsbasierter Entwurf diskreter Steuerungen für gemischtwertige Regelstrecken. In *Forschungsberichte aus dem Max-Planck-Institut für Dynamik komplexer technischer Systeme*, volume 2. Shaker Verlag, 1999. Dissertation.

[36] T. Moor and J. Raisch. Supervisory control of hybrid systems within a behavioural framework. *Systems and Control Letters*, 38:157–166, 1999.

[37] T. Moor and J. Raisch. Abstraction based supervisory controller synthesis for high order monotone continuous systems. In *Modelling, Analysis, and Design of Hybrid Systems*, volume 279 of *Lecture Notes in Control and Information Sciences*, pages 247–265. Springer Berlin Heidelberg, 2002.

[38] T. Moor, J. Raisch, and S. O'Young. Discrete supervisory control of hybrid systems based on l-complete approximations. *Discrete Event Dynamic Systems*, 12(1):83–107, 2002.

[39] T. Moor, K. Schmidt, and T. Wittmann. Abstraction-based control for not necessarily closed behaviours. In *Proceedings of the 18th IFAC World Congress*, pages 6988–6993, 2011.

[40] T. Moor, C. Baier, T.-S. Yoo, F. Lin, and S. Lafortune. On the computation of supremal sublanguages relevant to supervisory control. In *Proceedings of the 11th International Workshop on Diskrete Event Systems*, Mexico, October 2012.

[41] G. J. Pappas. Bisimilar linear systems. *Automatica*, 39:2035–2047, 2003.

[42] L. Paulson, T. Nipkow, and M. Wenzel. Isabelle/HOL, 2011. URL http://isabelle.in.tum.de.

[43] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Universität Hamburg, 1962.

[44] G. Pola and A. J. van der Schaft. Equivalence of switching linear systems by bisimulation. *International Jurnal of Control*, 79:74–92, 2006.

[45] J. Raisch and S. O'Young. A totally ordered set of discrete abstractions for a given hybrid or continuous system. In *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*, pages 342–360. Springer Berlin Heidelberg, 1997.

[46] J. Raisch and S. O'Young. Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control*, 43(4):569–573, Apr 1998.

[47] J. Raisch, T. Moor, N. Bajcinca, S. Geist, and V. Nenchev. Distributed state estimation for hybrid and discrete event systems using l-complete approximations. In *Proceedings of the 10th International Workshop on Discrete Event Systems*, volume 10, pages 129–134, 2010.

[48] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. In *Analysis and Optimization of Systems*, volume 63 of *Lecture Notes in Control and Information Sciences*, pages 475–498. Springer Berlin Heidelberg, 1984. ISBN 978-3-540-13552-4.

[49] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98, 1989.

[50] G. Reißig. Computing abstractions of nonlinear systems. *IEEE Transactions on Automatic Control*, 56(11):2583–2598, 2011.

[51] G. Reißig and M. Rungger. Feedback refinement relations for symbolic controller synthesis. *53rd IEEE Conference on Decision and Control, Los Angeles*, 2014.

[52] A.-K. Schmuck and J. Raisch. Constructing (bi)similar finite state abstractions using asynchronous l-complete approximations. In *Proc. 53rd Annual Conference on Decision and Control*, 2014.

[53] A.-K. Schmuck and J. Raisch. Asynchronous l-complete approximations. *Systems & Control Letters*, 73(0):67 – 75, 2014.

[54] A.-K. Schmuck and J. Raisch. Simulation and bisimulation over multiple time scales in a behavioral setting. In *Proc. 22nd Mediterranean Conf. on Control and Automation, Palermo, Italy*, 2014.

[55] A.-K. Schmuck, S. Schneider, J. Raisch, and U. Nestmann. Extending supervisory controller synthesis to deterministic pushdown automata—enforcing controllability least restrictively. In *Proceedings of the 12th IFAC - IEEE International Workshop on Discrete Event Systems*, pages 286–293, 2014.

[56] A.-K. Schmuck, S. Schneider, J. Raisch, and U. Nestmann. Supervisory control synthesis for deterministic context free specification languages – enforcing controllability least restrictively. *Discrete Event Dynamic Systems*, 2015. (to appear).

[57] A.-K. Schmuck, P. Tabuada, and J. Raisch. Comparing asynchronous l-complete approximations and quotient based abstractions. *Proc. 54rd Annual Conference on Decision and Control*, 2015. (to appear).

[58] A.-K. Schmuck, P. Tabuada, and J. Raisch. Comparing asynchronous l-complete approximations and quotient based abstractions. *IEEE Transactions on Automatic Control*, 2015. (under review, preprint available at http://arxiv.org/abs/1503.07139).

[59] S. Schneider. Behavioral optimizations for deterministic pushdown automata. In *(submitted for publication)*, 2014.

[60] S. Schneider and U. Nestmann. Enforcing operational properties including blockfreeness for deterministic pushdown automata. 2014. http://arxiv.org/abs/1403.5081.

[61] S. Schneider, A.-K. Schmuck, J. Raisch, and U. Nestmann. Reducing an operational supervisory control problem by decomposition for deterministic pushdown automata. In *Proceedings of the 12th IFAC - IEEE International Workshop on Discrete Event Systems*, pages 214–221, 2014.

[62] R. S. Sreenivas. On a weaker notion of controllability of a language k with respect to a language l. *Automatic Control, IEEE Transactions on*, 38(9):1446–1447, 1993.

[63] P. Tabuada. Symbolic control of linear systems based on symbolic subsystems. *IEEE Transactions on Automatic Control, Special issue on Symbolic Methods for Complex Control Systems.*, 51(6):1003–1013, June 2006.

[64] P. Tabuada. Symbolic models for control systems. *Acta Informatica Special Issue Hybrid Systems and Control Letters*, 43(7):477–500, Feb. 2007.

[65] P. Tabuada. An approximate simulation approach to symbolic control. *IEEE Transactions on Automatic Control*, 53(6):1406–1418, 2008.

[66] P. Tabuada. Controller synthesis for bisimulation equivalence. *Systems and Control Letters*, 57:443–452, 2008.

[67] P. Tabuada. *Verification and Control of Hybrid Systems - A Symbolic Approach*, volume 1. Springer, 2009.

[68] P. Tabuada and G. J. Pappas. From discrete specifications to hybrid control. In *Proceedings. 42nd IEEE Conference on Decision and Control*, volume 4, pages 3366–3371. IEEE, 2003.

[69] P. Tabuada and G. J. Pappas. Finite bisimulations of controllable linear systems. In *Proceedings. 42nd IEEE Conference on Decision and Control, 2003.*, volume 1, pages 634–639 Vol.1, Dec 2003.

[70] P. Tabuada and G. J. Pappas. Bisimilar control affine systems. *Systems & Control Letters*, 52(1):49 – 58, 2004.

[71] D. C. Tarraf and L. A. D. Espinosa. On finite memory approximations constructed from input/output snapshots. In *50th IEEE Conference on Decision and Control*, pages 3966–3973, 2011.

[72] D.C. Tarraf, A. Megretski, and M. A. Dahleh. Finite approximations of switched homogeneous systems for controller synthesis. *Automatic Control, IEEE Transactions on*, 56(5):1140–1145, May 2011.

[73] J. Thistle and W. M. Wonham. Control of infinite behavior of finite automata. *SIAM Journal on Control and Optimization*, 32(4):1075–1097, 1994.

[74] J. Thistle and W. M. Wonham. Supervision of infinite behavior of discrete-event systems. *SIAM Journal on Control and Optimization*, 32(4):1098–1113, 1994.

[75] A. J. van der Schaft. Equivalence of dynamical systems by bisimulation. *IEEE Transactions on Automatic Control*, 49(12):2160–2172, 2004.

[76] J. C. Willems. Models for dynamics. *Dynamics Reported*, 2:172–269, 1989.

[77] J. C. Willems. Paradigms and puzzles in the theory of dynamic systems. *IEEE Transactions on Automatic Control*, 36(3):258–294, 1991.

[78] W. M. Wonham and P. J. Ramadge. On the supremal controllable sublanguage of a given language. In *SIAM Journal on Control and Optimization*, volume 25, pages 637–659, 1987.

[79] M. Zamani, G. Pola, M. Mazo, and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *Automatic Control, IEEE Transactions on*, 57(7):1804–1809, 2012.

[80] M. Zamani, I. Tkachev, and A. Abate. Bisimilar symbolic models for stochastic control systems without state-space discretization. In *Proc. 17th Conf. on Hybrid Systems: Computation and Control*, pages 41–50, 2014. ISBN 978-1-4503-2732-9.