Science of Computer Programming 167 (2018) 1–24

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Checking multi-view consistency of discrete systems with respect to periodic sampling abstractions ☆

Maria Pittou [a,*], Panagiotis Manolios [b,*], Jan Reineke [c], Stavros Tripakis [a,*]

[a] Aalto University, Finland
[b] Northeastern University, United States of America
[c] Saarland University, Germany

ARTICLE INFO

Article history:
Received 25 June 2017
Received in revised form 15 June 2018
Accepted 4 July 2018
Available online 17 July 2018

Keywords:
Multi-view modeling
View consistency
Formal methods
Modeling
Verification

ABSTRACT

In multi-view modeling (MVM) the system under development is described by distinct models, called views, which capture different perspectives of the system. Possible overlaps of the views may give rise to inconsistencies. Following the formal MVM framework of [33], the *view consistency problem* asks to check the consistency of a given set of views with respect to a given set of abstraction functions. Existing work checks view consistency of discrete systems (transition systems or finite automata) with respect to two types of abstraction functions: (1) projections of state variables and (2) projections of an alphabet of events onto a subalphabet. In this paper, we study view consistency with respect to *timing* abstractions, specifically, *periodic sampling*, for automata and transition systems.

1. Introduction

Designing complex systems, such as distributed, embedded, or cyber-physical systems, is a challenging task. As many of these systems are safety-critical, design by trial-and-error is not a viable option, and more rigorous methods such as model-based design are preferred (see [38] for an overview). Model-based design often involves several experts and stakeholders, or teams thereof, each having their own perspective, or *view*, of the system. These views are typically captured concretely using different kinds of models [5,39,26]. These models cover different and potentially overlapping aspects of the system. In such a *multi-view modeling* setting, a basic problem is to check that the views are *consistent*, *i.e.*, that they do not contradict each other [34,26]. A formal framework that allows one to reason about multi-view modeling in general, and provides a mathematical definition of view consistency, has been proposed in [33]. In this paper we extend this formal framework along multiple directions, as explained below.

In the MVM framework of [33], *systems* and *views* are formalized as sets of behaviors. View behaviors are obtained by applying some kind of *abstraction functions* to system behaviors. In a nutshell, the view consistency problem can be formulated as follows. Given a set of views $\mathcal{V}_1, \dots, \mathcal{V}_n$, and corresponding abstraction functions $\alpha_1, \dots, \alpha_n$, does there exist a system $\mathcal{S}$ such that $\mathcal{V}_i = \alpha_i(\mathcal{S})$ for all $i = 1, \dots, n$? If such a *witness system* $\mathcal{S}$ exists, then the views $\mathcal{V}_1, \dots, \mathcal{V}_n$ are deemed consistent, otherwise they are deemed inconsistent.[1]

☆ This work was partially supported by the Academy of Finland and the U.S. National Science Foundation (awards #1329759, #1139138, and #1319580). This paper is an extended and modified version of conference paper [27].

* Corresponding authors.
E-mail address: mpittou5@gmail.com (M. Pittou).

[1] The framework of [33] allows for a more flexible definition of consistency where conformance of the view $\mathcal{V}_i$ to an abstracted system $\alpha_i(\mathcal{S})$ need not be equality. In this paper we use equality as the conformance relation.

https://doi.org/10.1016/j.scico.2018.07.003
0167-6423/© 2018 Elsevier B.V. All rights reserved.

The formal framework of [33] is abstract, in the sense that it does not *a priori* prescribe how to specify behaviors, systems, or views; neither does it specify what the abstraction functions are. This abstract framework has been instantiated in previous work [33,32,28,27] along several dimensions:

- In [33], an instantiation of the abstract MVM framework is considered where systems and views are specified as symbolic transition systems (STSs). The view consistency problem is studied w.r.t. abstraction functions which are projections of state variables.
- [32] extends the results of [33], by considering an instantiation of the abstract MVM framework where systems and views are specified as automata (over finite or infinite words) and where abstraction functions are projections of the alphabet of events onto a subalphabet.
- [28] further extends some of the results of [32] by studying the view consistency problem for the case of *infinitary* languages, *i.e.*, languages containing both finite and infinite words, and their corresponding representation as pairs of automata over finite words and ω-automata.
- In all three papers [33,32,28], the abstraction functions considered are projections, either of state variables, or of events in the alphabet. [27] considers a different kind of abstraction functions, namely, *periodic sampling*. This paper is the journal version of [27] and extends its results along multiple dimensions, while also providing new results to the general abstract multi-view modeling framework of [33].

Sampling is a widely used mechanism in observation, control, embedded software, and other settings, and therefore is a natural choice to consider when studying different kinds of abstractions for MVM. There are different types of sampling typically used in control, such as *time-driven/periodic*, or *event-driven*. In the former, the sampling points are determined based on time, whereas in the latter, they are determined based on some event (*e.g.*, some type of state change). In this paper, we consider only time-driven and in fact periodic sampling. The study of other types of sampling abstractions is left as future work. In periodic sampling the system is sampled at periodic intervals, *i.e.*, at times $0, T, 2T$, and so on, where T is the period.

For instance, consider a drive-by-wire system in a modern car. In the final system running in the car, there may be several sensors placed at different locations and monitoring different parts of the car, often by periodically sampling some physical values (*e.g.*, temperature, pressure, etc.). There may also be several control programs periodically executing, reading inputs from the sensors (or receiving them from other programs), performing computations, and writing their outputs to actuators (or sending them to other programs). The programs may communicate over networks which are themselves time-triggered, and often periodic. The periods in the system need not all be the same. Such *multi-periodic* systems are very common in the domain of embedded systems (*e.g.*, see [7]) and naturally give rise to multi-periodic sampling abstractions. Although *at runtime* the consistency of the different periodically-sampled views of the system is ensured (since by definition the witness system under execution exists), this is not necessarily the case *at design time*. This is where the MVM framework can be of use. In particular, when different design teams each employ a model of the system sampled at a different period, the consistency of those models/views needs to be ensured.

As a concrete example, consider two views of the system, denoted V_2 and V_5. V_2 is supposed to model the behavior of the system sampled with period 2, while V_5 models the behavior of the system sampled with period 5. Suppose that both V_2 and V_5 assert that the system contains a certain boolean variable b, which is initially 0 and then alternates between 0 and 1 at every transition. Although it may at first appear that V_2 and V_5 are consistent with each other, they are not. The reason is that V_2 asserts that b will be 0 at times $0, 4, 8, 12$, and so on, and 1 at times $2, 6, 10$, and so on. On the other hand, V_5 asserts that b will be 0 at times $0, 10, 20$, and so on, and 1 at times $5, 15$, and so on. At time 10, there is an inconsistency, as V_2 asserts b to be 1 while V_5 asserts b to be 0. Discovering such inconsistencies automatically (not only for simple cases like this one, but for more complex and general models) can be achieved using the models and algorithms provided in this paper.

Sometimes the sampling does not start at time 0 but at some other time called the *initial phase*. For simplicity, in this paper we consider that the initial phase is 0. However, our results should be easy to extend to the general case (an example of how this could be done is provided in Section 5.2).

This paper significantly extends the results of [27] on the study of MVM under multi-periodic sampling abstractions. Specifically, the contributions of this paper are the following:

(1) We introduce the concept of *canonical witness system candidate* (see Section 4.3), namely, $\mathcal{S}^{\#} = \bigcap_{i=1}^{n} \alpha_i^{-1}(\mathcal{V}_i)$, where α_i^{-1} is the inverse of abstraction function α_i. $\mathcal{S}^{\#}$ is important as it provides a necessary and sufficient condition for checking view consistency (Theorem 5). In particular, views $\mathcal{V}_1, \ldots, \mathcal{V}_n$ are consistent iff $\mathcal{S}^{\#}$ is a valid witness, *i.e.*, iff for all $i = 1, \ldots, n$, $\alpha_i(\mathcal{S}^{\#}) = \mathcal{V}_i$. The concept of canonical witness system candidate extends the abstract multi-view modeling framework of [33], as it is a general result which holds independently of the particular instantiation of the framework. The concept can be used in any instance of the framework, and is indeed used to derive algorithms for the several instances of the view consistency problem studied in this paper.

(2) We introduce and study the multi-view consistency problem for views represented as non-deterministic Büchi automata (NBA) and periodic sampling abstraction functions on infinite words. Such a periodic sampling function takes an infinite word w and returns a new infinite word by taking one out of every T letters in w, where T is the given period. We solve the multi-view consistency problem in this setting by applying the necessary and sufficient condition from contri-

bution (1) above. To this end, we develop methods to compute the canonical witness system candidate in the NBA setting. This involves being able to compute forward and inverse periodic samplings of ω-regular languages as represented by their corresponding NBA. We provide such methods, which proves closure of ω-regular languages with respect to forward and inverse periodic sampling. In the process, we need to consider NBA with ϵ-transitions, and we provide a method to eliminate such transitions while preserving the language of the NBA. All these results extend the results of [27]. Although [27] considers periodic sampling abstraction functions, the setting there is that of STSs, and not that of ω-regular languages and NBA, which are novel contributions of this work. STSs lack acceptance conditions and hence cannot capture liveness properties. Büchi automata can model liveness. On the other hand, Büchi automata are not symbolic. Therefore, both models are useful to consider. We also note that symbolic transition systems and Büchi automata are both standard formalisms used by widespread verification tools such as SMV [25] or Spin [14].

(3) We re-consider the setting of STSs and periodic sampling abstraction functions of [27]. The problem of checking view consistency in that setting was already studied in [27], but only a sound and incomplete algorithm was provided there. In this paper, we provide a novel, sound and complete method to solve the problem. The method applies the generic necessary and sufficient condition from (1) and relies on computing the canonical witness system candidate in the STS setting. For this computation, closure properties of STSs with respect to forward and inverse periodic sampling abstraction functions are studied. As in [27], STSs with observable and unobservable variables are considered, and both the classes of FOSs (fully-observable systems) and nFOSs (non-fully-observable systems) are considered and their closure w.r.t. forward and inverse periodic sampling is examined. We show that both FOSs and nFOSs are closed under forward periodic sampling, but only nFOSs are generally closed under inverse periodic sampling.

(4) We also answer another question left open in [27]. There, three variants of the view consistency problem were proposed. Using the numbering in this paper: Problem 3, which simply asks for a semantic witness system (*i.e.*, a set of behaviors); Problem 4, which asks for a witness system that can be represented as an nFOS; and Problem 5, which asks for a witness system that can be represented as a FOS. While it is clear that existence of a FOS witness implies existence of an nFOS, and that the latter implies existence of a semantic witness, the reverse directions are not obvious. The non-equivalence of Problems 4 and 5 was shown in [27] but the question whether Problems 3 and 4 are equivalent was left open. In this paper, we show that Problems 3 and 4 are equivalent, that is, the existence of a semantic witness implies the existence of an nFOS witness.

(5) We have added several new examples throughout the paper, plus two complete examples illustrating the process of checking view consistency in the NBA and the STS settings in Sections 5.5 and 6.5, respectively.

2. Related work

Multi-view modeling is a well-known concept in the software and system engineering communities, (*cf.*, standards such as ISO 42010 [16]). Existing work mainly focuses on designing architectures that combine various modeling tools or elements of these tools [5,12]. Then, checking for multi-view consistency consists in checking the consistency of the different architectures. Consistency problems on the behavioral aspects of a system have been discussed widely within the context of multi-modeling languages such as UML [31,41,9,22,23] and SysML [34]. Architectural views and structural consistency notions are also studied in [4,3,24]. A static, logic-based setting for consistency is also studied in [11].

The above work focuses on architectural, structural, and generally *static* notions of views. In this work we consider only behavioral aspects of systems, and in that sense, our approach focuses on *dynamics*. Dynamical, behavioral views are also studied in [29,30] within the context of cyber-physical systems. The aim there is mainly aiding the verification process, rather than checking view consistency. [31] follows a “hybrid” approach, where model-checking techniques are used to check consistency between class diagrams and state machines.

In our framework, view consistency problems admit a yes/no answer. This may be too strict in some contexts, and indeed, some work follows a more lenient approach which may leave room for inconsistencies [10], or opt for more light-weight detection and tracking of ontological overlaps [36].

Other approaches to multi-view modeling include interface theories [8,13], specification and abstraction/refinement theories [17,6], and contract theories [2]. In aspect-oriented modeling [20], aspects are used instead of views in order to describe tangled and scattered behaviors across a system. Similarly to multi-view modeling, the main concern is again to identify conflicts among the multiple aspects. Several multi-view modeling techniques encompass metamodeling methods [19,18] in order to transform the views into a common metamodel and capture their dependencies. Extensive surveys of multi-view modeling approaches can be found in [1,26].

Our work follows the setting introduced in [33] and further investigated in [32,28]. However, the abstraction functions studied in this paper are different from those considered in the papers above, as stated in the introduction. This paper is a modified version of conference paper [27]. Compared to [27] and previous publications [33,32], new material in this paper includes: Section 4.3 on the canonical witness system candidate, with the necessary and sufficient condition for view consistency Theorem 5; Section 5 on the instantiation of the multi-view modeling framework to the case of ω-regular languages represented as Büchi automata, and periodic sampling abstraction functions; Section 6.4 on solving view consistency problems in the case of STSs using new sound and complete methods (the sound but incomplete algorithm proposed in [27] has been omitted from this paper); extensive revisions compared to [27] in the constructions of forward and inverse periodic samplings of STSs (Sections 6.2 and 6.3); Section 3.1.4 on eliminating ϵ-transitions from Büchi automata; Section 3.2.4 on transforming nFOSs to Büchi automata; and two complete examples in Sections 5.5 and 6.5.

3. Background: automata and transition systems

Sets and functions Let S denote an arbitrary finite set: $|S|$ denotes its cardinality. For any set S, we use $\mathcal{P}(S)$ or 2^S to denote its powerset. Also: $\mathbb{B} = \{0, 1\}$ is the set of booleans, $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers and $\mathbb{N}_{>0} = \{1, 2, \ldots\}$. The notation $f : A \to B$ is used to denote a function f from set A to set B.

Alphabets, finite words, infinite words, ϵ-elimination A finite alphabet Σ is a non-empty finite set of symbols. As usual, ϵ denotes the empty word, Σ^* is the set of all finite words over Σ, and Σ^ω is the set of all infinite words over Σ. We denote a finite (resp. infinite) word over some alphabet Σ by $w = a_0 \cdots a_{n-1}$ (resp. $w = a_0 a_1 \cdots$), where $a_i \in \Sigma$ for all $0 \le i \le n-1$ (resp. for all $i \ge 0$). Consider, for example, the alphabet $\Sigma = \{a, b, c\}$. Then $cb, aabbcc \in \Sigma^*$, and $abbb \cdots = ab^\omega, abcabc \cdots = (abc)^\omega \in \Sigma^\omega$, where u^ω denotes the infinite repetition of finite word u. It is sometimes convenient to consider ϵ to be a special symbol, not in Σ. Then, given a word w over $\Sigma \cup \{\epsilon\}$, the *ϵ-elimination of w* is the new word w' obtained by removing from w all ϵ symbols, so that w' is a word over Σ. For example, if $w = a\epsilon\epsilon bc\epsilon$ then its ϵ-elimination is $w' = abc$.

Languages A $*$-language (star language) L on Σ is a set of finite words, subset of Σ^*, *i.e.*, $L \subseteq \Sigma^*$, and an ω-language (omega language) L on Σ is a set of infinite words, subset of Σ^ω, *i.e.*, $L \subseteq \Sigma^\omega$. For example, consider the alphabet $\Sigma = \{a, b\}$. Then, $L_1 = a^* b^* \subseteq \Sigma^*$ is a star language and $L_2 = a^*(ba)^\omega \subseteq \Sigma^\omega$ is an omega language, where u^* denotes a finite number of zero or more repetitions of finite word u.

Complexity classes We use *P*, *NP*, *PSPACE*, etc. for the standard complexity classes [21].

3.1. Automata

3.1.1. Nondeterministic Büchi automata

A nondeterministic Büchi automaton (NBA for short) over a finite alphabet Σ, is a tuple $A = (Q, \Sigma, Q_0, \Delta, F)$, where Q is the finite set of states, Σ is the alphabet, $Q_0 \subseteq Q$ is the set of initial states, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition function, and $F \subseteq Q$ is the set of final states. When $(q, a, q') \in \Delta$, we write $q \xrightarrow{a} q'$. A run P_w of A over an infinite word $w = a_0 a_1 \cdots \in \Sigma^\omega$ is an infinite sequence $P_w : (q_0, a_0, q_1)(q_1, a_1, q_2) \cdots$ such that $q_0 \in Q_0$ is an initial state and $(q_i, a_i, q_{i+1}) \in \Delta$ for every $i \ge 0$. For every run P_w of A over an infinite word $w \in \Sigma^\omega$ we denote with $\mathit{Inf}(P_w)$ the set of states occurring an infinite number of times along P_w. Then, a run P_w of A over $w \in \Sigma^\omega$ is called accepting if $\mathit{Inf}(P_w) \cap F \neq \emptyset$. An infinite word $w \in \Sigma^\omega$ is accepted by A if there is an accepting run P_w of A over w. The language accepted by A, written $L(A)$, is the omega language containing the set of all infinite words accepted by A: $L(A) = \{w \in \Sigma^\omega \mid \exists$ infinite accepting run P_w of A over $w\}$. A language L is called ω-regular if there exists a Büchi automaton A over Σ accepting L, *i.e.*, $L = L(A)$.

3.1.2. Closure properties of NBA under union and intersection

Next, we recall the closure of NBA under union and intersection. The latter result is necessary for the application of the theory presented in Section 5.

Lemma 1. *[37] Let $A_1 = (Q_1, \Sigma, Q_{0_1}, \Delta_1, F_1)$ and $A_2 = (Q_2, \Sigma, Q_{0_2}, \Delta_2, F_2)$ be two NBA. Let $A_1 \oplus A_2 = (Q_1 \cup Q_2, \Sigma, Q_{0_1} \cup Q_{0_2}, \Delta, F_1 \cup F_2)$ where $\Delta = \{(q, \sigma, q') \mid (q, \sigma, q') \in \Delta_1$ or $(q, \sigma, q') \in \Delta_2\}$. Then, $L(A_1 \oplus A_2) = L(A_1) \cup L(A_2)$.*

Lemma 2. *[37] Let $A_1 = (Q_1, \Sigma, Q_{0_1}, \Delta_1, F_1)$ and $A_2 = (Q_2, \Sigma, Q_{0_2}, \Delta_2, F_2)$ be two NBA. Let $A_1 \otimes A_2 = (Q_1 \times Q_2 \times \{1, 2\}, \Sigma, Q_{0_1} \times Q_{0_2} \times \{1\}, \Delta, Q_1 \times F_2 \times \{2\})$ where $\Delta = \{((q_1, q_2, 1), \sigma, (q'_1, q'_2, j)) \mid (q_1, \sigma, q'_1) \in \Delta_1$ and $(q_2, \sigma, q'_2) \in \Delta_2$ and if $q_1 \in F_1$ then $j = 2$ else $j = 1\} \cup \{((q_1, q_2, 2), \sigma, (q'_1, q'_2, j)) \mid (q_1, \sigma, q'_1) \in \Delta_1$ and $(q_2, \sigma, q'_2) \in \Delta_2$ and if $q_2 \in F_2$ then $j = 1$, else $j = 2\}$. Then, $L(A_1 \otimes A_2) = L(A_1) \cap L(A_2)$.*

3.1.3. Nondeterministic Büchi automata with epsilon transitions

In Section 5 we study problems that necessitate both the use of a variant of NBA with "silent" or ϵ-transitions and the elimination of such transitions with the aim of obtaining a standard NBA accepting an equivalent language. Although the elimination of ϵ-transitions is standard for finite automata over finite words [15], we have not managed to find a published procedure to eliminate ϵ-transitions in NBA. Therefore we propose such a procedure here.

An NBA with ϵ-transitions over alphabet Σ is an NBA over $\Sigma \cup \{\epsilon\}$, where ϵ is the empty-word symbol, assumed to be not in Σ. That is, an NBA with ϵ-transitions is a tuple $A = (Q, \Sigma \cup \{\epsilon\}, Q_0, \Delta, F)$, where $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$, with $\epsilon \notin \Sigma$. A can have transitions of two kinds: either of the form (q, a, q'), where $a \in \Sigma$, or (q, ϵ, q'). The latter is an ϵ-transition. Such transitions do not "add letters" to the word accepted by the automaton. Formally, consider an infinite word $w = x_0 x_1 \cdots$ over $\Sigma \cup \{\epsilon\}$. That is, every x_i can be either a symbol in Σ, or ϵ. A run P_w of A over w is an infinite sequence of transitions

$(q_0, x_0, q_1)(q_1, x_1, q_2)\cdots$ such that $q_0 \in Q_0$ and $(q_i, x_i, q_{i+1}) \in \Delta$ for every $i \geq 0$. P_w is accepting if it visits a state in F infinitely many times. A word $w' \in \Sigma^\omega$ is accepted by A if w' is the ϵ-elimination of some word $w \in (\Sigma \cup \{\epsilon\})^\omega$ and A has an accepting run over w. The language of A, $L(A) \subseteq \Sigma^\omega$, is the omega language containing the set of all infinite words over Σ that are accepted by A.

In the sequel, we use the notation $\xrightarrow{\epsilon^*}$ for the reflexive transitive closure of $\xrightarrow{\epsilon}$, *i.e.*, of ϵ-transitions. Namely, $q \xrightarrow{\epsilon^*} q'$ iff $q = q'$ or Δ contains a set of transitions $(q, \epsilon, q_1), (q_1, \epsilon, q_2), \ldots, (q_k, \epsilon, q')$, for some $k \geq 0$.

In what follows, when we say Büchi automaton or NBA we implicitly mean a Büchi automaton *without* ϵ-transitions. When we want to say that the automaton may have ϵ-transitions, we state that explicitly.

3.1.4. Eliminating epsilon transitions from NBA

Given an NBA with ϵ-transitions A, we want to produce a standard NBA A' (without ϵ-transitions) such that $L(A) = L(A')$. Let $A = (Q, \Sigma \cup \{\epsilon\}, Q_0, \Delta, F)$. Define $A' = (Q, \Sigma, Q_0, \Delta', F)$, where: $\Delta' = \{(q, a, q') \mid a \in \Sigma \wedge \exists q_1, q_2 \in Q : q \xrightarrow{\epsilon^*} q_1 \xrightarrow{a} q_2 \xrightarrow{\epsilon^*} q'\}$.

Theorem 1. $L(A) = L(A')$.

Proof. Let $w \in L(A')$ and consider an accepting run $P'_w = (q_0, a_0, q_1)(q_1, a_1, q_2)\cdots$ of A' over w. By construction of Δ', for every $i \geq 0$, there exist some states $\bar{q_i}$ and $\bar{q_i}'$ such that $q_i \xrightarrow{\epsilon^*} \bar{q_i}$, $\bar{q_i} \xrightarrow{a_i} \bar{q_i}'$, and $\bar{q_i}' \xrightarrow{\epsilon^*} q_{i+1}$ are transitions and runs of Δ. We construct a run P_w of A over w, such that for every $i \geq 0$, the transition (q_i, a_i, q_{i+1}) of P'_w is replaced with the finite subrun $q_i \xrightarrow{\epsilon^*} \bar{q_i} \xrightarrow{a_i} \bar{q_i}' \xrightarrow{\epsilon^*} q_{i+1}$. Every state visited infinitely often in P'_w is also visited infinitely often in P_w. Specifically, every accepting state visited infinitely often in P'_w is also visited infinitely often in P_w. Therefore, and since P'_w is an accepting run of A', P_w is an accepting run of A over w. Hence, $w \in L(A)$.

Conversely, let $w \in L(A)$. Note that w is by definition an infinite word in Σ^ω. Let $w = a_0 a_1 \cdots$, with $a_i \in \Sigma$ for all $i \geq 0$. Consider an accepting run P_w of A over w. P_w must be comprised of segments of the form

$$P_w = q_0 \xrightarrow{\epsilon^*} \bar{q}_0 \xrightarrow{a_0} q_1 \xrightarrow{\epsilon^*} \bar{q}_1 \xrightarrow{a_1} q_2 \cdots q_i \xrightarrow{\epsilon^*} \bar{q_i} \xrightarrow{a_i} q_{i+1} \cdots .$$

Let Q_i be the set of states appearing in $q_i \xrightarrow{\epsilon^*} \bar{q_i}$ and notice that

$$P'_w = q_0 \xrightarrow{a_0} q'_1 \xrightarrow{a_1} q'_2 \cdots q'_i \xrightarrow{a_i} q'_{i+1} \cdots$$

is a run of A' over w, when $q'_i \in Q_i$ for all $i > 0$, no matter how the q'_i are chosen, because:

$$\text{for all } i \geq 0, q'_i \xrightarrow{\epsilon^*} \bar{q_i} \xrightarrow{a_i} q_{i+1} \xrightarrow{\epsilon^*} q'_{i+1} \text{ in } \Delta, \text{ so } q'_i \xrightarrow{a_i} q'_{i+1} \text{ in } \Delta' \text{ (where } q'_0 = q_0\text{)}.$$

Since P_w is accepting, there exists some final state, say q, that appears infinitely often in P_w, *i.e.*, q is an element of Q_i for infinitely many i. Hence, P'_w is also an accepting run of A' over w, if we define q'_i, for $i > 0$, as follows: $q'_i = q$ when $q \in Q_i$ and $q'_i = q_i$ otherwise. Since q appears in infinitely many Q_i, it appears infinitely often in P'_w and we established above that P'_w is a run of A' over w, so it is also an accepting run, *i.e.*, $w \in L(A')$. □

Theorem 2. *Given an NBA with ϵ-transitions $A = (Q, \Sigma \cup \{\epsilon\}, Q_0, \Delta, F)$ we can construct an equivalent NBA A' without ϵ-transitions in $O(|Q|^3 + |\Sigma| \cdot |Q|^2)$ time and $O(|\Sigma| \cdot |Q|^2)$ space. Thus, the problem of transforming an NBA with ϵ-transitions into an equivalent NBA without ϵ-transitions is in P.*

Proof. Let $A' = (Q, \Sigma, Q_0, \Delta', F)$ where $\Delta' = \{(q, a, q') \mid a \in \Sigma \wedge \exists q_1, q_2 \in Q : q \xrightarrow{\epsilon^*} q_1 \xrightarrow{a} q_2 \xrightarrow{\epsilon^*} q'\}$. The transitive closure of epsilon transitions in A can be determined by Warshall's algorithm [40], which has run time complexity $O(|Q|^3)$. Then, constructing the NBA A' takes at most $|\Sigma| \cdot |Q|^2$ steps. Therefore, the construction of A' requires $O(|Q|^3 + |\Sigma| \cdot |Q|^2)$ time. Since A' has the same set of states Q as the original NBA A, we have that A' has at most $|\Sigma| \cdot |Q|^2$ transitions, *i.e.*, requires $O(|\Sigma| \cdot |Q|^2)$ space. □

Example 1. Consider the NBA A with ϵ-transitions over $\Sigma = \{b, c\}$, shown in the left part of Fig. 1. Eliminating ϵ-transitions from A results in the equivalent NBA A' shown in the right part of Fig. 1.

3.2. Symbolic transition systems

We consider finite state transition systems described symbolically, as in [33]. The state space of a transition system is described by a (finite) set of boolean variables X, resulting in 2^n states where $n = |X|$, and a *state* s over X is a function $s : X \to \mathbb{B}$. A *behavior over* X is an infinite sequence of states over X, $\sigma = s_0 s_1 \cdots$, where s_i is the state at position i. We

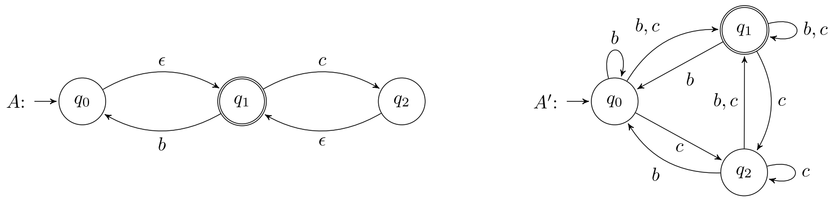

Fig. 1. Example of NBA epsilon removal. NBA A over $\Sigma = \{b, c\} \cup \{\epsilon\}$ and equivalent NBA A' without ϵ-transitions.

denote with $\mathcal{U}(X)$ the set of all possible behaviors over X. Semantically, a *transition system* $\mathcal{S}$ over the state space X is a set of behaviors over X, *i.e.*, $\mathcal{S} \subseteq \mathcal{U}(X)$.

To represent transition systems syntactically, we use a finite-state *symbolic* representation.[2] In the sequel we use the term *symbolic* transition system to distinguish the syntactic object defined below, from the semantic object (set of behaviors). Notation-wise, semantic transition systems are denoted by $\mathcal{S}$ and symbolic transition systems are denoted by S. Note that our transition systems have no notion of accepting conditions. Throughout this paper, we are interested only in the infinite behaviors generated by those systems (as these have good closure properties after periodic sampling), and hence behaviors that deadlock (*i.e.*, finite ones) are neglected.

As in [33,32], we consider symbolic transition systems of two kinds: first, *fully-observable* systems where all variables are observable; second, *non-fully-observable* systems which also have internal, unobservable variables. The latter are introduced because fully observable systems are not always closed under operations such as union [33]. As it turns out, fully-observable systems are also not generally closed under the various abstraction functions (or their inverses) considered in this paper.

3.2.1. Fully-observable symbolic transition systems

A *fully-observable* symbolic transition system (FOS for short) is a triple $S = (X, \theta, \phi)$ where X is the finite set of boolean variables, θ is a boolean expression over X characterizing the set of all initial states, and ϕ is a boolean expression over $X \cup X'$, where $X' = \{x' \mid x \in X\}$ is the set of *next state* variables. ϕ captures the transition relation: it characterizes pairs of states (s, s') representing a transition from s to s'. We write $\theta(s)$ to denote that s satisfies θ. We write $\phi(s, s')$ to denote that the pair (s, s') satisfies ϕ, *i.e.*, that there is a transition from s to s'.

A *behavior* of a FOS (X, θ, ϕ) is an infinite sequence of states over X, $\sigma = s_0 s_1 \cdots$, such that σ can be generated by the FOS, *i.e.*, such that $\theta(s_0)$ and $\forall i : \phi(s_i, s_{i+1})$. We denote by $[\![S]\!]$ the set of all behaviors of S.

3.2.2. Non-fully-observable symbolic transition systems

Fully-observable systems can be extended with a set of *internal, unobservable* state variables. Before describing how these are used, we introduce the notion of a hiding function.

Consider a state s over the set of variables X and a subset $Y \subseteq X$. The *hiding function* h_Y projects s onto the set of variables Y, hence h_Y hides from s all variables in $X \setminus Y$. Then $h_Y(s)$ is defined to be the new state s', that is, $s' : Y \to \mathbb{B}$ such that $s'(x) = s(x)$ for every $x \in Y$. We extend hiding to behaviors and systems in the standard way. If $\sigma = s_0 s_1 \cdots$ is a behavior over X, then $h_Y(\sigma)$ is a behavior over Y defined by $h_Y(\sigma) = h_Y(s_0) h_Y(s_1) \cdots$. If $\mathcal{S}$ is a transition system over X and $Y \subseteq X$, then $h_Y(\mathcal{S}) = \{h_Y(\sigma) \mid \sigma \in \mathcal{S}\}$.

Formally, a *non-fully-observable* symbolic transition system (nFOS for short) is a tuple $S = (X, Z, \theta, \phi)$ where X, Z are disjoint finite sets of variables such that X describes the set of observable variables, and Z the set of internal (unobservable) variables. The initial condition θ is a boolean expression over $X \cup Z$, and the transition relation ϕ is a boolean expression over $X \cup Z \cup X' \cup Z'$.

A behavior of an nFOS $S = (X, Z, \theta, \phi)$ is an infinite sequence of states over $X \cup Z$ which can be generated by S, in the same manner as with behaviors generated by a FOS. The *observable behavior* of a behavior σ over $X \cup Z$ is the behavior $h_X(\sigma)$ over X. In what follows we denote by $[\![S]\!]$ the set of all behaviors of S (over $X \cup Z$), and by $[\![S]\!]_o$ the set of its observable behaviors (over X). If S has no internal variables, *i.e.*, if $Z = \emptyset$, then S is a FOS and we have $[\![S]\!] = [\![S]\!]_o$.

3.2.3. Closure and non-closure properties of FOSs and nFOSs under union and intersection

In the sequel, we recall the closure and non-closure properties of FOSs and nFOSs with respect to the operations of union and intersection, studied in [33,32]. Some of these properties will be used in Section 6.

Lemma 3. *[33,32] Fully-observable systems over a set of variables X are not closed under union,* i.e., *there exist $S_1 = (X, \theta_1, \phi_1)$ and $S_2 = (X, \theta_2, \phi_2)$ such that there is no $S = (X, \theta, \phi)$ such that $[\![S]\!] = [\![S_1]\!] \cup [\![S_2]\!]$.*

Lemma 4. *[33,32] Let $S_1 = (X, \theta_1, \phi_1)$ and $S_2 = (X, \theta_2, \phi_2)$ be two FOSs and let $S_1 \otimes S_2 = (X, \theta_1 \wedge \theta_2, \phi_1 \wedge \phi_2)$. Then, $[\![S_1 \otimes S_2]\!] = [\![S_1]\!] \cap [\![S_2]\!]$.*

[2] Our syntactic representation is finite state, and therefore it cannot capture all possible semantic transition systems, since there are infinite sets that cannot be represented in a finite way.

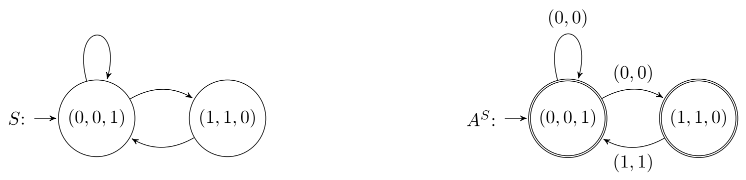

Fig. 2. Example nFOS-to-NBA transformation: from the nFOS S (left) to the NBA A^S (right).

Lemma 5. *[33,32] Let $S_1 = (X, Z_1, \theta_1, \phi_1)$ and $S_2 = (X, Z_2, \theta_2, \phi_2)$ be nFOSs, such that $Z_1 \cap Z_2 = \emptyset$. Let z be a fresh variable not in $X \cup Z_1 \cup Z_2$. Let: $S_1 \otimes S_2 = (X, Z_1 \cup Z_2, \theta_1 \wedge \theta_2, \phi_1 \wedge \phi_2)$ and $S_1 \oplus S_2 = (X, Z_1 \cup Z_2 \cup \{z\}, (\theta_1 \wedge z) \vee (\theta_2 \wedge \neg z), (z \to \phi_1 \wedge z') \wedge (\neg z \to \phi_2 \wedge \neg z'))$. Then, $[\![S_1 \otimes S_2]\!]_o = [\![S_1]\!]_o \cap [\![S_2]\!]_o$ and $[\![S_1 \oplus S_2]\!]_o = [\![S_1]\!]_o \cup [\![S_2]\!]_o$.*

3.2.4. Transformation of nFOSs to Büchi automata

As shown in Section 6.4, we can reduce view consistency problems for nFOSs to checking view consistency for Büchi automata. In order to achieve this, we need to be able to convert an nFOS to a Büchi automaton. In the sequel we show how we can obtain such a transformation from nFOSs to NBA.[3]

Consider an nFOS $S = (X, Z, \theta, \phi)$. We want to construct an NBA A^S, such that $[\![S]\!]_o = L(A^S)$. We define $A^S = (Q, \Sigma, Q_0, \Delta, F)$ such that Q is the set of functions from $X \cup Z$ to $\mathbb{B}$, Σ is the set of functions from X to $\mathbb{B}$, $Q_0 = \{q \in Q \mid \theta(q)\}$, $F = Q$, and the transition function $\Delta \subseteq Q \times \Sigma \times Q$ is defined by $\Delta = \{(q, a, q') \mid \phi(q, q') \text{ and } a = h_X(q)\}$. The idea of the transformation is that a transition (q, q') in S is mapped into a transition (q, a, q') in A^S where a is the observable part of the state q.

Theorem 3. *Let S and A^S be as above. Then: (1) The observable behaviors of S coincide with the language of A^S*, i.e., $[\![S]\!]_o = L(A^S)$. *(2) The problem of constructing NBA A^S from nFOS S such that $[\![S]\!]_o = L(A^S)$, is in EXPTIME.*

Proof. First we prove part (1), namely that $[\![S]\!]_o = L(A^S)$. Let $\sigma = s_0 s_1 \cdots \in [\![S]\!]_o$. Then there exists some behavior $\sigma' = s'_0 s'_1 \cdots \in [\![S]\!]$ such that $h_X(\sigma') = \sigma$. Moreover, $\theta(s'_0)$ and $\phi(s'_i, s'_{i+1})$ for $i \geq 0$, and by definition of A^S, we get that $s'_0 \in Q_0$ and $(s'_i, a_i, s'_{i+1}) \in \Delta$ where $a_i = h_X(s'_i)$, *i.e.*, $a_i = s_i$, for $i \geq 0$. Then, we define the path P_σ of A^S over σ by $P_\sigma = (s'_0, s_0, s'_1)(s'_1, s_1, s'_2) \cdots$, which is moreover accepting because by construction of A^S, $s'_i \in F$ for every $i \geq 0$. Hence, $\sigma \in L(A^S)$ and $[\![S]\!]_o \subseteq L(A^S)$.

Conversely, let $w = a_0 a_1 \cdots \in L(A^S)$ and consider the accepting path $P_w = (q_0, a_0, q_1)(q_1, a_1, q_2) \cdots$ of A^S over w, where $q_i \in F$ for every $i \geq 0$. By definition of A, the following conditions hold: (i) $\theta(q_0)$ because $q_0 \in Q_0$; (ii) $\phi(q_i, q_{i+1})$ and (iii) $h_X(q_i) = a_i$, because $(q_i, a_i, q_{i+1}) \in \Delta$ for $i \geq 0$. Then, by (i) and (ii) we get $q_0 q_1 \cdots \in [\![S]\!]$. Therefore, $h_X(q_0 q_1 \cdots) \in [\![S]\!]_o$, which by (iii) implies that $w = a_0 a_1 \cdots \in [\![S]\!]_o$, and we are done.

Next, we consider part (2), the complexity analysis. Let $n = |X| + |Z|$. Let $A^S = (Q, \Sigma, Q_0, \Delta, F)$ be defined according to the construction before Theorem 3. Then, we have that $|Q| = 2^{|X \cup Z|} = 2^{m+n}$. Also, the number of transitions in the worst case is $|\Delta| = |Q| \cdot |Q| = 2^{2(m+n)}$, because if $\theta = true$, then the system moves from any of the $|Q|$ states to any other state. Therefore, the space complexity of the construction of A^S is $O(2^{2(m+n)})$. This is not surprising, since NBA are explicit-state, whereas nFOSs are symbolic, which can be exponentially more succinct. For the time complexity, we additionally have to account for the time it takes to evaluate θ and ϕ for each valuation of the variables. Given a particular valuation of the variables, it is feasible to evaluate θ and ϕ in time polynomial in their sizes $|\theta|$ and $|\phi|$ and the number of variables $n + m$. Thus, the time complexity of the construction of A^S is $O(2^{2(m+n)} \cdot poly(|\theta|, |\phi|, n + m))$ which is in *EXPTIME*. □

Example 2. Consider the nFOS $S = (X = \{x_1, x_2\}, Z = \{z\}, \theta, \phi)$, where θ and ϕ are defined as shown in Fig. 2 (left). Then, the NBA A^S that corresponds to nFOS S, is shown in the right part of Fig. 2. We have that $[\![S]\!]_o = L(A)^S$.

3.2.5. Equivalence checking of nFOSs

In order to solve the view consistency problem for symbolic transition systems we need to be able to check equivalence of nFOS systems w.r.t. their observable behaviors. In the sequel, we provide a theorem which proves that the equivalence problem of nFOS systems w.r.t. their observable behaviors is decidable in *EXPSPACE*. The subsequent material is used in Section 6.

Theorem 4. *Given nFOS systems $S_1 = (X, Z_1, \theta_1, \phi_1)$ and $S_2 = (X, Z_2, \theta_2, \phi_2)$, where $Z_1 \cap Z_2 = \emptyset$, checking whether $[\![S_1]\!]_o = [\![S_2]\!]_o$ is in EXPSPACE.*

[3] Although we can transform nFOSs to NBA, the inverse transformation is not generally possible. This is because nFOSs have no acceptance conditions, therefore they cannot capture Büchi acceptance conditions.

Proof. Consider two nFOS systems $S_i = (X_i, Z_i, \theta_i, \phi_i)$ for $i = 1, 2$. By Theorem 3 we can construct for S_1 and S_2 the NBA A^{S_1} and A^{S_2}, respectively, such that $[\![S_1]\!]_o = L(A^{S_1})$ and $[\![S_2]\!]_o = L(A^{S_2})$. By Theorem 3 we have that the construction of each NBA A^{S_i} is in *EXPTIME* and that the resulting automata are at most exponential in the size of the input nFOS systems. Also the problem of checking $L(A^{S_1}) = L(A^{S_2})$ is an instance of the nondeterministic Buchi automaton (NBA) equivalence problem, which is in *PSPACE* [35]. Hence, checking whether S_1 and S_2 are equivalent w.r.t. their observable behaviors, is decidable and is in *EXPSPACE*. □

Remark 1. The above theorem is a special case of a similar result (Theorem 3.4) studied in [33,32], which compares the observable behaviors of nFOS systems w.r.t. an arbitrary partial order. However, the complexity in Theorem 4 is different from the *PSPACE* result proved in Theorem 3.4 of [32]. In the latter, the complexity is computed in terms of explicit-state representation of the systems rather than in terms of the size of their symbolic representation. On the other hand, Theorem 4 uses the symbolic representation of nFOS systems, which can be exponentially more succinct. This justifies the discrepancy between the two complexity results.

4. Multi-view modeling

In this section, we recall the formal multi-view modeling framework proposed in [33] and also followed in the journal version of that paper [32]. We also provide, in Section 4.3, a new result, not contained in [33,32]. This result is a generic theorem providing a necessary and sufficient condition for view consistency independent of the particular instantiation of systems, views, and abstraction functions. The result (Theorem 5) is used in subsequent sections to derive algorithms for checking view consistency in specific instances of the abstract framework.

4.1. Systems, views, and abstraction functions

Following [33], a system is a set of behaviors. The framework is semantic and abstract, in the sense that there is no restriction on the types of behaviors considered. The set of all possible system behaviors is denoted $\mathcal{U}$. A system $\mathcal{S}$ is a subset of $\mathcal{U}$: $\mathcal{S} \subseteq \mathcal{U}$.

A view is also a set of behaviors, but the view behaviors "live" in a different domain than the system behaviors (cf. examples in [33,32]). Intuitively a view is an incomplete picture of a system, and may therefore be obtained by some kind of *abstraction*, of the system behaviors. Formally, let $\mathcal{D}_i$ denote the i-th *view domain*, *i.e.*, the universe of behaviors of the i-th view.[4] A *view* $\mathcal{V}_i$ from the i-th viewpoint *is a subset of* $\mathcal{D}_i$: $\mathcal{V}_i \subseteq \mathcal{D}_i$. The *abstraction function* corresponding to the i-th viewpoint is a function $\alpha_i : \mathcal{U} \to \mathcal{D}_i$, for $i = 1, \ldots, n$ and $n \in \mathbb{Z}_{>0}$.

We lift abstraction functions to sets in the usual way: $\alpha_i : 2^{\mathcal{U}} \to 2^{\mathcal{D}_i}$, where for $\mathcal{S} \subseteq \mathcal{U}$:

$$\alpha_i(\mathcal{S}) = \{\alpha_i(\rho) \mid \rho \in \mathcal{S}\}$$

Moreover, we also define the inverse (image) of an abstraction function in the usual way: $\alpha_i^{-1} : 2^{\mathcal{D}_i} \to 2^{\mathcal{U}}$, where for $\mathcal{V} \subseteq \mathcal{D}_i$:

$$\alpha_i^{-1}(\mathcal{V}) = \{\rho \in \mathcal{U} \mid \alpha_i(\rho) \in \mathcal{V}\}$$

or equivalently

$$\alpha_i^{-1}(\mathcal{V}) = \bigcup\{\mathcal{S} \subseteq \mathcal{U} \mid \alpha_i(\mathcal{S}) = \mathcal{V}\}.$$

4.2. View consistency

In multi-view modeling, the system $\mathcal{S}$ is typically unknown. Instead, only the views are available. Then, the following questions arise: *are the views consistent with each other?* and *what does consistency mean formally?* The formal framework developed in [33,32] proposes a versatile definition of view consistency which is parameterized by a notion of *conformance*. In the strictest case, conformance is modeled by equality. For simplicity, we only consider this strict notion of view consistency, w.r.t. =, in this paper. Formally, a set of views $\mathcal{V}_1, \ldots, \mathcal{V}_n$ over view domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$, are *consistent with respect to a set of abstraction functions* $\alpha_1, \ldots, \alpha_n$, if there exists a system $\mathcal{S}$ over $\mathcal{U}$ so that $\mathcal{V}_i = \alpha_i(\mathcal{S})$, for all $i = 1, \ldots, n$. We call such a system $\mathcal{S}$ a *witness* to the consistency of $\mathcal{V}_1, \ldots, \mathcal{V}_n$. If there is no such system, then we conclude that the views are inconsistent. For the special case where $n = 1$ (*i.e.*, when we have just one view $\mathcal{V}_1$) we say that $\mathcal{S}$ is a witness to the *lonely consistency* of $\mathcal{V}_1$.

The proof of the following lemma is straightforward and is omitted.

Lemma 6 *(Monotonicity of abstraction functions and their inverses). Let α_i be an abstraction function, and let α_i^{-1} be its inverse. We have: (1) If $\mathcal{S} \subseteq \mathcal{S}'$ then $\alpha_i(\mathcal{S}) \subseteq \alpha_i(\mathcal{S}')$. (2) If $\mathcal{V} \subseteq \mathcal{V}'$ then $\alpha_i^{-1}(\mathcal{V}) \subseteq \alpha_i^{-1}(\mathcal{V}')$. (3) $\alpha_i^{-1}(\alpha_i(\mathcal{S})) \supseteq \mathcal{S}$. (4) $\alpha_i(\alpha_i^{-1}(\mathcal{V})) = \mathcal{V}$.*

[4] There is no a *priori* relation between the system domain $\mathcal{U}$ and the view domain $\mathcal{D}_i$.

4.3. The canonical witness system candidate and its use in a necessary and sufficient condition for view consistency

Given views $\mathcal{V}_1, \ldots, \mathcal{V}_n$ and the corresponding abstraction functions $\alpha_1, \ldots, \alpha_n$, we define the following system, called the *canonical witness system candidate*, and denoted by $\mathcal{S}^\#$:

$$\mathcal{S}^\# = \bigcap_{i=1}^{n} \alpha_i^{-1}(\mathcal{V}_i) \tag{1}$$

Note that $\alpha_i^{-1}(\mathcal{V}_i) \subseteq \mathcal{U}$, for all i, therefore also $\mathcal{S}^\# \subseteq \mathcal{U}$, which means that $\mathcal{S}^\#$ is indeed a system (*i.e.*, a set of behaviors in $\mathcal{U}$).

Theorem 5. *$\mathcal{V}_1, \ldots, \mathcal{V}_n$ are consistent iff for all $i = 1, \ldots, n$, $\alpha_i(\mathcal{S}^\#) = \mathcal{V}_i$.*

Proof. *If part*: If for all $i = 1, \ldots, n$, $\alpha_i(\mathcal{S}^\#) = \mathcal{V}_i$ then $\mathcal{S}^\#$ is a valid witness system, and therefore by definition the views are consistent.

Only if part: Suppose that the views are consistent. So there exists a witness system $\mathcal{S}$. We need to show that $\mathcal{S}^\#$ is also a valid witness. Pick an index i in the range $1, \ldots, n$. We know that $\alpha_i(\mathcal{S}) = \mathcal{V}_i$. By Lemma 6, $\alpha_i^{-1}(\alpha_i(\mathcal{S})) \supseteq \mathcal{S}$. Therefore, $\mathcal{S} \subseteq \alpha_i^{-1}(\mathcal{V}_i)$. Since this holds for all $i = 1, \ldots, n$, we also have $\mathcal{S} \subseteq \bigcap_{i=1}^{n} \alpha_i^{-1}(\mathcal{V}_i)$, *i.e.*, $\mathcal{S} \subseteq \mathcal{S}^\#$. By Lemma 6, $\alpha_i(\mathcal{S}) \subseteq \alpha_i(\mathcal{S}^\#)$, *i.e.*, $\mathcal{V}_i \subseteq \alpha_i(\mathcal{S}^\#)$, for all $i = 1, \ldots, n$. To show that $\mathcal{S}^\#$ is a valid witness, it remains to show that also $\mathcal{V}_i \supseteq \alpha_i(\mathcal{S}^\#)$, for all $i = 1, \ldots, n$. Pick an index i in the range $1, \ldots, n$. By definition of $\mathcal{S}^\#$, we have $\mathcal{S}^\# = \bigcap_{i=1}^{n} \alpha_i^{-1}(\mathcal{V}_i) \subseteq \alpha_i^{-1}(\mathcal{V}_i)$. By Lemma 6, $\alpha_i(\mathcal{S}^\#) \subseteq \alpha_i(\alpha_i^{-1}(\mathcal{V}_i))$, and by Lemma 6, $\alpha_i(\mathcal{S}^\#) \subseteq \mathcal{V}_i$. □

Remark 2. In the proof of Theorem 5 it is shown that $\mathcal{S}^\# \supseteq \mathcal{S}$ for any other witness $\mathcal{S}$. This means that the canonical witness $\mathcal{S}^\#$ is the most general witness (*i.e.*, the greatest w.r.t. the subset order).

Theorem 5 suggests the following generic algorithm to check the consistency of multiple views:

1. Compute the canonical witness system candidate $\mathcal{S}^\#$. This in itself involves computing the inverse abstractions $\alpha_i^{-1}(\mathcal{V}_i)$ and then calculating their intersection.
2. Compute the abstractions $\alpha_i(\mathcal{S}^\#)$ of $\mathcal{S}^\#$, and check whether they are equal to the corresponding views $\mathcal{V}_i$.

In Sections 5 and 6 we show how to instantiate this generic algorithm for systems and views represented by Büchi automata and symbolic transition systems, respectively, where abstraction functions perform periodic sampling.

Remark 3. Note that when $n = 1$, the single view $\mathcal{V}_1 \subseteq \mathcal{D}_1$ is always consistent to itself provided that $a_1^{-1}(\mathcal{V}_1)$ exists. Indeed, in that case the canonical witness is $\mathcal{S}^\# = a_1^{-1}(\mathcal{V}_1)$.

5. Multi-view consistency of Büchi automata w.r.t. periodic sampling

In this section we instantiate the multi-view modeling framework to the case where systems and views are described by Büchi automata, and where abstraction functions perform periodic sampling. We show how the multi-view consistency problem can be solved in this setting by applying Theorem 5. This involves showing how to implement on Büchi automata the various operations needed to compute the canonical witness system candidate from (1) and check that its forward periodic sampling equals each corresponding view.

5.1. Periodic sampling of infinite words and languages

A *period* is any $T \in \mathbb{N}_{>0}$. We define the periodic sampling of an infinite word so that one out of every T letters of the word is selected and the rest are discarded. For simplicity, we assume that there is no initial offset, *i.e.*, the periodic sampling starts at position 0. Let $w = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ be an infinite word, with $a_i \in \Sigma$, for all $i \geq 0$. Let $w(i)$ denote a_i. We define the *periodic sampling abstraction function w.r.t. T*, $\alpha_T : \Sigma^\omega \to \Sigma^\omega$, where for given word $w \in \Sigma^\omega$, $\alpha_T(w)$ is the word $w' \in \Sigma^\omega$ such that $w'(i) = w(i \cdot T)$. For instance, if $T = 2$ and $w = a_0 a_1 a_2 \cdots$, then $w' = a_0 a_2 a_4 \cdots$. We "lift" α_T to sets of words in the usual way. If $L \subseteq \Sigma^\omega$ is a set of words, then $\alpha_T(L) = \{\alpha_T(w) \mid w \in L\}$. In what follows we refer to periodic sampling abstraction functions simply by periodic sampling. Note that α_T is an abstraction function from system domain $\mathcal{U} = \Sigma^\omega$ to view domain $\mathcal{D} = \Sigma^\omega$.

The inverse abstraction function, which we will call *inverse periodic sampling*, is the function $\alpha_T^{-1} : 2^{\Sigma^\omega} \to 2^{\Sigma^\omega}$, defined as explained in Section 4.1, namely: $\alpha_T^{-1}(L) = \{w \in \Sigma^\omega \mid \alpha_T(w) \in L\}$.

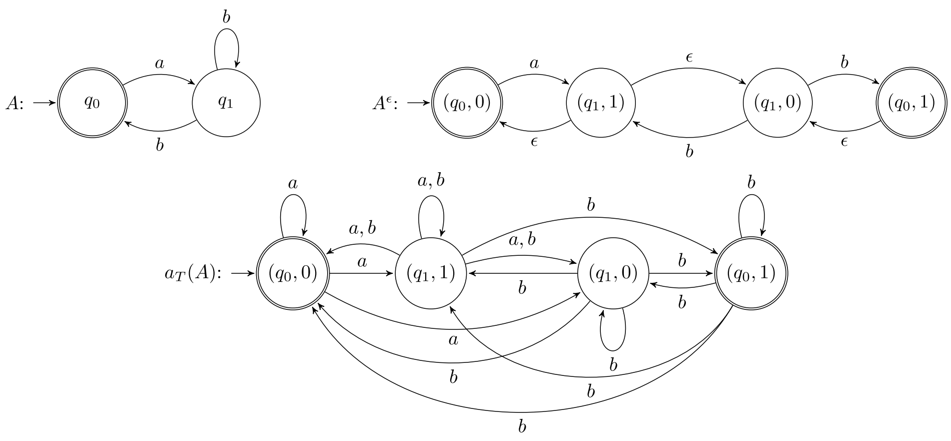

Fig. 3. Example of NBA periodic sampling. NBA A over $\Sigma = \{a, b\}$, and A^ϵ, $a_T(A)$, obtained with period $T = 2$.

5.2. Periodic sampling of Büchi automata

Given a Büchi automaton (without ϵ-transitions) A, and a period T, we want to construct a new Büchi automaton (without ϵ-transitions) denoted $\alpha_T(A)$, such that $L(\alpha_T(A)) = \alpha_T(L(A))$. We will show that such a construction indeed exists, which proves that Büchi automata (or equivalently, ω-regular languages) are closed under (forward) periodic sampling. We assume $T \geq 2$ (if $T = 1$ then $\alpha_T(A)$ is identical to A).

Let $A = (Q, \Sigma, Q_0, \Delta, F)$. We first define A^ϵ, which is a Büchi automaton with ϵ-transitions:

$$A^\epsilon = (Q \times \{0, \ldots, T-1\}, \Sigma \cup \{\epsilon\}, Q_0 \times \{0\}, \Delta', F \times \{0, \ldots, T-1\}) \tag{2}$$

where

$$\begin{aligned}\Delta' = \{&((q,0), a, (q',1)) \mid (q,a,q') \in \Delta\} \\ &\cup \{((q,i), \epsilon, (q', i+1)) \mid 1 \leq i < T-1 \text{ and } (q,a,q') \in \Delta\} \\ &\cup \{((q, T-1), \epsilon, (q', 0)) \mid (q,a,q') \in \Delta\}.\end{aligned}$$

The idea behind A^ϵ is that states are augmented with a modulo-T counter and only the letters on transitions from 0 to 1 are kept. On the remaining transitions, the letters are "hidden", *i.e.*, replaced by ϵ.

Next, we transform A^ϵ into $\alpha_T(A)$ by eliminating from A^ϵ all its ϵ-transitions, as explained in Section 3.1.4.

Example 3. Consider the NBA A over $\Sigma = \{a, b\}$, shown in the top-left part of Fig. 3. Applying periodic sampling on A with period $T = 2$ we obtain the automaton A^ϵ shown in the top-right part of Fig. 3. Removing the epsilon transitions from the latter, we obtain the automaton $a_T(A)$ at the bottom of Fig. 3.

Theorem 6 that follows shows that the language of $\alpha_T(A)$ coincides with the periodic sampling with period T of the language of A.

Theorem 6. $L(\alpha_T(A)) = \alpha_T(L(A))$.

Proof. Let $w = a_0 a_1 \cdots \in \alpha_T(L(A))$. Then, there exists some word $w' = a'_0 a'_1 \cdots \in L(A)$ such that $\alpha_T(w') = w$, *i.e.*, $a'_{i \cdot T} = a_i$ for all $i \geq 0$. By Theorem 1, $L(\alpha_T(A)) = L(A^\epsilon)$, where A^ϵ is the intermediate automaton with ϵ-transitions built during the construction of $\alpha_T(A)$, as defined in (2) above. Therefore, to show $w \in L(\alpha_T(A))$ it suffices to show $w \in L(A^\epsilon)$. For this, and since $w' \in L(A)$, consider an infinite accepting run $P_{w'} = q_0 \xrightarrow{a'_0} q_1 \xrightarrow{a'_1} q_2 \cdots q_{T-1} \xrightarrow{a'_{T-1}} q_T \xrightarrow{a'_T} q_{T+1} \cdots$ of A over w'. Define $P_w = (q_0, 0) \xrightarrow{a'_0} (q_1, 1) \xrightarrow{\epsilon} (q_2, 2) \cdots (q_{T-1}, T-1) \xrightarrow{\epsilon} (q_T, 0) \xrightarrow{a'_T} (q_{T+1}, 1) \cdots$. Observe that, by definition of A^ϵ, P_w is an accepting run of A^ϵ over w, and hence $w \in L(A^\epsilon)$.

Conversely, let $w = a_0 a_1 \cdots \in L(\alpha_T(A))$. By Theorem 1, $L(A^\epsilon) = L(\alpha_T(A))$, and hence $w \in L(A^\epsilon)$. By construction of A^ϵ, there exists an accepting run P_w of A^ϵ over w, such that $P_w = (q_0, 0) \xrightarrow{a_0} (q_1, 1) \xrightarrow{\epsilon} (q_2, 2) \cdots (q_{T-1}, T-1) \xrightarrow{\epsilon} (q_T, 0) \xrightarrow{a_1} (q_{T+1}, 1) \cdots$. By construction of A^ϵ, this means that A has an accepting run $P_{w'} = q_0 \xrightarrow{a_0} q_1 \xrightarrow{b_1} q_2 \xrightarrow{b_2} \cdots q_{T-1} \xrightarrow{b_{T-1}} q_T \xrightarrow{a_1} q_{T+1} \cdots$ over $w' = a_0 b_1 \cdots b_{T-1} a_1 \cdots$. Therefore, $w' \in L(A)$. Observe that $w = \alpha_T(w')$. Therefore, $w \in \alpha_T(L(A))$ and the proof is complete. $\square$

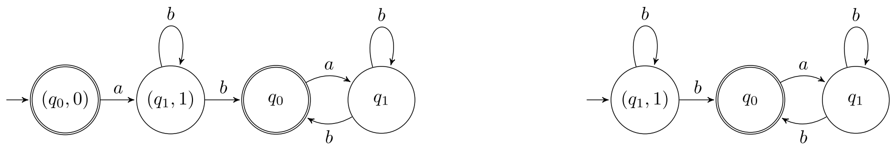

Fig. 4. Left: unfolding the NBA A of the top-left part of Fig. 3 for $\tau = 1$. Right: modifying the initial state and discarding unreachable states.

Theorem 7. *Given a NBA $A = (Q, \Sigma, Q_0, \Delta, F)$ we can construct: (1) the NBA A^ϵ described above in $O(T \cdot (|Q| + |\Delta|))$ time and space; (2) the NBA $a_T(A)$ in $O(T^3 \cdot |Q|^3 + |\Sigma| \cdot T^2 \cdot |Q|^2)$ time and $O(|\Sigma| \cdot T^2 \cdot |Q|^2)$ space.*

Proof. Part (1): Let $A^\epsilon = (Q', \Sigma', Q_0', \Delta', F')$ be constructed according to Section 5.2. Then, we have that $|Q'| = T \cdot |Q|$, $|\Sigma'| = |\Sigma| + 1$, and $|\Delta'| = T \cdot |\Delta|$. Therefore, constructing A^ϵ requires $O(T \cdot (|Q| + |\Delta|))$ time and space.

Part (2): The NBA A has at most $|\Delta| = |\Sigma| \cdot |Q|^2$ transitions. By part (1) of the current proof, the automaton A^ϵ has $|Q'| = T \cdot |Q|$ states and $|\Delta'| = T \cdot |\Delta| \leq T \cdot |\Sigma| \cdot |Q|^2$ transitions. Following the construction in Section 3.1.4, we derive the NBA $a_T(A) = (Q', \Sigma, Q_0', \Delta'', F')$. Then, the number of the transitions in $a_T(A)$ is at most $|\Delta''| = |\Sigma| \cdot (T \cdot |Q|)^2$. Hence, the construction of $a_T(A)$ requires $T \cdot |Q| + T \cdot |\Sigma| \cdot |Q|^2 + |\Sigma| \cdot T^2 \cdot |Q|^2$ space, and hence the space complexity is $O(|\Sigma| \cdot T^2 \cdot |Q|^2)$. Computing the closure of epsilon transitions in A^ϵ requires, by Warshall's algorithm [40], $O(T^3 \cdot |Q|^3)$ time. Thus, computing $a_T(A)$ requires at most $T^3 \cdot |Q|^3 + T \cdot |\Sigma| \cdot |Q|^2 + |\Sigma| \cdot T^2 \cdot |Q|^2 + T \cdot |Q|$ steps. Therefore, the time complexity is $O(T^3 \cdot |Q|^3 + |\Sigma| \cdot T^2 \cdot |Q|^2)$. □

Both the constructions of A^ϵ and $a_T(A)$ are polynomial in the size of the input automata and the value of T, but exponential in T's representation. Hence, the problem of constructing $a_T(A)$ is in *EXPTIME*.

As mentioned earlier, throughout this paper we assume for simplicity that sampling is done with an initial phase/offset of 0. A thorough treatment of the problem for offsets greater than 0 is left for future work. Here, we illustrate by example how some of the results can be easily extended to offsets greater than 0. For example, suppose we want to compute the sampling of the infinite word *ababab*$\cdots$ with period $T = 2$. If the offset is 0, the result is *aaa*$\cdots$. If the offset is 1, the result is *bbb*$\cdots$. Suppose we want to compute the sampling of a Büchi automaton A, *i.e.*, of all infinite words accepted by the automaton, w.r.t. some period T and some initial offset $\tau \geq 0$. To do that, we can use the same method presented above for $\tau = 0$, after transforming A into a new automaton A_τ to take into account the possibly non-zero initial offset. To obtain A_τ from A, we *unfold* the transitions of A for τ steps, starting from the initial states of A. We then discard the newly unfolded states up to the first $\tau - 1$ steps, and make the new states reached at the τ-th step the new initial states of A_τ. An example of this process applied to the NBA A of the top-left part of Fig. 3 for $\tau = 1$ is shown in Fig. 4.

5.3. Inverse periodic sampling of Büchi automata

Given a Büchi automaton A, and a period T, we want to construct a new Büchi automaton, which we will denote $\alpha_T^{-1}(A)$, such that $L(\alpha_T^{-1}(A)) = \alpha_T^{-1}(L(A))$. We will show that such a construction indeed exists, which proves that Büchi automata (or equivalently, ω-regular languages) are closed under inverse periodic sampling. We assume $T \geq 2$ (if $T = 1$ then $\alpha_T^{-1}(A)$ is identical to A).

Let $A = (Q, \Sigma, Q_0, \Delta, F)$. We define $\alpha_T^{-1}(A)$ to be the following Büchi automaton:

$$\alpha_T^{-1}(A) = (Q \cup Q \times \{1, \ldots, T-1\}, \Sigma, Q_0, \Delta', F)$$

where

$$\begin{aligned}\Delta' = &\{(q, a, (q', 1)) \mid (q, a, q') \in \Delta\}\\ &\cup \{((q', i), a, (q', i+1)) \mid 1 \leq i < T-1 \text{ and } a \in \Sigma\}\\ &\cup \{((q', T-1), a, q') \mid a \in \Sigma\}.\end{aligned}$$

The idea behind the construction of $\alpha_T^{-1}(A)$ is to replace each transition (q, a, q') of A by a DAG (directed acyclic graph) starting at q and ending at q'. Each run in this DAG starting at q and ending at q' corresponds to a sequence of T transitions, the first of which has the original letter a of the original transition of A, while the remaining may have any letter in Σ (since that letter will be deleted during sampling). A state of $\alpha_T^{-1}(A)$ can be of two kinds: either an original state q of A, or a state of the form (q', i), where q' is a state of A and i is between 1 and $T - 1$ (note that we assume $T \geq 2$, so this range contains at least the number 1). In the latter case, the state (q', i) records the fact that there are $T - i$ steps remaining until the destination state q' is reached.

Example 4. Consider the NBA A over $\Sigma = \{a, b, c, d\}$ shown in the left part of Fig. 5. The inverse periodic sampling of A w.r.t. $T = 3$ is the automaton $a_T^{-1}(A)$ shown to the right part of Fig. 5.

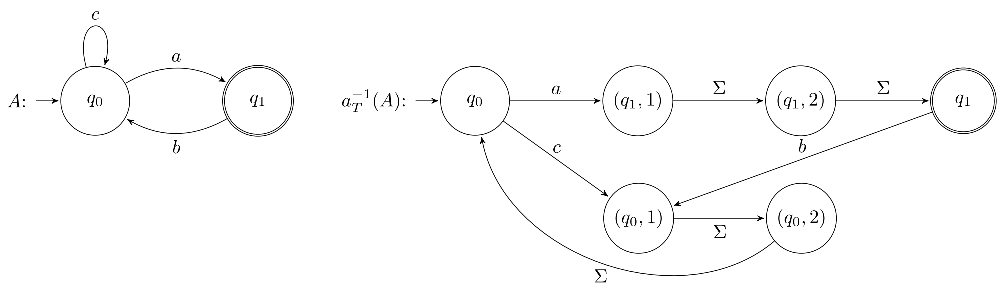

Fig. 5. Example of NBA inverse periodic sampling. NBA A over $\Sigma = \{a, b, c, d\}$ and $a_T^{-1}(A)$ when $T = 3$.

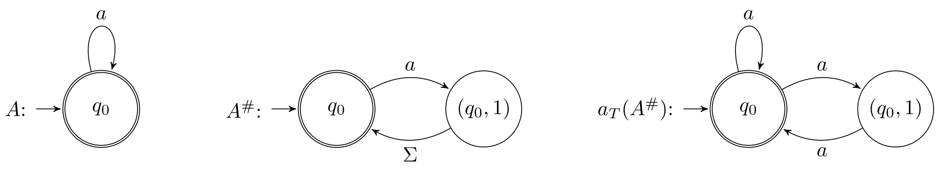

Fig. 6. NBA view A (left), canonical witness (middle), and its periodic sampling (right).

Theorem 8. $L(\alpha_T^{-1}(A)) = \alpha_T^{-1}(L(A))$.

Proof. Let $w = a_0 a_1 \cdots \in L(\alpha_T^{-1}(A))$. By definition of $\alpha_T^{-1}(A)$, an accepting run of $\alpha_T^{-1}(A)$ over w must have the form:

$$\begin{aligned} P_w =& q_0 \xrightarrow{a_0} (q_1,1) \cdots (q_1, T-1) \xrightarrow{a_{T-1}} q_1 \xrightarrow{a_T} (q_2,1) \cdots (q_2, T-1) \xrightarrow{a_{2T-1}} q_2 \\ & \xrightarrow{a_{2T}} (q_3,1) \cdots q_{k-1} \xrightarrow{a_{(k-1)T}} (q_k,1) \cdots (q_k, T-1) \xrightarrow{a_{kT-1}} q_k \xrightarrow{a_{kT}} (q_{k+1},1) \cdots \end{aligned}$$

Let $P_{w'} = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_T} q_2 \cdots$. By definition of $\alpha_T^{-1}(A)$, $P_{w'}$ must be an accepting run of A over w', where $w' = \alpha_T(w)$. Therefore, $\alpha_T(w) \in L(A)$, which means that $w \in \alpha_T^{-1}(L(A))$.

Conversely, let $w = a_0 a_1 \cdots \in \alpha_T^{-1}(L(A))$. Then, there exists a word $w' \in L(A)$ such that $w' = \alpha_T(w) = a_0 a_T a_{2T} \cdots$. Consider an accepting run $P_{w'} = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_T} q_2 \cdots q_{k-1} \xrightarrow{a_{(k-1)T}} q_k \xrightarrow{a_{kT}} q_{k+1} \cdots$, of A over w'. By definition, $\alpha_T^{-1}(A)$ must have a run $P_w = q_0 \xrightarrow{a_0} (q_1,1) \xrightarrow{a_1} (q_1,2) \cdots (q_1, T-1) \xrightarrow{a_{T-1}} q_1 \xrightarrow{a_T} \cdots$ over w. Moreover, since $P_{w'}$ is accepting, P_w is also accepting, because all states visited in $P_{w'}$, namely, $q_0, q_1, q_2, \cdots$, are also visited in P_w. Thus, $w \in L(\alpha_T^{-1}(A))$ and the proof is complete. □

Theorem 9. *Given a NBA $A = (Q, \Sigma, Q_0, \Delta, F)$ we can construct the NBA $a_T^{-1}(A)$ in $O(T \cdot (|Q| + |\Delta|))$ time and space.*

Proof. Let $a_T^{-1}(A) = (Q', \Sigma', Q_0', \Delta', F')$ be constructed according to Section 5.3. Then, $|Q'| = |Q \cup Q \times \{1, \ldots, T-1\}| = T \cdot |Q|$, $|\Sigma'| = |\Sigma|$, and $|\Delta'| = T \cdot |\Delta|$. The latter is obtained by the fact that for every transition in A we create a DAG of $T - 1$ transitions in $a_T^{-1}(A)$. Since $a_T^{-1}(A)$ has T times the number of states and transitions of the input automaton A, the construction of the former requires $O(T \cdot (|Q| + |\Delta|))$ space and time. □

The following is a simple example involving a single view. It is provided to illustrate the difference between an arbitrary witness, and the canonical witness which is the most general, as stated in Remark 2.

Example 5. Consider the NBA view A over the alphabet $\Sigma = \{a, b\}$, shown to the left of Fig. 6. Note that $L(A) = a^\omega = \{aaa \cdots\}$. Let $T = 2$. Then, a witness to the lonely consistency of $L(A)$ is $L(A)$ itself (this is because $\alpha_T(a^\omega) = a^\omega$). But $L(A)$ is not the most general witness. The latter is the canonical witness $L^\# = (a\Sigma)^\omega$ obtained as the language of $A^\# = a_T^{-1}(A)$, shown in Fig. 6. Indeed, applying periodic sampling to $A^\#$ we derive the automaton $a_T(A^\#)$ shown in the rightmost part of Fig. 6. Then, we get that $a_T(L(A^\#)) = L^\#$, and $L^\# \supseteq L(A)$.

5.4. Checking consistency of Büchi automaton views w.r.t. periodic sampling

Next we show how we can apply the results of Theorems 5, 6, and 8 to check view consistency in a multi-periodic sampling setting where the views are ω-regular languages represented by Büchi automata.

Let Σ be a finite alphabet and let $\mathcal{U} = \mathcal{D}_1 = \mathcal{D}_2 = \cdots = \mathcal{D}_n = \Sigma^\omega$, for some $n \geq 1$. Consider n ω-regular languages over Σ, $L_1, L_2, \ldots, L_n \subseteq \Sigma^\omega$, where each L_i is represented by a Büchi automaton A_i, so that $L_i = L(A_i)$, for $i = 1, \ldots, n$. Let $T_1, \ldots, T_n$ be n periods. Then, we consider the following two variants of the consistency problem.

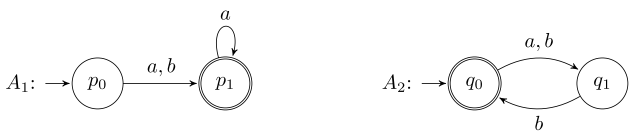

Fig. 7. NBA A_1 and A_2.

Problem 1. Check whether $L_1, \ldots, L_n$ are consistent, *i.e.*, check whether there exists an ω-language L over Σ, such that $\alpha_{T_i}(L) = L_i$ for every $1 \le i \le n$.

Problem 2. Check whether $L_1, \ldots, L_n$ are consistent with an ω-regular witness, *i.e.*, check whether there exists a Büchi automaton A over Σ, such that $\alpha_{T_i}(L(A)) = L_i$ for every $1 \le i \le n$.

The two problems are different a priori, because Problem 1 asks for an arbitrary witness ω-language L, not necessarily regular, whereas Problem 2 asks for a regular witness, represented by a Büchi automaton. Obviously a solution to Problem 2 implies a solution to Problem 1. In the result that follows we show that the two problems are indeed equivalent.

Theorem 10.

1. *There is a solution to Problem 1 if and only if there is a solution to Problem 2.*
2. *Problems 1 and 2 are decidable.*
3. *Problems 1 and 2 can be decided in $2-EXPSPACE$.*

Proof. Part 1: The *if part is trivial*. For the *only if* part, suppose there is a solution for Problem 1, that is, $L_1, \ldots, L_n$ are consistent. Then, by Theorem 5, the ω-language $L^{\#} = \alpha_{T_1}^{-1}(L_1) \cap \cdots \cap \alpha_{T_n}^{-1}(L_n)$ is a witness to their consistency. By Theorem 8, $\alpha_{T_i}^{-1}(L_i) = \alpha_{T_i}^{-1}(L(A_i)) = L(\alpha_{T_i}^{-1}(A_i))$, for $i = 1, \ldots, n$, hence $L^{\#} = L(\alpha_{T_1}^{-1}(A_1)) \cap \cdots \cap L(\alpha_{T_n}^{-1}(A_n))$. By Lemma 2, which can be easily extended to n NBA, there exists a product automaton $A^{\#} = \alpha_{T_1}^{-1}(A_1) \otimes \cdots \otimes \alpha_{T_n}^{-1}(A_n)$ such that $L(A^{\#}) = L(\alpha_{T_1}^{-1}(A_1)) \cap \cdots \cap L(\alpha_{T_n}^{-1}(A_n))$. Therefore, there exists a product automaton $A^{\#}$ such that $L(A^{\#}) = L^{\#}$, and $A^{\#}$ is a solution to Problem 2.

Part 2: We will only consider Problem 1, since Problems 1 and 2 are equivalent. By Theorem 5, $L_1, \ldots, L_n$ are consistent iff $\alpha_{T_i}(L^{\#}) = L_i$ for every $1 \le i \le n$. By Part 1 of the current theorem, $L^{\#} = L(A^{\#})$, hence $L_1, \ldots, L_n$ are consistent iff $\alpha_{T_i}(L(A^{\#})) = L_i$ for every $1 \le i \le n$. By Theorem 6, we have $\alpha_{T_i}(L(A^{\#})) = L(\alpha_{T_i}(A^{\#}))$, for $i = 1, \ldots, n$. Therefore, $L_1, \ldots, L_n$ are consistent iff for all $i = 1, \ldots, n$, $L(\alpha_{T_i}(A^{\#})) = L_i$, or equivalently, $L(\alpha_{T_i}(A^{\#})) = L(A_i)$. The latter equality is a language equivalence problem for Büchi automata, which is decidable [37]. Therefore, the whole process is decidable, by computing inverse periodic sampling, product, forward periodic sampling, and then checking language equivalence on Büchi automata.

Part 3: We will only consider Problem 1, since Problems 1 and 2 are equivalent. By Part 2 of the current theorem, $L_1, \ldots, L_n$ are consistent iff for all $i = 1, \ldots, n$, $L(\alpha_{T_i}(A^{\#})) = L_i$, or equivalently, $L(\alpha_{T_i}(A^{\#})) = L(A_i)$. The latter is a language equivalence problem for Büchi automata, for each $1 \le i \le n$, which is *PSPACE*-complete [37]. By Theorem 9 we have that the construction of $A^{\#}$ is linear in the value of T_i but it is exponential in time and space in the representation of T_i, for $1 \le i \le n$. Fix an i, with $1 \le i \le n$. By Theorem 7, the size of $a_{T_i}(A^{\#})$ is doubly exponential in the representation of T_i, and exponential in the representations of all T_j with $j \ne i$, $1 \le j \le n$. Hence, checking whether $L_1, \ldots, L_n$ are consistent is decided in $2-EXPSPACE$. □

5.5. Example: checking consistency of two NBA views

Consider two NBA views $A_1 = (\{p_0, p_1\}, \Sigma, p_0, \Delta_1, p_1)$ and $A_2 = (\{q_0, q_1\}, \Sigma, q_0, \Delta_2, q_0)$, where $\Sigma = \{a, b, c\}$ and Δ_i are defined as shown in Fig. 7, for $i = 1, 2$. Let $T_1 = 2$ and $T_2 = 3$. We will show that the views A_1 and A_2 are inconsistent. For this, we construct the canonical witness system candidate $A^{\#} = a_{T_1}^{-1}(A_1) \otimes a_{T_2}^{-1}(A_2)$. Then, it suffices to prove that there exists an $i = 1, 2$, such that $L(a_{T_i}(A^{\#})) \ne L(A_i)$.

We first compute the inverse periodic samplings $a_{T_i}^{-1}(A_i)$, for $i = 1, 2$, according to the algorithm described in Section 5.3. The inverse periodic samplings are shown in Fig. 8. For convenience in the product construction that follows, we rename the states of the inverse automata with new labels $x_1, x_2, x_3, y_1, \ldots, y_6$. Next, we compute according to Lemma 2 the product NBA $A^{\#} = a_{T_1}^{-1}(A_1) \otimes a_{T_2}^{-1}(A_2)$, shown in Fig. 9.

In order to show inconsistency, we will show that $L(a_{T_2}(A^{\#})) \ne L(A_2)$. For this, we need to compute $a_{T_2}(A^{\#})$. We do that using the algorithm described in Section 5.2. First, we construct $A^{\#,\epsilon}$, shown in Fig. 10. $A^{\#,\epsilon}$ is obtained from $A^{\#}$ by replacing every second and third transition of $A^{\#}$ with an ϵ transition, to model sampling with period $T_2 = 3$. Then we remove ϵ transitions from $A^{\#,\epsilon}$ as explained in Section 3.1.4, to obtain the NBA $a_{T_2}(A^{\#})$ shown in Fig. 11. In order not to clatter the figure, the letters of (all) outgoing transitions from a state are written below the state's label. For instance, the state $(k_2, 1)$ moves with the letter b to either of states $(k_7, 0)$, $(k_6, 2)$, or $(k_5, 1)$. We observe that all behaviors of $a_{T_2}(A^{\#})$ are of the form $(a \cup b)ba\cdots$, while all behaviors of A_2 are of the form $(a \cup b)b(a \cup b)\cdots$. Hence, $L(a_{T_2}(A^{\#}))$ is missing all

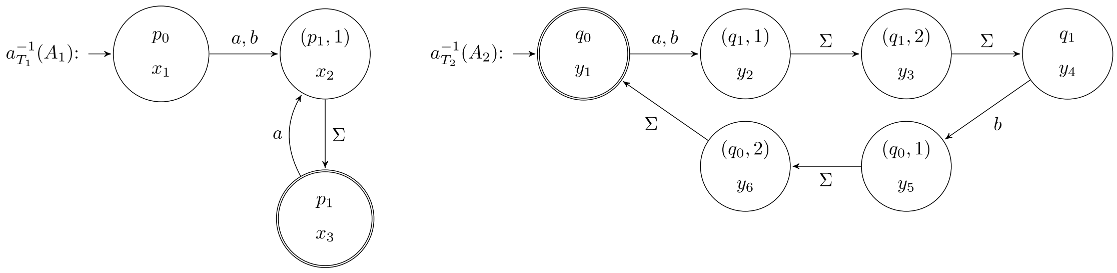

Fig. 8. NBA $a_{T_1}^{-1}(A_1)$ and $a_{T_2}^{-1}(A_2)$, where $T_1 = 2, T_2 = 3$, and A_1, A_2 are the NBA of Fig. 7.

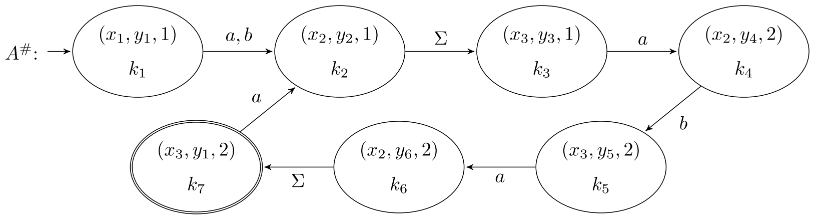

Fig. 9. $A^{\#} = a_{T_1}^{-1}(A_1) \otimes a_{T_2}^{-1}(A_2)$.

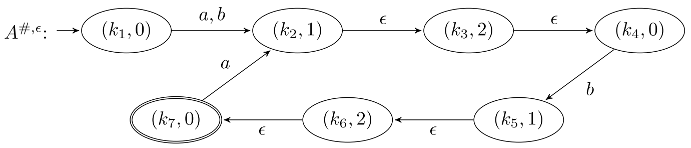

Fig. 10. NBA with ϵ-transitions $A^{\#,\epsilon}$ obtained from $A^{\#}$ of Fig. 9 w.r.t. period $T_2 = 3$.

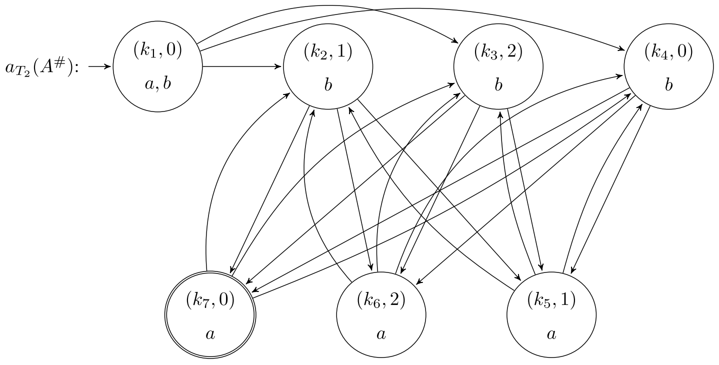

Fig. 11. $a_{T_2}(A^{\#})$ where $A^{\#}$ is the NBA of Fig. 9 and $T_2 = 3$.

words of $L(A_2)$ whose third letter is b. Therefore, $L(a_{T_2}(A^{\#})) \neq L(A_2)$ and by Theorem 5 we conclude that the views A_1 and A_2 are inconsistent.

6. Multi-view consistency of symbolic transition systems w.r.t. periodic sampling

In Section 5 we instantiate the multi-view modeling framework to the case of ω-regular languages and Büchi automata, where abstraction functions perform periodic sampling of infinite words by removing letters from these words. In this section, we instantiate the multi-view modeling framework to the case of symbolic transition systems as defined in Section 3.2. Here, as in Section 5, abstraction functions perform periodic sampling. The difference is that in Section 5 abstraction functions perform periodic sampling of infinite words generated by Büchi automata, whereas in this section abstraction functions perform periodic sampling of infinite sequences of states generated by symbolic transition systems.

Initially, we define the notion of periodic sampling and its inverse for transition systems, and then we investigate the closure of symbolic transition systems under these operations. Afterwards, we solve the multi-view consistency problem in this setting by applying Theorem 5.

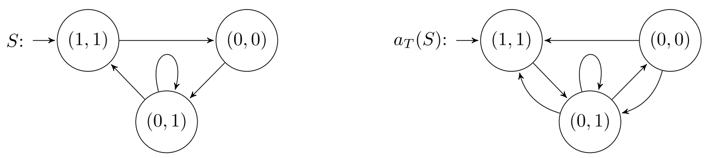

Fig. 12. Example of FOS periodic sampling. FOS S (left) and $a_T(S)$ with period $T = 2$ (right).

6.1. Periodic sampling of transition systems

As in Section 5, a period is any $T \in \mathbb{N}_{>0}$, and periodic sampling starts at position 0. Let X be a finite set of variables. Then the domain of system behaviors is $\mathcal{U}(X)$, and we let the view domain be the same: *i.e.*, $\mathcal{D}(X) = \mathcal{U}(X)$. Given T, the periodic sampling abstraction function from $\mathcal{U}(X)$ to $\mathcal{D}(X)$ w.r.t. period T, denoted by α_T, is defined by the mapping $\alpha_T : \mathcal{U}(X) \to \mathcal{D}(X)$ such that for every behavior $\sigma = s_0 s_1 \cdots \in \mathcal{U}(X)$, $\alpha_T(\sigma) = s'_0 s'_1 \cdots \in \mathcal{D}(X)$ where $s'_i = s_{i \cdot T}$ for every $i \geq 0$. As is standard, we lift the notion of periodic sampling abstraction function to (semantic) transition systems, *i.e.*, sets of behaviors. For a transition system $\mathcal{S} \subseteq \mathcal{U}(X)$, we define $\alpha_T(\mathcal{S}) = \{\alpha_T(\sigma) \mid \sigma \in \mathcal{S}\}$. Since $\alpha_T(\mathcal{S}) \subseteq \mathcal{D}(X)$, $\alpha_T(\mathcal{S})$ is a view over $\mathcal{D}(X)$. Similarly to the previous section, we refer to periodic sampling abstraction functions simply by periodic sampling.

The inverse periodic sampling is defined again as is standard for inverse functions (*cf.* Section 4.1), namely: $\alpha_T^{-1}(\mathcal{V}) = \{\sigma \in \mathcal{U}(X) \mid \alpha_T(\sigma) \in \mathcal{V}\}$ or equivalently $\alpha_T^{-1}(\mathcal{V}) = \bigcup\{\mathcal{S} \subseteq \mathcal{U}(X) \mid \alpha_T(\mathcal{S}) = \mathcal{V}\}$. When $\mathcal{V}$ contains a single behavior σ, we write $\alpha_T^{-1}(\sigma)$ instead of $\alpha_T^{-1}(\{\sigma\})$. It can easily be seen that if $\sigma = s_0 s_1 \cdots \in \mathcal{V} \subseteq \mathcal{D}(X)$, then $\alpha_T^{-1}(\sigma) = \{\sigma' \mid \sigma' = s'_0 s'_1 \cdots \in \mathcal{U}(X) \text{ s.t. } s'_{i \cdot T} = s_i, \text{ for all } i \geq 0\}$.

6.2. Periodic sampling of symbolic transition systems

The above definitions are semantical and apply to sets of behaviors. To be able to deal with systems algorithmically, we must work with syntactic descriptions, *i.e.*, symbolic transition systems such as FOSs and nFOSs. Then, questions arise as to what extent are these formalisms closed under forward and inverse periodic samplings. In this subsection we show that both FOSs and nFOSs are closed under forward periodic sampling. Specifically, we will show that given a symbolic transition system S, we can construct another symbolic transition system which we will denote $\alpha_T(S)$, such that $[\![\alpha_T(S)]\!] = \alpha_T([\![S]\!])$. We assume $T \geq 2$ (if $T = 1$ then $\alpha_T(S)$ is identical to S). Moreover, if S is a FOS then $\alpha_T(S)$ is also a FOS, and if S is an nFOS then $\alpha_T(S)$ is also an nFOS.

Consider an nFOS $S = (X, Z, \theta, \phi)$ and periodic sampling $\alpha_T : \mathcal{U}(X \cup Z) \to \mathcal{D}(X \cup Z)$, with $\mathcal{D}(X \cup Z) = \mathcal{U}(X \cup Z)$. The *periodic sampling of S through α_T*, denoted by $\alpha_T(S)$, is the nFOS $S' = (X, Z, \theta, \phi')$ such that ϕ' contains all pairs of states (s, s') such that S has a run from s to s' of length exactly T. Formally:

$$\phi'(s, s') = \exists s_1, s_2, \ldots, s_{T-1} : \phi(s, s_1) \wedge \phi(s_1, s_2) \wedge \cdots \wedge \phi(s_{T-1}, s').$$

Note that, even though the above formula for ϕ' contains quantifiers over $s_1, s_2, \ldots, s_{T-1}$, they can be eliminated easily because all variables are boolean. Therefore, the final formula for ϕ' is a boolean expression as required.

The theorem that follows states that the semantics of the periodic sampling of S is equal to the periodic sampling of the semantics of S.

Theorem 11. $[\![\alpha_T(S)]\!] = \alpha_T([\![S]\!])$.

Proof. Consider an arbitrary behavior $\sigma = s_0 s_1 s_2 \cdots \in [\![S]\!]$. Applying the periodic sampling α_T to σ we obtain the behavior $\alpha_T(\sigma) = s_0 s_T s_{2T} \cdots \in \alpha_T([\![S]\!])$. Since $\theta(s_0)$, and by construction of $\alpha_T(S)$, $\phi'(s_{i \cdot T}, s_{(i+1) \cdot T})$ for every $i \geq 0$, we get that $\alpha_T(\sigma) \in [\![\alpha_T(S)]\!]$. Hence, $\alpha_T([\![S]\!]) = \{\alpha_T(\sigma) \mid \sigma \in [\![S]\!]\} \subseteq [\![\alpha_T(S)]\!]$. Conversely, let $\sigma' = s'_0 s'_1 s'_2 \cdots \in [\![\alpha_T(S)]\!]$. By definition of $\alpha_T(S)$, it holds that $\theta(s'_0)$. Moreover, for σ' we have that $\phi'(s'_i, s'_{i+1})$, thus there exists a run in S from s'_i to s'_{i+1} of length exactly T for every $i \geq 0$. Then, we obtain the behavior $\sigma = s_0 s_1 s_2 \cdots \in [\![S]\!]$ where $s_{i \cdot T} = s'_i$ for every $i \geq 0$. Hence, $\alpha_T(\sigma) = \sigma' \in \alpha_T([\![S]\!])$ and $[\![\alpha_T(S)]\!] \subseteq \alpha_T([\![S]\!])$ which completes our proof. □

Note that if S is a FOS, *i.e.*, $Z = \emptyset$, then according to its definition $\alpha_T(S)$ is also a FOS. Consequently, Theorem 11 shows that not just nFOSs, but also FOSs, are closed under forward periodic sampling.

Example 6. Consider the FOS S shown to the left of Fig. 12. Applying periodic sampling on S with period $T = 2$ we obtain the FOS $a_T(S)$ shown to the right of the figure.

Theorem 12. *Given an nFOS $S = (X, Z, \theta, \phi)$ the problem of constructing $a_T(S)$ is in EXPSPACE.*

Proof. Let $a_T(S) = (X, Z, \theta, \phi')$, where ϕ' is defined according to the description before Theorem 11. We need to determine the complexity of constructing ϕ'. In particular this involves the complexity of eliminating the existential quantifiers in ϕ'

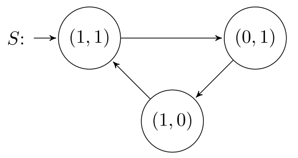

Fig. 13. A FOS whose inverse with respect to period $T=2$ is not a FOS.

since the transition relation of an nFOS is by definition a boolean expression (thus not including quantifiers). We assume that the quantifiers in ϕ' are eliminated by enumerating all satisfying assignments, from which we then derive a formula for the transition relation not involving any quantifiers. Each assignment (including the quantified variables) consists of $(T+1)\cdot(|X|+|Z|)$ variables, because ϕ' contains $T-1$ quantified intermediate states. Checking whether a given assignment satisfies the formula can be determined in polynomial time in ϕ''s length, which is exponential in the representation of T, and linear in the length of ϕ. Thus all assignments can be evaluated in *EXPSPACE*. Constructing the quantifier-free formula from the satisfying assignments is in $O(2^{2\cdot(|X|+|Z|)})$. Thus both determining the satisfying assignments and constructing the quantifier-free formula from them can be performed in *EXPSPACE*, and so we conclude that the problem of constructing $a_T(S)$ is in *EXPSPACE*. □

Recall that views described by symbolic transition systems are checked for consistency w.r.t. their observable behaviors. Observable behaviors are obtained by projection to observable variables. Therefore, we need to be able to combine a periodic sampling abstraction function with projection to observable variables. The following result shows that the above construction behaves correctly under such a combination.

Corollary 1. $[\![\alpha_T(S)]\!]_o = \alpha_T([\![S]\!]_o)$.

Proof. Consider a behavior $\sigma^o = s_0^o s_1^o s_2^o \cdots \in [\![S]\!]_o$. Then, there exists a behavior $\sigma = s_0 s_1 s_2 \cdots \in [\![S]\!]$ such that $h_X(\sigma) = \sigma^o$. Applying the periodic sampling α_T to σ we obtain the behavior $\sigma' = \alpha_T(\sigma) = s_0 s_T s_{2T} \cdots \in \alpha_T([\![S]\!])$. Then, $h_X(\sigma') = s_0^o s_T^o s_{2T}^o \cdots$, and hence $h_X(\sigma') = \alpha_T(\sigma^o)$. Therefore, by $\sigma^o \in [\![S]\!]_o$, $h_X(\sigma') \in \alpha_T([\![S]\!]_o)$. Since $\theta(s_0)$, and by construction of $\alpha_T(S)$, $\phi'(s_{i\cdot T}, s_{(i+1)\cdot T})$ for every $i \geq 0$, we get that $\sigma' \in [\![\alpha_T(S)]\!]$. Therefore, $h_X(\sigma') \in [\![\alpha_T(S)]\!]_o$, and hence $\alpha_T(\sigma^o) \in [\![\alpha_T(S)]\!]_o$. Conversely, let $\sigma^{o\prime} = s_0^{o\prime} s_1^{o\prime} s_2^{o\prime} \cdots \in [\![\alpha_T(S)]\!]_o$. Then, there exists a behavior $\sigma' = s_0' s_1' s_2' \cdots \in [\![\alpha_T(S)]\!]$ such that $h_X(\sigma') = \sigma^{o\prime}$. Since $\theta(s_0')$ and $\phi'(s_i', s_{i+1}')$ for every $i \geq 0$, and by definition of $\alpha_T(S)$, we can construct the behavior $\sigma = s_0 s_1 s_2 \cdots \in [\![S]\!]$ where $s_{i\cdot T} = s_i'$ for every $i \geq 0$. Then, there exists a behavior $\sigma^o = h_X(\sigma) = s_0^o s_1^o s_2^o \cdots \in [\![S]\!]_o$ where $s_{i\cdot T}^o = s_i^{o\prime}$ for every $i \geq 0$, which implies that $\alpha_T(\sigma^o) = \sigma^{o\prime} \in \alpha_T([\![S]\!]_o)$. □

Remark 4. Note that if S is a FOS, *i.e.*, $Z = \emptyset$, then by definition we get that $[\![\alpha_T(S)]\!] = \alpha_T([\![S]\!])$ iff $[\![\alpha_T(S)]\!]_o = \alpha_T([\![S]\!]_o)$.

Remark 5. The result of Corollary 1 shows that we can combine two different kinds of abstraction functions for deriving a view for a system. This is in line with Theorem 5 that imposes no restriction on the type of the abstraction functions for generating a view.

6.3. Inverse periodic sampling of symbolic transition systems

Computing the inverse periodic sampling on FOSs and nFOSs is less straightforward than computing forward periodic sampling. In fact, as we show next, FOSs are *not* closed under inverse periodic sampling, although nFOSs are. Intuitively, the reason why FOSs are not closed under inverse periodic sampling is that in general the inverse system may need extra variables to count how many positions remain until a sampled state is reached. Such extra variables cannot be added in a FOS, because every variable in FOS is observable, and adding extra variables means that the observable behaviors also change. On the other hand, adding extra variables is not a problem in nFOSs, as long as these extra variables are added as unobservable ones.

6.3.1. FOSs are not closed under inverse periodic sampling

Theorem 13. *Given a FOS $S = (X, \theta, \phi)$ and inverse periodic sampling $\alpha_T^{-1} : \mathcal{D}(X) \to \mathcal{U}(X)$, with $\mathcal{D}(X) = \mathcal{U}(X)$, there does not always exist a FOS $S' = (X, \theta', \phi')$ such that $[\![S']\!] = \alpha_T^{-1}([\![S]\!])$.*

Proof. Consider the FOS $S = (\{x, y\}, \theta, \phi)$ where both x and y are boolean variables, $\theta = x \wedge y$ and $\phi = (x \wedge y \to \neg x' \wedge y') \wedge (\neg x \wedge y \to x' \wedge \neg y') \wedge (x \wedge \neg y \to x' \wedge y')$ as shown in Fig. 13.

Suppose that we want to compute the inverse set of behaviors, $\alpha_T^{-1}([\![S]\!])$, with respect to inverse periodic sampling with period $T = 2$. Suppose there exists a FOS $S' = (X', \theta', \phi')$ such that $[\![S']\!] = \alpha_T^{-1}([\![S]\!])$. First, note that X' must be equal

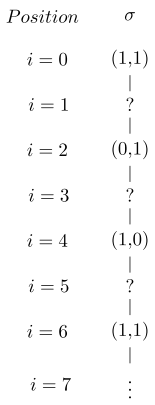

Fig. 14. Form of behavior σ of $\alpha_T^{-1}([\![S]\!])$, where S is the FOS of Fig. 13.

to $\{x, y\}$, otherwise, the states of S' (and therefore also the behaviors in $[\![S']\!]$) are incomparable with those of S. Because $X' = \{x, y\}$, S' can have at most 4 reachable states, since both x and y are boolean variables. Moreover, θ' should coincide with θ. Indeed, our periodic samplings start always at position 0, and $\alpha_T^{-1}([\![S]\!])$ contains by definition all those behaviors over $\mathcal{U}(X)$ which are at state $(1, 1)$ at position 0. Therefore, S' should generate only behaviors whose initial state is $(1, 1)$ in order to coincide semantically with $\alpha_T^{-1}([\![S]\!])$. Hence, θ' equals θ. Then, $\alpha_T^{-1}([\![S]\!])$ contains all infinite behaviors of the form shown in Fig. 14, where the '?' can be replaced by arbitrary states over $\{x, y\}$.

We claim that S' cannot generate all behaviors of the form shown in Fig. 14. Indeed, $\alpha_T^{-1}([\![S]\!])$ contains all behaviors of the form $(1, 1)v \cdots$ where v can be any of the four states $(1, 1), (0, 1), (1, 0)$ or $(0, 0)$. From Fig. 14, the state v of $\alpha_T^{-1}([\![S]\!])$ corresponds to the top-most '?'. From Fig. 14, there must be a transition in S' from state $(1, 1)$ to v. Assume that $v = (1, 1)$. Then, $(1, 1)$ has a self-loop transition, which means that S' can also generate behaviors of the form $(1, 1), (1, 1), (1, 1), \ldots,$ which are not in $\alpha_T^{-1}([\![S]\!])$. Thus, it should be the case that $v \neq (1, 1)$. But this contradicts the requirement that S' be able to generate also behaviors which are at state $(1, 1)$ at position $i = 1$. If S' does not generate such behaviors then it is missing certain behaviors that belong in $\alpha_T^{-1}([\![S]\!])$. Therefore, $[\![S']\!]$ cannot be semantically equal to $\alpha_T^{-1}([\![S]\!])$. □

6.3.2. nFOSs are closed under inverse periodic sampling

As can be seen from the proof of Theorem 13, FOSs are not closed under inverse periodic sampling because the variables of FOSs in general cannot represent all the distinct states in between the sampled states. Hence it is not possible to generate all the behaviors of the inverse semantic system. The problem is that we cannot add extra variables in a FOS, because there every variable has to be observable. However, this is not a problem in nFOSs, where we can add unobservable variables. Adding extra unobservable variables is the main idea behind the inverse periodic sampling construction for nFOSs, which demonstrates that nFOSs are closed under inverse periodic sampling. Note that, due to the presence of extra variables in the inverse periodic sampling system for a given nFOS, closure is obtained w.r.t. to the corresponding observable behaviors. Before we provide the formal construction, we discuss informally the idea behind the construction and illustrate how it works on the example of Fig. 13.

Example 7. Consider again the FOS $S = (\{x, y\}, \theta, \phi\})$ of Fig. 13, which can also be viewed as the nFOS $(\{x, y\}, \emptyset, \theta, \phi\})$ with an empty set of unobservable variables. The inverse periodic sampling of S with period $T = 2$ is the nFOS shown in Fig. 15. s represents the state vector, that is, values for all observable and non-observable variables. In this case, there are only two observable variables x, y, so $s = (0, 1)$ says that $x = 0$ and $y = 1$. The inverse system contains one extra variable c that counts the number of states remaining until a sampled state is reached. c is used to add to the inverse system the proper number of "layers" with states in between each pair of sampled states in the given system. Hence, c is always 0 for the sampled states, and the values of c range in $0, \cdots, T - 1$. In Fig. 15, the value of c is always 1 for every non-sampled state because $T = 2$. Hence, we always add one layer of states in between each pair of sampled states. The states in these layers correspond to the candidates for the symbol '?' in Fig. 14.

The inverse system also contains duplicates of the observable and unobservable variables of the original nFOS denoted by $\hat{s}$. These variables are necessary to define the extra distinct states in each layer, that cannot be obtained by the existing variables of the system. Moreover, these states should also memorize the correct next sampled state. For instance, according to Fig. 14, there should be a transition from $(0, 1)$ to the second ? and from the latter to $(1, 0)$. Then, it should be the case that the inverse system contains a transition from $(0, 1)$ to each of the 4 possible states $(0, 0), (0, 1), (1, 0)$ and $(1, 1)$, and from each of these states a transition back to state $(1, 0)$. Hence $(0, 0), (0, 1), (1, 0)$ and $(1, 1)$ are augmented with $\hat{s} = (1, 0)$ to memorize the next sampled state to be reached; also this differentiates the states of the second layer from those in the first layer, which would not be possible without adding the duplicates.

As stated earlier, $\alpha_T^{-1}(S)$ should be able to generate all observable behaviors of the form shown in Fig. 14, where '?' denotes any arbitrary state over X. Indeed, this is what the system of Fig. 15 does. In each state shown in the figure, we list

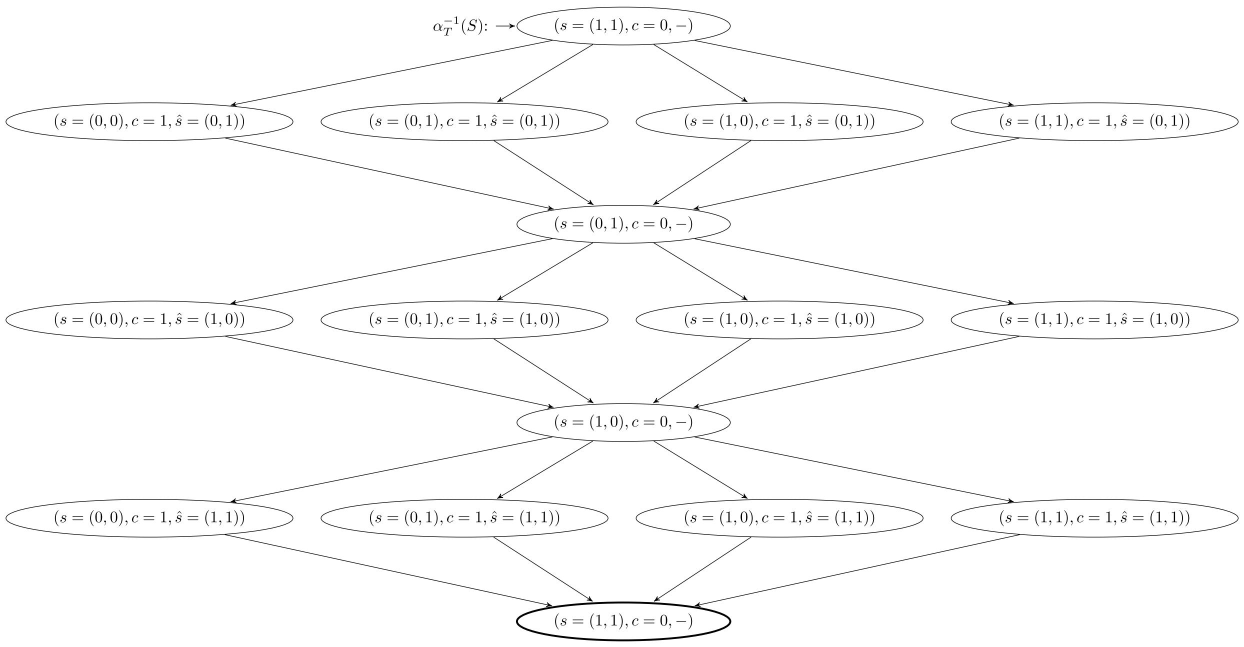
$\alpha_T^{-1}(S)$:
$(s=(1,1),c=0,-)$
$(s=(0,0),c=1,\hat{s}=(0,1))$
$(s=(0,1),c=1,\hat{s}=(0,1))$
$(s=(1,0),c=1,\hat{s}=(0,1))$
$(s=(1,1),c=1,\hat{s}=(0,1))$
$(s=(0,1),c=0,-)$
$(s=(0,0),c=1,\hat{s}=(1,0))$
$(s=(0,1),c=1,\hat{s}=(1,0))$
$(s=(1,0),c=1,\hat{s}=(1,0))$
$(s=(1,1),c=1,\hat{s}=(1,0))$
$(s=(1,0),c=0,-)$
$(s=(0,0),c=1,\hat{s}=(1,1))$
$(s=(0,1),c=1,\hat{s}=(1,1))$
$(s=(1,0),c=1,\hat{s}=(1,1))$
$(s=(1,1),c=1,\hat{s}=(1,1))$
$(s=(1,1),c=0,-)$

Fig. 15. nFOS $\alpha_T^{-1}(S)$ where S is the FOS of Fig. 13 and $T=2$.

the values of s, c, and $\hat{s}$. The $-$ stands for an arbitrary value, so that, for instance, $(s=(1,1), c=0, -)$ represents four states, namely, $(s=(1,1), c=0, \hat{s}=(0,0))$, $(s=(1,1), c=0, \hat{s}=(0,1))$, $(s=(1,1), c=0, \hat{s}=(1,0))$, $(s=(1,1), c=0, \hat{s}=(1,1))$. All these four states are possible initial states. A transition into one such state with $-$ represents multiple transitions to each possible instance of that state. The state in bold at the bottom is the same as the one at the top, indicating that the system loops eventually back to one of its initial states.

Next we present the formal construction. Let $S=(X,Z,\theta,\phi)$ be an nFOS and let T be a period. Let $\widehat{X}=\{\hat{x}\mid x\in X\}$ and $\widehat{Z}=\{\hat{z}\mid z\in Z\}$ be sets of copies (duplicates) of the sets of variables X and Z, respectively. We use the notation $s=(v)_{v\in X\cup Z}$ to denote the vector of all variables in $X\cup Z$. For example, if $X=\{x,y\}$ and $Z=\{z\}$, then $s=(x,y,z)$. We will also use s', $\hat{s}$, and $\hat{s}'$ for the corresponding vectors of primed, hatted, and hatted-primed variables. For example, if $s=(x,y,z)$, then $s'=(x',y',z')$, $\hat{s}=(\hat{x},\hat{y},\hat{z})$, and $\hat{s}'=(\hat{x}',\hat{y}',\hat{z}')$.

We will define a new nFOS $\alpha_T^{-1}(S)$, called *the inverse periodic sampling of S through T*, and we will prove that $[\![\alpha_T^{-1}(S)]\!]_o=\alpha_T^{-1}([\![S]\!]_o)$, that is, the observable behaviors of $\alpha_T^{-1}(S)$ correspond to the inverse of the observable behaviors of S. In the sequel we assume $T\geq 2$; if $T=1$ then we can simply take $\alpha_T^{-1}(S)=S$. We define $\alpha_T^{-1}(S)=(X,Z\cup W,\theta',\phi')$, where

- $W=\{c\}\cup\widehat{X}\cup\widehat{Z}$ where c is a modulo-T counter (for simplicity we will treat c as an integer variable, with the understanding that it can be encoded using $\log T$ boolean variables)

and:

$$\theta'=\theta\wedge c=0 \tag{3}$$

$$\phi'=\big(c=0\wedge c'=1\wedge\exists s':\phi\wedge\hat{s}'=s'\big) \tag{4}$$

$$\vee\big(0<c<T-1\wedge c'=c+1\wedge\hat{s}'=\hat{s}\big) \tag{5}$$

$$\vee\big(c=T-1\wedge c'=0\wedge s'=\hat{s}\big) \tag{6}$$

The intuition is as follows. An initial state of $\alpha_T^{-1}(S)$ is an initial state of S augmented by the modulo-T counter c initialized to 0. Notice that the duplicate variables $\hat{s}$ of $\alpha_T^{-1}(S)$ can take any arbitrary value initially. The definition of the transition relation ϕ' of $\alpha_T^{-1}(S)$ is broken into three disjoint cases, depending on the value of the counter c.

- The first case is when $c=0$. In that case, if S has a transition (s,s'), then $\alpha_T^{-1}(S)$ has a transition from $(s,c=0,\_)$ to $(\_,c=1,\hat{s})$, where $\hat{s}$ memorizes s' and where the _ variables can take arbitrary values. The idea is that we need to remember for the next $T-1$ transitions the destination state s' in which we need to arrive after T transitions, when the next sampling point occurs. In-between, the observable variables s can take any value, since these values will be sampled away.

- The second case is when $0 < c < T-1$. In that case, the duplicate variables $\hat{s}$ remain constant during the transition, while the observable variables s can take any value (and are thus unconstrained in ϕ').
- The third case is when $c = T-1$. In that case, we need to copy the memorized destination state from $\hat{s}$ to s, and reset the counter to 0.

The construction of $\alpha_T^{-1}(S)$ is similar to the construction of inverse periodic sampling of Büchi automata given in Section 5.3, where a modulo-T counter is also used. The difference is that in the case of Büchi automata the sampling is done on the letters annotating the transitions, instead of on the states, and for this reason, that construction does not require the use of duplicate state variables for memorization of the target state (which is memorized directly at the first step from $c=0$ to $c=1$).

In the sequel we prove the closure of nFOSs under inverse periodic sampling. Firstly, we state an auxiliary lemma which shows that the hiding function and the periodic sampling commute when applied to the behaviors of symbolic transition systems.

Lemma 7. *Given an nFOS system $S = (X, Z, \theta, \phi)$ and periodic sampling α_T, it holds that $h_\diamond(\alpha_T(\llbracket S \rrbracket)) = \alpha_T(h_\diamond(\llbracket S \rrbracket))$ where $\diamond$ is either X or Z.*

Proof. We only prove that $h_X(\alpha_T(\llbracket S \rrbracket)) = \alpha_T(h_X(\llbracket S \rrbracket))$. The proof of $h_Z(\alpha_T(\llbracket S \rrbracket)) = \alpha_T(h_Z(\llbracket S \rrbracket))$ is similar. We first prove that $h_X(\alpha_T(\llbracket S \rrbracket)) \subseteq \alpha_T(h_X(\llbracket S \rrbracket))$. Let $\sigma = s_0 s_1 s_2 \cdots \in h_X(\alpha_T(\llbracket S \rrbracket))$. There exists some behavior $\sigma' = s'_0 s'_1 s'_2 \cdots \in \alpha_T(\llbracket S \rrbracket)$ such that $h_X(\sigma') = \sigma$ *i.e.*, $h_X(s'_i) = s_i$ for $i \geq 0$. Moreover, $\sigma' \in \alpha_T(\llbracket S \rrbracket)$, hence there exists some $\sigma'' = s''_0 s''_1 s''_2 \cdots \in \llbracket S \rrbracket$ such that $\alpha_T(\sigma'') = \sigma'$, *i.e.*, $s''_{i \cdot T} = s'_i$ for $i \geq 0$. Consider the behavior $\sigma'' \in \llbracket S \rrbracket$. Then, $h_X(\sigma'') = h_X(s''_0) h_X(s''_1) h_X(s''_2) \cdots \in h_X(\llbracket S \rrbracket)$, where by construction $h_X(s''_{i \cdot T}) = h_X(s'_i) = s_i$ for $i \geq 0$. Moreover, $\alpha_T(h_X(\sigma'')) \in \alpha_T(h_X(\llbracket S \rrbracket))$, and $\alpha_T(h_X(\sigma'')) = \alpha_T(h_X(s''_0) h_X(s''_1) h_X(s''_2) \cdots) = h_X(s''_0) h_X(s''_T) h_X(s''_{2T}) \cdots = s_0 s_1 s_2 \cdots$. Hence $\alpha_T(h_X(\sigma'')) = \sigma$, and $\sigma \in \alpha_T(h_X(\llbracket S \rrbracket))$. Similarly, it can be proved that $\alpha_T(h_X(\llbracket S \rrbracket)) \subseteq h_X(\alpha_T(\llbracket S \rrbracket))$. □

Theorem 14. *Given an nFOS system $S = (X, Z, \theta, \phi)$ and inverse periodic sampling $\alpha_T^{-1} : \mathcal{D}(X \cup Z) \to \mathcal{U}(X \cup Z)$ with $\mathcal{D}(X \cup Z) = \mathcal{U}(X \cup Z)$, it holds that: $\llbracket \alpha_T^{-1}(S) \rrbracket_o = \alpha_T^{-1}(\llbracket S \rrbracket_o)$.*

Proof. Let $\sigma^o = s^o_0 s^o_1 s^o_2 \cdots \in \alpha_T^{-1}(\llbracket S \rrbracket_o)$. Then, there exists a behavior $\sigma^{o\prime} = s^{o\prime}_0 s^{o\prime}_1 s^{o\prime}_2 \cdots \in \llbracket S \rrbracket_o$ such that $\sigma^{o\prime} = \alpha_T(\sigma^o)$, *i.e.*, $s^o_{i \cdot T} = s^{o\prime}_i$ for every $i \geq 0$. Moreover, there exists a behavior $\sigma' = s'_0 s'_1 s'_2 \cdots \in \llbracket S \rrbracket$ such that $h_X(\sigma') = \sigma^{o\prime}$. We consider the behavior $\sigma = s_0 s_1 s_2 \cdots \in \mathcal{U}(X \cup Z)$ such that $h_X(\sigma) = \sigma^o$ and $s_{i \cdot T} = s'_i$ for every $i \geq 0$. We extend σ over the set $W = \{c\} \cup \widehat{X} \cup \widehat{Z}$ and we construct the behavior $\sigma^w = s^w_0 s^w_1 s^w_2 \cdots \in \mathcal{U}(X \cup Z \cup W)$, satisfying the following conditions:

1. $h_c(s^w_k) = 0$ for $k = i \cdot T$ with $i \geq 0$;
2. $h_{\widehat{X} \cup \widehat{Z}}(s^w_k) = s_{(i+1) \cdot T} = s'_{i+1}$ and $h_c(s^w_k) = k - i \cdot T$ for $i \cdot T < k < (i+1) \cdot T$ with $i \geq 0$.

By construction of σ^w, $h_{X \cup Z}(s^w_{i \cdot T}) = s_{i \cdot T} = s'_i$ for $i \geq 0$. Then, by 1 for $i = 0$ and by the condition $\theta(h_{X \cup Z}(s^w_0))$, we get that $\theta'(s^w_0)$. Also, by 1, 2, and $h_{X \cup Z}(\sigma^w) = \sigma$, we get that $\phi'(s^w_k, s^w_{k+1})$ for every $k \geq 0$. Therefore, $\sigma^w \in \llbracket \alpha_T^{-1}(S) \rrbracket$. Moreover, $h_X(\sigma) = \sigma^o$, and by construction of σ^w, $h_X(\sigma^w) = h_X(\sigma)$ which implies that $h_X(\sigma^w) = \sigma^o$. Hence, by $\sigma^w \in \llbracket \alpha_T^{-1}(S) \rrbracket$, we get that $h_X(\sigma^w) = \sigma^o \in \llbracket \alpha_T^{-1}(S) \rrbracket_o$.

Conversely, let $\sigma^o = s^o_0 s^o_1 s^o_2 \cdots \in \llbracket \alpha_T^{-1}(S) \rrbracket_o$. Then, there exists a behavior $\sigma = s_0 s_1 s_2 \cdots \in \llbracket \alpha_T^{-1}(S) \rrbracket$ such that $h_X(\sigma) = \sigma^o$. Applying periodic sampling α_T on σ we get the behavior $\alpha_T(\sigma) = s_0 s_T s_{2T} \cdots$, where by definition of $\alpha_T^{-1}(S)$, $\theta'(s_0)$, and hence $\theta(s_0)$ and $c = 0$; and for every $i \geq 0$, $\phi'(s_i, s_{i+1})$, and hence $\phi(s_{i \cdot T}, s_{(i+1) \cdot T})$ and $c = 0$. Therefore, $\sigma' = h_{X \cup Z}(\alpha_T(\sigma)) \in \llbracket S \rrbracket$, and hence $h_X(\sigma') \in \llbracket S \rrbracket_o$. By applying Lemma 7 for the case $\llbracket S \rrbracket$ is singleton, $h_X(\sigma') = h_X(\alpha_T(\sigma)) = \alpha_T(h_X(\sigma))$, and since $h_X(\sigma) = \sigma^o$, we get that $h_X(\sigma') = \alpha_T(\sigma^o)$. Moreover, by $h_X(\sigma') \in \llbracket S \rrbracket_o$, $\alpha_T(\sigma^o) = s^o_0 s^o_T s^o_{2T} \cdots \in \llbracket S \rrbracket_o$, which implies that $\sigma^o \in \alpha_T^{-1}(\llbracket S \rrbracket_o)$. □

Theorem 15. *Given an nFOS $S = (X, Z, \theta, \phi)$ the problem of constructing $a_T^{-1}(S)$ is in $O(T \cdot (\log T + |X| + |Z|) + |\phi| + |\theta|)$.*

Proof. Let $n = |X| + |Z|$ and $a_T^{-1}(S) = (X, Z \cup W, \theta', \phi')$ be defined as in the procedure before Lemma 7. For the complexity analysis we assume that the modulo-T counter c is represented by $\log T$ boolean variables. Then, the number of variables, defining the states in $a_T^{-1}(S)$ is $2n + \lvert \log T \rvert$. From Equation (3), the size of θ is $|\theta'| = O(|\theta| + \log T)$. Next we calculate the size of ϕ' according to its syntactic construction in the Equations (4)–(6). The part of ϕ' defined in (4) has size $O(\log T + |\phi|)$, since the quantifier elimination of "$\exists s' : \phi \wedge \hat{s} = s'$" can be obtained by simply replacing s' by $\hat{s}$ in ϕ. Then, each of the $T-2$ parts of the disjunction defined in (5) has size $O(n + \log T)$. Similarly, the part of ϕ' in (6) has size $O(n + \log T)$.

Hence, we get that the size of ϕ' is in $O(T \cdot (\log T + n) + |\phi|)$, and the overall time complexity of constructing $a_T^{-1}(S)$ is in $O(T \cdot (\log T + n) + |\phi| + |\theta|)$. □

6.4. *Checking consistency of symbolic transition system views w.r.t. periodic sampling*

Next we apply the previous results to show how we can check view consistency in a multi-periodic sampling setting where the views are symbolic transition systems (FOSs or nFOSs). Let X be a finite set of variables and let $\mathcal{U} = \mathcal{D}_1 = \mathcal{D}_2 = \cdots = \mathcal{D}_n = \mathcal{U}(X)$, for some $n \geq 1$. Consider n views represented as nFOSs $S_i = (X, Z_i, \theta_i, \phi_i)$, for $i = 1, \ldots, n$, where Z_i are disjoint. Note that S_i may be a FOS as a special case, if $Z_i = \emptyset$. Also note that $\mathcal{D}_i = \mathcal{U}(X)$, which means that unobservable variables are not included in the semantics of the views. In other words, semantically view S_i is the set $[\![S_i]\!]_o \subseteq \mathcal{U}(X)$. Let $T_1, \ldots, T_n$ be n periods. We consider several variants of the view consistency problem, depending on whether we seek a witness system which is semantic, nFOS, or FOS.

Problem 3. Check whether $S_1, \ldots, S_n$ are consistent, *i.e.*, check whether there exists $\mathcal{S} \subseteq \mathcal{U}(X)$ such that $\alpha_{T_i}(\mathcal{S}) = [\![S_i]\!]_o$ for every $1 \leq i \leq n$.

Problem 4. Check whether $S_1, \ldots, S_n$ are consistent with an nFOS witness, *i.e.*, check whether there exists an nFOS $S = (X, Z, \theta, \phi)$, for some $Z \supseteq Z_1 \cup \cdots \cup Z_n$, such that $\alpha_{T_i}([\![S]\!]_o) = [\![S_i]\!]_o$ for every $1 \leq i \leq n$.

Problem 5. Check whether $S_1, \ldots, S_n$ are consistent with a FOS witness, *i.e.*, check whether there exists a FOS $S = (X, \theta, \phi)$, such that $\alpha_{T_i}([\![S]\!]) = [\![S_i]\!]_o$ for every $1 \leq i \leq n$.

Observe that the three problems are different: Problem 3 asks for a *semantic* witness system, not necessarily representable as a symbolic transition system, while Problems 4 and 5 ask for a symbolic transition witness system with or without internal variables respectively. Clearly, a solution to Problem 5 is also a solution to Problem 4, and a solution to Problem 4 is also a solution to Problem 3. As shown below (and also shown in [27]) Problems 4 and 5 are not equivalent, that is, existence of an nFOS witness does not always imply existence of a FOS witness. As for Problems 3 and 4, it was left as an open question in [27] whether existence of a solution to Problem 3 also implies existence of a solution to Problem 4. We next provide a positive answer to this question:

Theorem 16. *There is a solution to Problem 3 if and only if there is a solution to Problem 4.*

Proof. The *if part* is trivial. For the *only if* part, suppose Problem 3 has a solution, *i.e.*, $S_1, \ldots, S_n$ are consistent. Then, by Theorem 5, the transition system $\mathcal{S}^\# = \alpha_{T_1}^{-1}([\![S_1]\!]_o) \cap \cdots \cap \alpha_{T_n}^{-1}([\![S_n]\!]_o)$ is a witness to their consistency. By Theorem 14, $\alpha_{T_i}^{-1}([\![S_i]\!]_o) = [\![\alpha_{T_i}^{-1}(S_i)]\!]_o$, for $i = 1, \ldots, n$, hence $\mathcal{S}^\# = [\![\alpha_{T_1}^{-1}(S_1)]\!]_o \cap \cdots \cap [\![\alpha_{T_n}^{-1}(S_n)]\!]_o$. By Lemma 5 (which can be easily extended to n instead of just 2 systems), there exists an nFOS system $S^\#$ such that $[\![S^\#]\!]_o = \mathcal{S}^\# = [\![\alpha_{T_1}^{-1}(S_1)]\!]_o \cap \cdots \cap [\![\alpha_{T_n}^{-1}(S_n)]\!]_o$, and $S^\#$ is a solution to Problem 4. □

Next we show why existence of a solution to Problem 4 does not generally imply existence of a solution to Problem 5. Indeed, consider the case of having a single view described by an nFOS, say S_1. Moreover, suppose that S_1 cannot be represented as a FOS (as mentioned in Section 3.2, there are such cases). Assume that the period is $T_1 = 1$. Then, S_1 is a witness system to its own consistency, since $\alpha_{T_1}(S_1) = S_1$, but clearly there is no FOS witness system, since we assumed that S_1 cannot be represented by a FOS. This example shows that in general existence of an nFOS witness does not imply existence of a FOS witness. This holds even when all views are FOS, as shown by Example 8 which follows.

Example 8. Consider a single view described by the FOS $S = (\{x, y\}, \theta, \phi\})$ of Fig. 13, and let the period be $T = 2$. Then, a witness system to the "lonely" consistency of S is the nFOS system shown in Fig. 15, *i.e.*, the system $\alpha_T^{-1}(S)$ describing its inverse periodic sampling w.r.t. to period $T = 2$. According to Theorem 13, there exists no FOS capturing $\alpha_T^{-1}(S)$, and hence there is no FOS witness system to the consistency of S with itself.

Theorem 17. *Problems 3, 4, and 5 are decidable.*

Proof. Problem 5 is trivially decidable, as there is a finite number of candidate FOS witness systems (since the number of state variables is finite and their domains are also finite). Thus, we can simply enumerate all possible FOS over the given set of variables X and check whether one of them is a valid witness w.r.t. the given views and periods. This brute-force method is not practical. Finding more efficient ways to solve Problem 5 is an open question.

Next we consider the decidability of Problems 3 and 4. There are several ways to prove this result, each resulting in a different method for checking view consistency. Therefore we provide several alternative proofs.

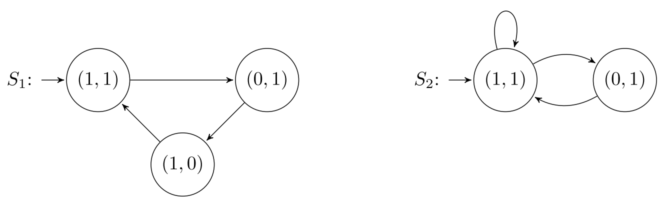

Fig. 16. nFOS S_1 and S_2.

First, we can show that Problem 3 is decidable; then Theorem 16 implies that Problem 4 is also decidable. Consider the set of nFOS views $S_i = (X, Z_i, \theta_i, \phi_i)$ where Z_i are disjoint, for $1 \leq i \leq n$. By Theorem 3, we can construct for every nFOS S_i, the NBA A^{S_i}, such that $[\![S_i]\!]_o = L(A^{S_i})$, for $i \geq 1$. We let $L(A^{S_i}) = L^{S_i}$ for $i \geq 1$. Then, checking whether there exists a semantic witness system to the consistency of $S_1, \ldots, S_n$, *i.e.*, whether there exists system $\mathcal{S} \subseteq \mathcal{U}(X)$ such that $\alpha_{T_i}(\mathcal{S}) = [\![S_i]\!]_o$, reduces to checking whether there exists an ω-regular language over Σ, $L \subseteq \Sigma^\omega$, such that $\alpha_{T_i}(L) = L^{S_i}$ for $i \geq 1$. However, the latter problem is an instance of Problem 1, which by Part 2 of Theorem 10, is decidable. Therefore, Problem 3 is decidable. Notice that using this method, in case of a positive answer to consistency, we can also obtain a witness system represented as a Büchi automaton (*cf.* the proof of Theorem 10).

Alternatively, we can also show directly that Problem 4 is decidable. Consider the nFOS systems $\alpha_{T_i}^{-1}(S_i)$, for each nFOS view S_i for $i = 1, \ldots, n$, respectively. By Lemma 5 (which can be easily extended to n instead of just 2 systems) there exists an nFOS $S^\#$ such that $[\![S^\#]\!]_o = [\![\alpha_{T_1}^{-1}(S_1)]\!]_o \cap \cdots \cap [\![\alpha_{T_n}^{-1}(S_n)]\!]_o$ for $i = 1, \ldots, n$. By Theorem 5, $S_1, \ldots, S_n$ are consistent iff for all $i = 1, \ldots, n$, $\alpha_{T_i}([\![S^\#]\!]_o) = [\![S_i]\!]_o$. By Corollary 1, we have $\alpha_{T_i}([\![S^\#]\!]_o) = [\![\alpha_{T_i}(S^\#)]\!]_o$, for $i = 1, \ldots, n$. Therefore, $S_1, \ldots, S_n$ are consistent iff for all $i = 1, \ldots, n$, $[\![\alpha_{T_i}(S^\#)]\!]_o = [\![S_i]\!]_o$. The latter equality is an equivalence problem for nFOSs, which by Theorem 4 is decidable. Using this method, in case of a positive answer to consistency, we obtain a witness system represented directly as an nFOS. $\square$

Theorem 18.

1. *Problem 3 can be decided in* $3-$*EXPSPACE.*
2. *Problem 4 can be decided in* $3-$*EXPSPACE.*
3. *Problem 5 can be decided in* $2-$*EXPSPACE.*

Proof. Part 2: By the last part of the proof of Theorem 17, we get that solving Problem 4 is equivalent to checking whether $[\![\alpha_{T_i}(S^\#)]\!]_o = [\![S_i]\!]_o$. By definition of $S^\#$ in (1), and by Theorem 15, we get that the construction of $S^\#$ is in *EXPSPACE*. Then, by Theorem 12, the periodic sampling $\alpha_{T_i}(S^\#)$ is computed in *EXPSPACE*, and thus in $2-$*EXPSPACE* in terms of the inputs of Problem 4. Finally, checking $[\![\alpha_{T_i}(S^\#)]\!]_o = [\![S_i]\!]_o$ is, by Theorem 4, in *EXPSPACE*, and hence solving Problem 4 can be decided in $3-$*EXPSPACE*.

Part 1: By Theorem 16, we get that Problem 3 is equivalent to Problem 4, which by part 1 of the current proof can be solved in $3-$*EXPSPACE*. Hence, Problem 3 is also in $3-$*EXPSPACE*.

Part 3: The domain of variables X is finite. Let $n = |X|$. The construction of each possible FOS witness system S requires at most 2^n states with at most $2^{2\cdot n}$ transitions. By Theorem 12, computing $\alpha_{T_i}([\![S]\!])$ is in $2-$*EXPTIME*, for $1, \ldots, n$. Then, checking whether $\alpha_{T_i}([\![S]\!]) = [\![S_i]\!]_o$ is an instance of the nFOSs equivalence problem w.r.t. their observable behaviors, which by Theorem 4 is in *EXPSPACE*. Therefore, solving Problem 5 is in $2-$*EXPSPACE*. $\square$

6.5. Example: checking consistency of two FOS views

Consider the two FOS views $S_1 = (X = \{x_1, x_2\}, \theta_1, \phi_1)$ and $S_2 = (X = \{x_1, x_2\}, \theta_2, \phi_2)$ of Fig. 16. S_1 and S_2 can also be viewed as the nFOSs $(X = \{x_1, x_2\}, Z_1 = \emptyset, \theta_1, \phi_1)$ and $(X = \{x_1, x_2\}, Z_2 = \emptyset, \theta_2, \phi_2)$, respectively. We remark that S_1 is identical to the system shown in Fig. 13 and is repeated here for convenience.

Let $T_1 = 2$ and $T_2 = 3$. We prove that S_1 and S_2 are inconsistent. For this, we construct the canonical witness system candidate $S^\# = a_{T_1}^{-1}(S_1) \otimes a_{T_2}^{-1}(S_2)$. Then, it suffices to prove that there exists an $i = 1, 2$ such that $[\![a_{T_i}(S^\#)]\!]_o \neq [\![S_i]\!]_o$.

We first compute according to the method described in Section 6.3.2 the inverse nFOSs $a_{T_1}^{-1}(S_1)$ and $a_{T_2}^{-1}(S_2)$, shown respectively in Figs. 17 and 18. We remark that $a_{T_1}^{-1}(S_1)$ is identical to the nFOS of Fig. 15 except that the annotations $\hat{s}$ have been removed in order to simplify the figure. In both Figs. 17 and 18, nodes in bold correspond to duplicated states where the system loops back to. We have duplicated those states in order to make the transitions easier to follow.

We then follow the construction of Lemma 5 to compute $S^\# = a_{T_1}^{-1}(S_1) \otimes a_{T_2}^{-1}(S_2)$, shown in Fig. 19.

Finally, we compute according to the method described in Section 6.2 the periodic sampling of $S^\#$ w.r.t. $T_1 = 2$ and $T_2 = 3$, obtaining $a_{T_1}(S^\#)$ and $a_{T_2}(S^\#)$ shown respectively in Figs. 20 and 21. We observe that all the observable behaviors of $a_{T_1}(S^\#)$ coincide with $[\![S_1]\!]_o = ((1,1)(0,1)(1,0))^\omega$. However, this is not the case for $a_{T_2}(S^\#)$ and $[\![S_2]\!]_o$. For instance,

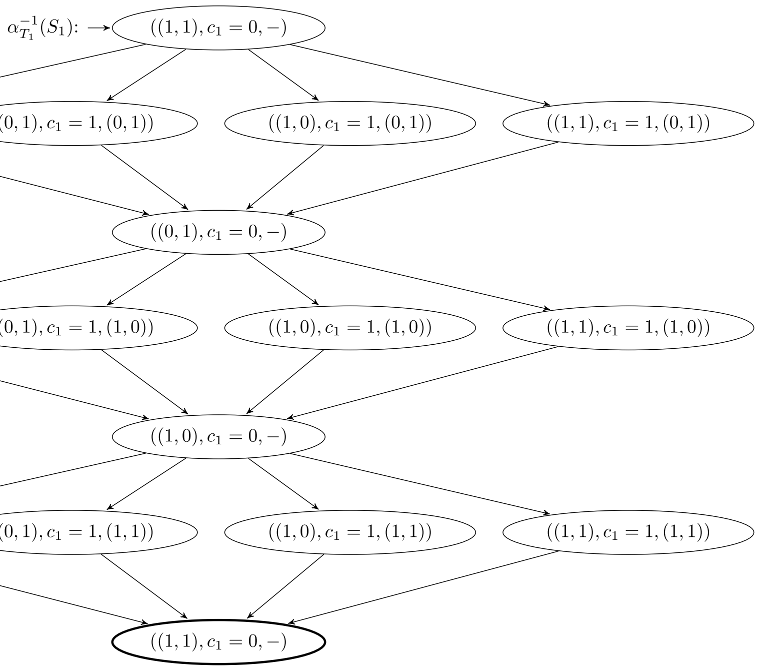
$\alpha_{T_1}^{-1}(S_1)$:
$((1,1), c_1 = 0, -)$
$((0,0), c_1 = 1, (0,1))$
$((0,1), c_1 = 1, (0,1))$
$((1,0), c_1 = 1, (0,1))$
$((1,1), c_1 = 1, (0,1))$
$((0,1), c_1 = 0, -)$
$((0,0), c_1 = 1, (1,0))$
$((0,1), c_1 = 1, (1,0))$
$((1,0), c_1 = 1, (1,0))$
$((1,1), c_1 = 1, (1,0))$
$((1,0), c_1 = 0, -)$
$((0,0), c_1 = 1, (1,1))$
$((0,1), c_1 = 1, (1,1))$
$((1,0), c_1 = 1, (1,1))$
$((1,1), c_1 = 1, (1,1))$
$((1,1), c_1 = 0, -)$

Fig. 17. nFOS $\alpha_{T_1}^{-1}(S_1)$ where S_1 is the nFOS of Fig. 16 and $T_1 = 2$.

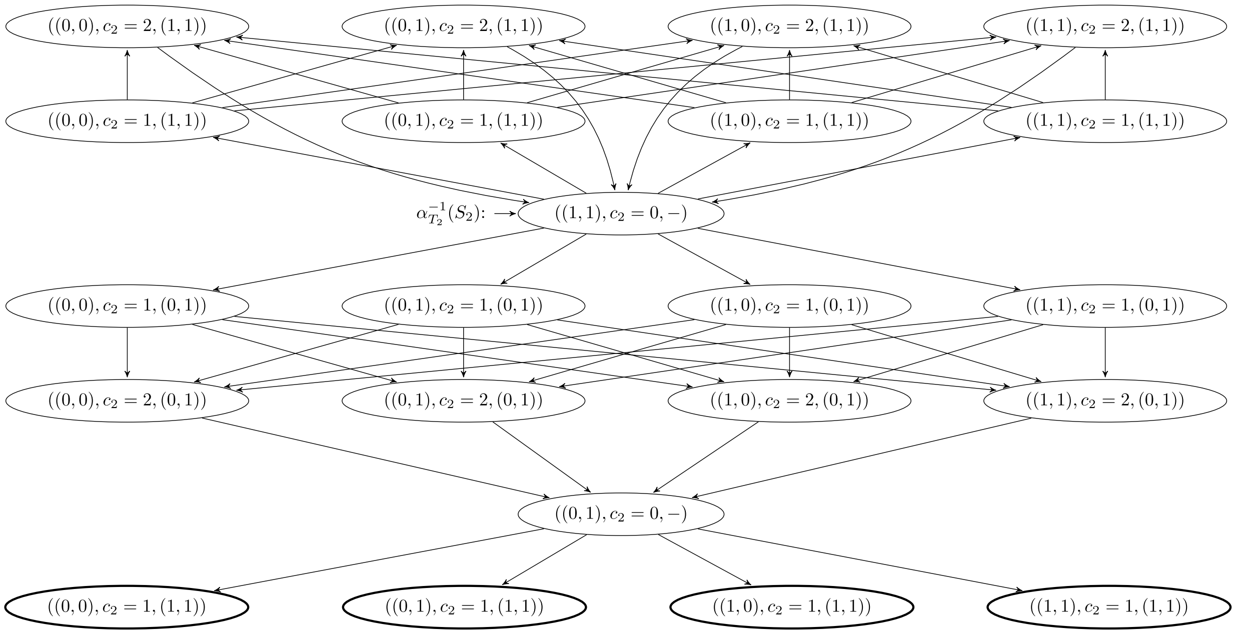
$((0,0), c_2 = 2, (1,1))$
$((0,1), c_2 = 2, (1,1))$
$((1,0), c_2 = 2, (1,1))$
$((1,1), c_2 = 2, (1,1))$
$((0,0), c_2 = 1, (1,1))$
$((0,1), c_2 = 1, (1,1))$
$((1,0), c_2 = 1, (1,1))$
$((1,1), c_2 = 1, (1,1))$
$\alpha_{T_2}^{-1}(S_2)$:
$((1,1), c_2 = 0, -)$
$((0,0), c_2 = 1, (0,1))$
$((0,1), c_2 = 1, (0,1))$
$((1,0), c_2 = 1, (0,1))$
$((1,1), c_2 = 1, (0,1))$
$((0,0), c_2 = 2, (0,1))$
$((0,1), c_2 = 2, (0,1))$
$((1,0), c_2 = 2, (0,1))$
$((1,1), c_2 = 2, (0,1))$
$((0,1), c_2 = 0, -)$
$((0,0), c_2 = 1, (1,1))$
$((0,1), c_2 = 1, (1,1))$
$((1,0), c_2 = 1, (1,1))$
$((1,1), c_2 = 1, (1,1))$

Fig. 18. nFOS $\alpha_{T_2}^{-1}(S_2)$ where S_2 is the nFOS of Fig. 16 and $T_2 = 3$.

consider the behavior $(1,1)^\omega \in [\![S_2]\!]_o$. Because $a_{T_2}(S^\#)$ does not loop in the initial state, it cannot generate the observable behavior $(1,1)^\omega$. Therefore, we conclude that the views S_1 and S_2 are inconsistent.

7. Conclusions and future work

In the presence of large and complex systems, multi-view modeling has become a predominant design principle. In order to understand and describe the versatile aspects of a system under development, multi-view modeling methods use several views that serve as partial models of the system. Then, a crucial issue in multi-view modeling, is ensuring consistency among the views, as these are developed by designers coming from various disciplines and using heterogeneous formalisms,

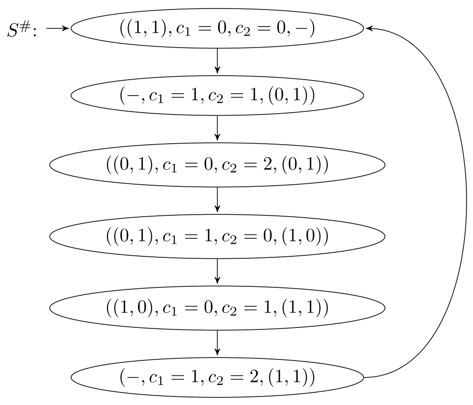
$S^{\#}$:
$((1,1), c_1 = 0, c_2 = 0, -)$
$(-, c_1 = 1, c_2 = 1, (0,1))$
$((0,1), c_1 = 0, c_2 = 2, (0,1))$
$((0,1), c_1 = 1, c_2 = 0, (1,0))$
$((1,0), c_1 = 0, c_2 = 1, (1,1))$
$(-, c_1 = 1, c_2 = 2, (1,1))$

Fig. 19. nFOS $S^{\#} = \alpha_{T_1}^{-1}(S_1) \otimes \alpha_{T_2}^{-1}(S_2)$.

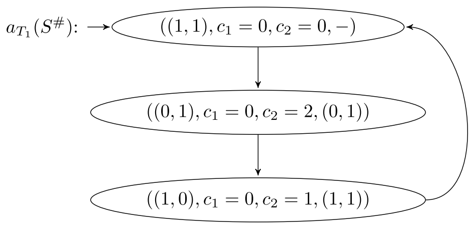
$a_{T_1}(S^{\#})$:
$((1,1), c_1 = 0, c_2 = 0, -)$
$((0,1), c_1 = 0, c_2 = 2, (0,1))$
$((1,0), c_1 = 0, c_2 = 1, (1,1))$

Fig. 20. nFOS $\alpha_{T_1}(S^{\#})$ for $T_1 = 2$.

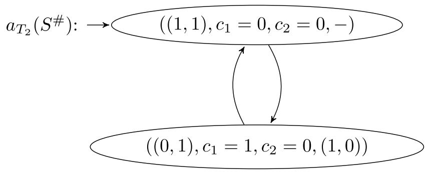
$a_{T_2}(S^{\#})$:
$((1,1), c_1 = 0, c_2 = 0, -)$
$((0,1), c_1 = 1, c_2 = 0, (1,0))$

Fig. 21. nFOS $\alpha_{T_2}(S^{\#})$ for $T_2 = 3$.

languages, and tools. In this work we studied the view consistency problem within the formal framework of [33,32], but for a different type of abstraction functions than those investigated previously. In particular, we studied view consistency w.r.t. periodic sampling abstractions. We considered two different settings for modeling views, ω-regular languages (Büchi automata) and symbolic transition systems, and provided complete solutions to the view consistency problems arising in these settings. In the process we answered several of the questions left open in the earlier version of this work [27]. We also introduced an important new result (Theorem 5) which complements the abstract framework of [33,32]. The result is a generic necessary and sufficient condition for view consistency which can be used to derive algorithms for multiple instantiations of the abstract framework, *e.g.*, those considered in this paper in particular.

Several problems remain open, and there are several directions for future work. Specific to the results in this paper is the open question of finding a more efficient method to solve Problem 5, than the brute-force method proposed here. Other future research directions include considering abstraction functions other than periodic samplings, including non-periodic and event-triggered samplings. Another direction is to consider other types of models, including non-purely-discrete models, such as timed or hybrid automata, as well as heterogeneous instantiations of the multi-view modeling framework, *e.g.*, where some views are discrete, some continuous, some hybrid, and so on. In addition to these theoretical directions, ongoing work includes an implementation of the current framework and experimentation with case studies.

References

[1] V. Amaral, C. Hardebolle, H. Vangheluwe, L. Lengyel, P. Bunus, Recent advances in multi-paradigm modeling, Electron. Commun. EASST 50 (2011) 1–10.

[2] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, C. Sofronis, Multiple viewpoint contract-based specification and design, in: FMCO, Springer, 2008, pp. 200–225.

[3] A. Bhave, B.H. Krogh, D. Garlan, B.R. Schmerl, View consistency in architectures for cyber-physical systems, in: ICCPS, IEEE, 2011, pp. 151–160.

[4] X. Blanc, I. Mounier, A. Mougenot, T. Mens, Detecting model inconsistency through operation-based model construction, in: ICSE, ACM, 2008, pp. 511–520.

[5] D. Broman, E.A. Lee, S. Tripakis, M. Törngren, Viewpoints, formalisms, languages, and tools for cyber-physical systems, in: 6th Intl. Workshop on Multi-Paradigm Modeling (MPM'12), 2012.

[6] M. Broy, Software and system modeling: structured multi-view modeling, specification, design and implementation, in: Conquering Complexity, Springer, 2012, pp. 309–372.
[7] P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, P. Niebert, From Simulink to SCADE/Lustre to TTA: a layered approach for distributed embedded applications, in: Proceedings of the 2003 ACM SIGPLAN Conf. on Languages, Compilers, and Tools for Embedded Systems (LCTES'03), ACM, June 2003, pp. 153–162.
[8] L. de Alfaro, T.A. Henzinger, Interface theories for component-based design, in: EMSOFT, in: LNCS, vol. 2211, Springer, 2001, pp. 148–165.
[9] R.M. Dijkman, Consistency in Multi-Viewpoint Architectural Design, PhD thesis, University of Twente, 2006.
[10] S. Easterbrook, M. Chechik, A framework for multi-valued reasoning over inconsistent viewpoints, in: ICSE, IEEE, 2001, pp. 411–420.
[11] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, B. Nuseibeh, Inconsistency handling in multiperspective specifications, IEEE TSE 20 (8) (1994) 569–578.
[12] S. Getir, L. Grunske, C.K. Bernasko, V. Käfer, T. Sanwald, M. Tichy, Cowolf – a generic framework for multi-view co-evolution and evaluation of models, in: ICMT, in: LNCS, vol. 9152, Springer, 2015, pp. 34–40.
[13] T.A. Henzinger, D. Nickovic, Independent implementability of viewpoints, in: Monterey Workshop, in: LNCS, vol. 7539, Springer, 2012, pp. 380–395.
[14] G.J. Holzmann, The model checker SPIN, IEEE Trans. Softw. Eng. 23 (5) (1997) 279–295.
[15] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1990.
[16] ISO/IEC/IEEE 42010:2011, Systems and software engineering – architecture description, the latest edition of the original IEEE Std 1471:2000, Recommended Practice for Architectural Description of Software-intensive Systems, IEEE and ISO, 2011.
[17] D. Jackson, Structuring Z Specifications with Views, Technical Report CMU-CS-94-126, CMU, 1994.
[18] E.K. Jackson, T. Levendovszky, D. Balasubramanian, Automatically reasoning about metamodeling, Softw. Syst. Model. 14 (1) (2015) 271–285.
[19] E.K. Jackson, J. Sztipanovits, Formalizing the structural semantics of domain-specific modeling languages, Softw. Syst. Model. 8 (4) (2009) 451–478.
[20] J. Kienzle, W.A. Abed, J. Klein, Aspect-oriented multi-view modeling, in: Aspect-Oriented Software Development, ACM, 2009, pp. 87–98.
[21] H. Lewis, C. Papadimitriou, Elements of the Theory of Computation, Prentice-Hall, 1997.
[22] F.J. Lucas, F. Molina, J.A.T. Álvarez, A systematic review of UML model consistency management, Inf. Softw. Technol. 51 (12) (2009) 1631–1645.
[23] S. Maoz, J.O. Ringert, B. Rumpe, Semantically configurable consistency analysis for class and object diagrams, CoRR, arXiv:1409.2313, 2014.
[24] S. Maoz, J.O. Ringert, B. Rumpe, Verifying component and connector models against crosscutting structural views, in: ICSE, ACM, 2014, pp. 95–105.
[25] K.L. McMillan, Symbolic Model Checking, Kluwer, 1993.
[26] M. Persson, M. Törngren, A. Qamar, J. Westman, M. Biehl, S. Tripakis, H. Vangheluwe, J. Denil, A characterization of integrated multi-view modeling in the context of embedded and cyber-physical systems, in: EMSOFT, IEEE, 2013, pp. 1–10.
[27] M. Pittou, S. Tripakis, Checking multi-view consistency of discrete systems with respect to periodic sampling abstractions, in: FACS, in: LNCS, vol. 10231, 2016, pp. 73–91.
[28] M. Pittou, S. Tripakis, Multi-view consistency for infinitary regular languages, in: SAMOS, IEEE, 2016, pp. 148–155.
[29] A. Rajhans, B.H. Krogh, Heterogeneous verification of cyber-physical systems using behavior relations, in: HSCC, ACM, 2012, pp. 35–44.
[30] A. Rajhans, B.H. Krogh, Compositional heterogeneous abstraction, in: HSCC, ACM, 2013, pp. 253–262.
[31] H. Rasch, H. Wehrheim, Checking consistency in UML diagrams: classes and state machines, in: Formal Methods for Open Object-Based Distributed Systems, 6th IFIP WG 6.1 International Conference, FMOODS, 2003, pp. 229–243.
[32] J. Reineke, C. Stergiou, S. Tripakis, Basic problems in multi-view modeling, Softw. Syst. Model. (Dec. 2017) 1–35.
[33] J. Reineke, S. Tripakis, Basic problems in multi-view modeling, in: TACAS, in: LNCS, vol. 8413, Springer, 2014, pp. 217–232.
[34] A.A. Shah, A.A. Kerzhner, D. Schaefer, C.J.J. Paredis, Multi-view modeling to support embedded systems engineering in SysML, in: Graph Transformations and Model-Driven Engineering, in: LNCS, vol. 5765, Springer, 2010, pp. 580–601.
[35] A.P. Sistla, M.Y. Vardi, P. Wolper, The complementation problem for Büchi automata with applications to temporal logic, Theor. Comput. Sci. 49 (1987) 217–237.
[36] G. Spanoudakis, A. Finkelstein, Reconciling requirements: a method for managing interference, inconsistency and conflict, Ann. Softw. Eng. 3 (1996) 433–457, Special Issue on Software Requirements Engineering.
[37] W. Thomas, Automata on infinite objects, in: Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B), Elsevier Science Publishers, 1990, pp. 133–192.
[38] S. Tripakis, Compositionality in the science of system design, Proc. IEEE 104 (5) (May 2016).
[39] R. von Hanxleden, E.A. Lee, C. Motika, H. Fuhrmann, Multi-view modeling and pragmatics in 2020, in: 17th Intl. Monterey Workshop, in: LNCS, vol. 7539, Springer, 2012.
[40] S. Warshall, A theorem on boolean matrices, J. ACM 9 (1) (1962) 11–12.
[41] X. Zhao, Q. Long, Z. Qiu, Model checking dynamic UML consistency, in: Formal Methods and Software Engineering, in: LNCS, vol. 4260, Springer, 2006, pp. 440–459.