Basic Problems in Multi-View Modeling

Jan Reineke
Stavros Tripakis

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-4
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-4.html

January 21, 2014

Acknowledgement

This research has been partially supported by the National Science Foundation and the Academy of Finland, via projects ExCAPE: Expeditions in Computer Augmented Program Engineering and COSMOI: Compositional System Modeling with Interfaces, by the Deutsche Forschungsgemeinschaft as part of the Transregional Collaborative Research Centre SFB/TR 14 AVACS, and by the centers TerraSwarm and iCyPhy (Industrial Cyber-Physical Systems) at UC Berkeley.

Basic Problems in Multi-View Modeling*

Jan Reineke
Saarland University

Stavros Tripakis
UC Berkeley and Aalto University

January 21, 2014

Abstract

Modeling all aspects of a complex system within a single model is a difficult, if not impossible, task. Multi-view modeling is a methodology where different aspects of the system are captured by different models, or *views*. A key question then is *consistency*: if different views of a system have some degree of overlap, how can we guarantee that they are consistent, i.e., that they do not contradict each other? In this paper we formulate this and other basic problems in multi-view modeling within an abstract formal framework. We then instantiate this framework in a discrete, finite-state system setting, and study how some key verification and synthesis problems can be solved in that setting.

1 Introduction

Real systems are usually complex objects, and grasping all the details of a system at the same time is often difficult. In addition, each of the various stakeholders in the system are concerned with different system aspects. For these reasons, modeling and design teams usually deal only with partial and incomplete *views* of a system, which are easier to manage separately. For example, when designing a digital circuit, architects may be concerned with general (boolean) functionality issues, while ignoring performance. Other stakeholders, however, may be concerned about timing aspects such as the delay of the critical path, which ultimately affects the clock rate at which the circuit can be run. Yet other stakeholders may be interested in a different aspect, namely, energy consumption of the circuit which affects battery life.

Modeling and simulation are often used to support system design. In this paper, when we talk about views, we refer concretely to the different *models* of a system that designers build. Such models may be useful as models of an *existing* system: the system exists, and a model is built in order to study the system. Then, the model is only a partial or incomplete view of the system, since it focuses on certain aspects and omits others. For example, an energy consumption model for an airplane ignores control, air dynamics, and other aspects. Models may also be used for a *system-to-be-built*: an energy consumption model as in the example above could be developed as part of the design process, even before the airplane is built.

For large systems, each aspect of the system is typically designed by a dedicated design team. These teams often use different modeling languages and tools to capture different views, which is generally referred to as *multi-view modeling* (MVM). MVM presents a number of challenges, such as the crucial issue of *consistency*: if different views of the system are captured by different models, and these models have some degree of overlap, how can we guarantee that the models are consistent, i.e., that they do not contradict each other? Understanding the precise meaning of such questions, and developing techniques to answer them, ideally fully automatically, is the main goal of this paper.

Toward this goal, we begin in Section 2 by introducing an example of simple 3-dimensional structure modeling. Even though our focus is on dynamic behaviors, we will still use this static system as an illustrative

*This research is partially supported by the National Science Foundation and the Academy of Finland, via projects *ExCAPE: Expeditions in Computer Augmented Program Engineering* and *COSMOI: Compositional System Modeling with Interfaces*, by the Deutsche Forschungsgemeinschaft as part of the Transregional Collaborative Research Centre SFB/TR 14 *AVACS*, and by the centers *TerraSwarm* and *iCyPhy (Industrial Cyber-Physical Systems)* at UC Berkeley.

running example to demonstrate the salient concepts of our formal MVM framework. The latter is itself presented in Section 3. The main concepts are as follows: (1) views can be derived from systems using abstraction functions, which map system behaviors to view behaviors; (2) conformance formalizes how "faithful" a view is to a system; (3) consistency of a set of views is defined as existence of a witness system to which all views conform; (4) view reduction allows to "optimize" views by using the information contained in other views; (5) orthogonality captures independence between views.

The framework proposed in Section 3 is abstract, in the sense that it does not refer to specific notions of behaviors, neither to concrete representations of systems and views. In the rest of the paper we instantiate this abstract framework for the case of discrete systems. The latter, defined in Section 4, are finite-state symbolic transition systems consisting of a set of state variables, a predicate over the state variables characterizing the set of initial states, and a predicate characterizing the transition relation.

In Section 5 we study projections as abstraction functions for discrete systems. Fully-observable systems, where all variables are observable, are not closed under projection, therefore we also consider systems with internal (unobservable) variables. We show how to effectively solve a number of verification and synthesis problems on discrete systems and views, including view conformance and consistency checking.

2 Running Example: 3D Objects

To illustrate the concept of views we introduce a running example that we will refer to throughout the paper.

Consider the 3D structure shown at the left of Figure 1. It can be modeled as a set of points in a $4 \times 4 \times 4$ space, each point (x, y, z) representing a "box" appearing at coordinate (x, y, z), for $x, y, z \in \{1, 2, 3, 4\}$. The object shown to the left of the figure contains 16 such boxes, and the corresponding set contains 16 points.

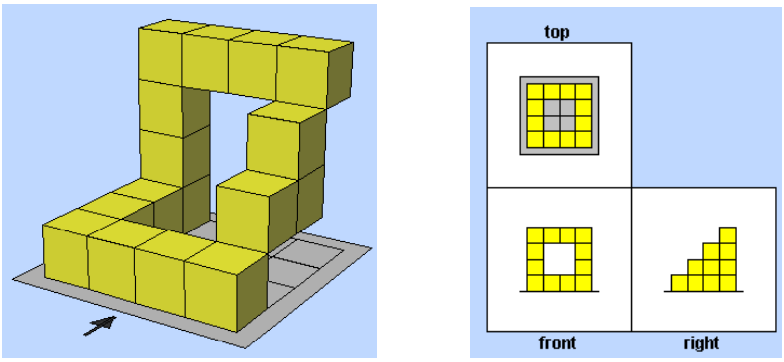

Figure 1: A 3D structure (left) and 3 views of it (right) – image produced using this tool: http://www.fi.uu.nl/toepassingen/02015/toepassing_wisweb.en.html.

Three views of the object are shown to the right of the figure: a top view, a front view, and a side view. These views can be formalized as 2D projections. Let S be the set of points representing the 3D object. Then the three views can be formalized as sets $V_{\text{top}}, V_{\text{front}}, V_{\text{side}}$, where:

$$V_{top} = \{(x, y) \mid \exists z : (x, y, z) \in S\}, \quad V_{front} = \{(x, z) \mid \exists y : (x, y, z) \in S\}, \quad V_{side} = \{(y, z) \mid \exists x : (x, y, z) \in S\}.$$

The above projections can be seen as *abstractions* of S. In fact, they are generally *strict* abstractions in the sense that some information about S is lost during the abstraction. In the case of Figure 1, e.g., the same views would be obtained if one were to add to the object the missing boxes so that no box under the "staircase structure" hangs in the air.

3 Views: a Formalization

3.1 Systems

We define a *system* semantically, as a set of behaviors. As in [15], there is no restriction on the type of behaviors: they could be discrete traces, continuous trajectories, hybrid traces, or something else. We only assume given a domain of possible behaviors, $\mathcal{U}$. Then, a *system S over domain of behaviors $\mathcal{U}$* is a subset of $\mathcal{U}$: $S \subseteq \mathcal{U}$.

3.2 View domains

A view is intuitively an "incomplete picture" of a system. It can be incomplete in different ways:

- Some behaviors may be missing from the view, i.e., the view may contain only a *subset* of system behaviors. (As we shall see when we discuss conformance, the view may also be a *superset*.)
- Some parts of a behavior itself may be missing in the view. E.g., if the behavior refers to a state vector with, say, 10 state variables, the view could refer only to 2 state variables. In this case the view can be seen as a *projection*.
- More generally, the view may be obtained by some other kind of *transformation* (not necessarily a projection) to behaviors. E.g., the original system behaviors may contain temperature as a state variable, but the view only contains temperature averages over some period of time.

From the above discussion, it appears that: semantically, views can be formalized as sets of behaviors, just like systems are. However, because of projections or other transformations, the domain of behaviors of a view is not necessarily the same as the domain of system behaviors, $\mathcal{U}$. Therefore, we let $\mathcal{D}_i$ be the domain of behaviors of view i (there can be more than one views, hence the subscript i). When we refer to a general view domain, we drop the subscript and simply write $\mathcal{D}$.

In the case of our running example, $\mathcal{U} = \{1,2,3,4\}^3$, and $\mathcal{D}_{top} = \mathcal{D}_{front} = \mathcal{D}_{side} = \{1,2,3,4\}^2$.

3.3 Views

A view is a set of behaviors over a given view domain. That is, a *view V over view domain $\mathcal{D}$* is defined to be a subset of $\mathcal{D}$: $V \subseteq \mathcal{D}$.

3.4 Abstraction functions

Given a domain of behaviors $\mathcal{U}$ and a view domain $\mathcal{D}$, we would like to relate systems over $\mathcal{U}$ and views over $\mathcal{D}$. In order to do this, we will first introduce *abstraction functions*, which map behaviors from $\mathcal{U}$ to $\mathcal{D}$. An *abstraction function from $\mathcal{U}$ to $\mathcal{D}$* is defined to be a mapping $a : \mathcal{U} \to \mathcal{D}$. Abstraction functions can be projections or other types of transformations, as discussed above.

In the case of our running example, the abstraction functions $a_{top}, a_{front}, a_{side}$ are 3D-to-2D projections on the corresponding planes.

An abstraction function a can be naturally "lifted" from behaviors to systems. If $S \subseteq \mathcal{U}$, then $a(S)$ is defined to be: $a(S) := \{a(\sigma) \mid \sigma \in S\}$. Note that $a(S) \subseteq \mathcal{D}$, therefore, $a(S)$ is a view over $\mathcal{D}$.

3.5 Conformance

Given system $S \subseteq \mathcal{U}$, view $V \subseteq \mathcal{D}$, and abstraction function $a : \mathcal{U} \to \mathcal{D}$, we say that *$V$ is a complete view of S w.r.t. a* if $V = a(S)$. The notion of complete view is a reasonable way of capturing how "faithful" a given view is to a certain system. For example, if S is an object containing two boxes, $S = \{(1,1,1),(2,2,2)\}$ and a_{top} is the top view, then $V_1 = \{(1,1),(2,2)\}$ is complete w.r.t. a_{top}, whereas $V_2 = \{(2,2)\}$ and $V_3 = \{(1,1),(2,2),(3,3)\}$ are not complete.

But faithfulness need not always require a strict equality as in the condition $V = a(S)$. Depending on the usage one makes of a view, weaker conditions may be appropriate. Because of this, we introduce the notion of *conformance*. Conformance is defined with respect to a partial order $\sqsupseteq$ on the set of all views over view domain $\mathcal{D}$. That is, $\sqsupseteq$ is a partial order on $2^{\mathcal{D}}$, the powerset of $\mathcal{D}$. Then, we say that V *conforms to* S *w.r.t.* a *and* $\sqsupseteq$, denoted $V \sqsupseteq_a S$, if $V \sqsupseteq a(S)$.

For example, if one uses the top view to decide whether it is safe to drop a box to the floor without touching another box during landing, then a view that safely approximates the set of free (x, y) positions could be acceptable. In this case, the partial order $\sqsupseteq$ is $\supseteq$, i.e., conformance is defined as $V \supseteq a_{top}(S)$. Indeed, dropping a box to $(x, y) \notin V$ would be safe, since $(x, y) \notin V$ and $V \supseteq a_{top}(S)$ imply $(x, y) \notin a_{top}(S)$. In another scenario, it may be more appropriate to require that the view *under*-approximates $a(S)$, thus *over*-approximates the set of free (x, y) positions. For example, if one uses the top view to decide whether it is safe to drop an object so that it does *not* hit the floor, then it is more appropriate to define conformance as $V \subseteq a_{top}(S)$. In this case, $\sqsupseteq$ is $\subseteq$.

This definition of conformance is also able to capture the fact that views, in addition to containing projected/abstracted behaviors, may also contain a *subset* of the original system behaviors, as discussed above.

3.6 An alternative formalization: starting with conformance

In the way we formalized things so far, we started with an abstraction function a and a partial order $\sqsupseteq$, and defined the conformance relation $\sqsupseteq_a$ with respect to those. As an alternative, we can start with a conformance relation $\models \subseteq 2^{\mathcal{D}} \times 2^{\mathcal{U}}$, which relates a view V and a system S, i.e., $V \models S$, and derive an abstraction function a. We can do this provided that $\models$ satisfies the conditions described below, and that the domain of views equipped with $\sqsupseteq$, denoted $(2^{\mathcal{D}}, \sqsupseteq)$, forms a complete lattice. Let $\bigsqcap$ denote the greatest lower bound in this lattice. Note that the interpretation of the lattice is that the smaller an element the more accurate it is, and $x \sqsupseteq y$ says that y is smaller than x. Therefore, when $\sqsupseteq$ is $\supseteq$, top is $\mathcal{D}$, bottom is $\emptyset$, and $\bigsqcap$ is $\bigcap$. When $\sqsupseteq$ is $\subseteq$, $\bigsqcap$ is $\bigcup$. Then, $\models$ induces an abstraction function a defined as follows:

$$a_{\models}(S) := \bigsqcap \{V \subseteq \mathcal{D} \mid V \models S\}.$$

For this to work, however, we need $\models$ to have the two following properties:

1. (monotonicity) $V_1 \models S \wedge V_2 \sqsupseteq V_1 \Rightarrow V_2 \models S$.
2. (conformance preserved by $\bigsqcap$) $\forall W \subseteq 2^{\mathcal{D}} : (\forall V \in W : V \models S) \Rightarrow (\bigsqcap W) \models S$.

Condition 1 says that if V_1 conforms to S then any view greater than V_1 also conforms to S. Condition 2 says that if a set of views all conform to a system S, then their greatest lower bound also conforms to S. Any relation $\sqsupseteq_a$ defined by an abstraction function a and an order $\sqsupseteq$ forming a complete lattice has these two properties by construction.

3.7 Consistency

It is often the case in practice that S is not available. Instead, only some views of S are available. The goal may be in fact to design S based on the views. This is typical in real-life projects, where every design team has its own view (and corresponding designs/models), whereas no team has a complete model for the entire system. In this scenario, the notion of consistency becomes particularly important, and we formalize it next.

Consider a set of views, $V_1, V_2, ..., V_n$, over view domains $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n$. For each view domain $\mathcal{D}_i$, consider given a conformance relation $\models_i$ (which could be derived from given abstraction function a_i and partial order $\sqsupseteq_i$, or defined as a primitive notion as explained above). We say that $V_1, V_2, ..., V_n$ *are consistent w.r.t.* $\models_1, \models_2, ..., \models_n$ if there exists a system S over $\mathcal{U}$ such that $\forall i = 1, ..., n : V_i \models_i S$. We call such a system S a *witness* to the consistency of $V_1, V_2, ..., V_n$. Clearly, if no such S exists, then one must conclude that the views are inconsistent, as there is no system from which these views could be derived. When $\sqsupseteq_i$ is $=$ for all i, i.e.,

when $V_i = a_i(S)$ for all i, we say that $V_1, ..., V_n$ are *strictly consistent*. Note that if $\sqsupseteq_i$ is $\supseteq$ for all i, then consistency trivially holds as the empty system is a witness, since $V_i \supseteq \emptyset = a_i(\emptyset)$ for all i. Also, if $\sqsupseteq_i$ is $\subseteq$ for all i and every a_i satisfies $a_i(\mathcal{U}) = \mathcal{D}_i$, then consistency trivially holds as the system $\mathcal{U}$ is a witness, since $V_i \subseteq \mathcal{D}_i = a_i(\mathcal{U})$ for all i.

In our 3D objects example, if V_{top} is non-empty but V_{side} is empty, then the two views are inconsistent w.r.t. strict conformance $V = a(S)$. A less trivial case is when $V_{top} = \{(1,1)\}$ and $V_{side} = \{(2,2)\}$. Again the two views are inconsistent (w.r.t. $=$): V_{top} asserts that some box must be in the column with (x, y) coordinates $(1, 1)$, but V_{side} implies that there is no box whose y coordinate is 1.

The last example may mislead to believe that consistency (w.r.t. $=$) is equivalent to "intersection of inverse projection of views being non-empty." This is not true. Even in the case where abstraction functions are projections, non-empty intersection of inverse projections is a necessary, but not a sufficient condition for consistency. To see this, consider views $V_{top} = \{(1,1),(3,3)\}$ and $V_{side} = \{(2,2),(1,2)\}$ in the context of our running example. These two views are inconsistent w.r.t. $=$. Yet the intersection of their inverse projections is non-empty, and equal to $\{(1,1,2)\}$.

Lemma 1. *Let $V_1, ..., V_n$ be a set of consistent views w.r.t. $=_{a_1}, ..., =_{a_n}$ and let S_1, S_2 be two witness systems to this. Then $S_1 \cup S_2$ is also a witness system.*

Proof. S_1 and S_2 being witness to consistency w.r.t. $=_{a_1}, ..., =_{a_n}$ means that $\forall i = 1, ..., n : a_i(S_1) = a_i(S_2) = V_i$. Consider some $i = 1, ..., n$. By the manner in which abstraction functions are lifted to sets, they distribute over set union, that is, $a_i(S_1 \cup S_2) = a_i(S_1) \cup a_i(S_2) = V_i \cup V_i = V_i$. □

3.8 View reduction

Given a set of views $V_1, ..., V_n$ of a system S, it may be possible to "reduce" each view V_i based on the information contained in the other views, and as a result obtain views $V'_1, ..., V'_n$ that are "more accurate" views of S. We use the term *reduction* inspired from similar work in abstract interpretation [5, 10].

For example, if we assume that conformance is defined as $V \supseteq a(S)$, then the views $V_{top} = \{(1,1),(3,3)\}$ and $V_{side} = \{(2,2),(1,2)\}$ can be reduced to $V'_{top} = \{(1,1)\}$ and $V'_{side} = \{(1,2)\}$. V'_{top} is still a valid top view, in the sense that for every system S, if both $V_{top} \supseteq a_{top}(S)$ and $V_{side} \supseteq a_{side}(S)$, then $V'_{top} \supseteq a_{top}(S)$. In addition, V'_{top} is more accurate than V_{top} in the sense that V'_{top} is a strict subset of V_{top}. Indeed, V_{top} does not contain the "bogus" square $(3,3)$ which cannot occur in S, as we learn from V_{side}.

Let us now define the notion of view reduction formally. First, given a conformance relation between views and systems, $\models \subseteq 2^{\mathcal{D}} \times 2^{\mathcal{U}}$, we define the concretization function $c_{\models}$ which, given a view V, returns the set of all systems which V conforms to:

$$c_{\models}(V) := \{S \subseteq \mathcal{U} \mid V \models S\} = \{S \subseteq \mathcal{U} \mid V \sqsupseteq a(S)\}.$$

Note that $V_1, ..., V_n$ are consistent w.r.t. $\models_1, ..., \models_n$ iff $\bigcap_{i=1}^{n} c_{\models_i}(V_i) \neq \emptyset$. Also observe that, by definition, $a_{\models}(S) \models S$. As a consequence, $S \in c_{\models}(a_{\models}(S))$ for all $S \subseteq \mathcal{U}$.

We next lift $a_{\models}$ to sets of systems. For this, we will again assume that $(2^{\mathcal{D}}, \sqsupseteq)$ forms a lattice, with $\bigsqcap$ denoting its greatest lower bound.[1] Then, if $\mathcal{S}$ is a set of systems over $\mathcal{U}$, we define $a_{\models}(\mathcal{S})$ to be the "most accurate" view that conforms to all systems in $\mathcal{S}$:

$$a_{\models}(\mathcal{S}) := \bigsqcap\{V \subseteq \mathcal{D} \mid c_{\models}(V) \supseteq \mathcal{S}\} = \bigsqcap\{V \subseteq \mathcal{D} \mid \forall S \in \mathcal{S} : V \models S\}.$$

Lemma 2. *The most accurate view that conforms to a set of systems $\mathcal{S}$ can also be determined from the individual systems' abstractions:*

$$a_{\models}(\mathcal{S}) = \bigsqcup\{a_{\models}(S) \mid S \in \mathcal{S}\}.$$

[1] Note that when $\sqsupseteq$ is a set-theoretic relation such as $\subseteq$ or $\supseteq$, this obviously holds and $\bigsqcap$ is $\bigcup$ or $\bigcap$. When $\sqsupseteq$ is $=$ then $(2^{\mathcal{D}}, =)$ is not a lattice, and the definition of view reduction given below does not apply. This is not a problem, as in that case we require views to be complete.

Proof. Let us first show that $\bigsqcup\{a_{\models}(S) \mid S \in \mathcal{S}\} \sqsubseteq a_{\models}(\mathcal{S})$. We do so by showing that $a_{\models}(S') \sqsubseteq a_{\models}(\mathcal{S})$ for all systems $S' \in \mathcal{S}$, which implies the above statement: Plugging in the definitions, we get $\bigsqcap\{V \subseteq \mathcal{V} \mid V \models S'\} \sqsubseteq \bigsqcap\{V \subseteq \mathcal{D} \mid \forall S \in \mathcal{S} : V \models S\}$. This holds, because if $A \supseteq B$, then $\bigsqcap A \sqsubseteq \bigsqcap B$.

To finish the proof, we still need to show $a_{\models}(\mathcal{S}) \sqsubseteq \bigsqcup\{a_{\models}(S) \mid S \in \mathcal{S}\}$. We know that for all systems S: $S \in c_{\models}(a_{\models}(S))$. Also, by monotonicity, "greater" views conform to "more" systems. Formally, $V \models S$ and $V' \sqsupseteq V$ imply $V' \models S$. As $\bigsqcup\{a_{\models}(S) \mid S \in \mathcal{S}\} \sqsupseteq a_{\models}(S)$ for all $S \in \mathcal{S}$, $\bigsqcup\{a_{\models}(S) \mid S \in \mathcal{S}\} \models S$ for all $S \in \mathcal{S}$. As $a(\mathcal{S})$ is the greatest lower bound over all V with $V \models S$ for all $S \in \mathcal{S}$, $\bigsqcup\{a_{\models}(S) \mid S \in \mathcal{S}\} \sqsupseteq a_{\models}(\mathcal{S})$. □

Given the above, and assuming n view domains with corresponding conformance relations, $(\mathcal{D}_1, \models_1), ..., (\mathcal{D}_n, \models_n)$, view reduction can be defined as follows:

$$reduce_i(V_1, V_2, ..., V_n) := a_{\models_i}\big(\bigcap_{i=1}^{n} c_{\models_i}(V_i)\big).$$

Lemma 3. *Reduction is a reductive operation, i.e., $V_i \sqsupseteq reduce_i(V_1, V_2, ..., V_n)$ for all i. The set of witnesses to the consistency of views $V_1, ..., V_n$ is invariant under reduction, i.e., $\bigcap_{i=1}^{n} c_{\models_i}(reduce_i(V_1, V_2, ..., V_n)) = \bigcap_{i=1}^{n} c_{\models_i}(V_i)$ for all i.*

Proof. By definition of $a_{\models}$, we have $a_{\models}(c_{\models}(V)) = \bigsqcap\{V' \subseteq \mathcal{D} \mid c_{\models}(V') \supseteq c_{\models}(V)\}$. Thus, $a_{\models}(c_{\models}(V))$ is a lower bound on $\{V' \subseteq \mathcal{D} \mid c_{\models}(V') \supseteq c_{\models}(V)\}$, which contains V. So $a_{\models}(c_{\models}(V)) \sqsubseteq V$. Also, observe that $a_{\models}$ is monotone, i.e., $\mathcal{S} \subseteq \mathcal{T}$ implies $a_{\models}(\mathcal{S}) \sqsubseteq a_{\models}(\mathcal{T})$. Clearly, $\bigcap_{i=1}^{n} c_{\models_i}(V_i) \subseteq c_{\models_{i'}}(V_{i'})$ for all $i' = 1...n$. So we have $reduce_{i'}(V_1, V_2, ..., V_n) = a_{\models}(\bigcap_{i=1}^{n} c_{\models_i}(V_i)) \sqsubseteq a_{\models}(c_{\models_{i'}}(V_{i'})) \sqsubseteq V_{i'}$ for all $i' = 1...n$.

We now turn to the second part of the lemma. Due to Lemma 2, $a_{\models}(\mathcal{S}) \sqsupseteq a_{\models}(S)$, if $S \in \mathcal{S}$. By definition, $c_{\models}(a_{\models}(\mathcal{S})) = \{S \subseteq \mathcal{U} \mid a_{\models}(\mathcal{S}) \sqsupseteq a_{\models}(S)\}$. Thus, we have $c_{\models}(a_{\models}(\mathcal{S})) \supseteq \mathcal{S}$. Using this and the definition of $reduce_i$ we get:

$$\begin{array}{ll} & \bigcap_{i=1}^{n} c_{\models_i}(reduce_i(V_1, V_2, ..., V_n)) \\ \overset{\text{Def. } reduce_i}{=} & \bigcap_{i=1}^{n} c_{\models_i}(a_{\models_i}(\bigcap_{i=1}^{n} c_{\models_i}(V_i))) \\ \overset{c_{\models}(a_{\models}(\mathcal{S})) \supseteq \mathcal{S}}{\supseteq} & \bigcap_{i=1}^{n} (\bigcap_{i=1}^{n} c_{\models_i}(V_i)) = \bigcap_{i=1}^{n} c_{\models_i}(V_i) \end{array}$$

By the first part of the lemma, we have $reduce_i(V_1, V_2, ..., V_n) \sqsubseteq V_i$ for all i. As $c_{\models_i}$ is monotone, this implies $c_{\models_i}(reduce_i(V_1, V_2, ..., V_n)) \subseteq c_{\models_i}(V_i)$ for all i, which trivially implies $\bigcap_{i=1}^{n} c_{\models_i}(reduce_i(V_1, V_2, ..., V_n)) \subseteq \bigcap_{i=1}^{n} c_{\models_i}(V_i)$. □

The second part of the lemma implies that reduction is idempotent, i.e., for all i: $reduce_i(V_1, ..., V_n) = reduce_i(V'_1, ..., V'_n)$, where $V'_i = reduce_i(V_1, V_2, ..., V_n)$.

3.9 Orthogonality

In some fortunate cases different aspects of a system are independent of each other. Intuitively, what this means is that each aspect can be defined separately without the need for communication between development teams to avoid inconsistencies.

Formally, we say that view domains $\mathcal{D}_1, ..., \mathcal{D}_n$ are *orthogonal* if all sets of non-empty views $V_1, ..., V_n$ from these view domains are mutually irreducible, i.e., if $reduce_i(V_1, ..., V_n) = V_i$ for all $i = 1, ..., n$. The view domains from our example of 3D objects, capturing projections onto two dimensions, are *not* orthogonal, as the reduction example involving the domains shows. On the other hand, view domains corresponding to the projection onto individual dimensions would indeed be orthogonal to each other.

Alternatively, orthogonal view domains can be defined by requiring that all sets of non-empty views $V_1, ..., V_n$ from these domains are consistent w.r.t. $=$.

The following lemma shows that the two definitions of orthogonal domains are equivalent, if we assume that conformance is defined based on abstraction functions and the superset and equality relations as the partial orders on views.

Lemma 4. *Given non-empty views $V_1, ..., V_n$, the following statements are equivalent:*

1. $V_1, ..., V_n$ *are consistent w.r.t.* $=_{a_1}, ..., =_{a_n}$.
2. $V_1, ..., V_n$ *are mutually irreducible w.r.t.* $\supseteq_{a_1}, ..., \supseteq_{a_n}$.
3. $V_1, ..., V_n$ *are mutually irreducible w.r.t.* $\subseteq_{a_1}, ..., \subseteq_{a_n}$.

Proof. We prove equivalence of (1) and (2). Equivalence between (1) and (3) can be shown analogously. (1) $\rightarrow$ (2): Since $V_1, ..., V_n$ are consistent w.r.t. $=_{a_1}, ..., =_{a_n}$, there must be a witness system $S^* \subseteq \mathcal{U}$, such that $\forall j = 1, ..., n : a_j(S^*) = V_j$. Therefore, we also have: $\forall i = 1, ..., n : V_j \supseteq a_i(S^*)$, which means that S^* is in the concretization w.r.t. the conformance $\models_i = \supseteq_{a_i}$ of each view V_i, i.e., $\forall i = 1, ..., n : S^* \in c_{\models_i}(V_i)$. Therefore, $S^* \in \bigcap_{i=1}^n c_{\models_i}(V_i)$. By Lemma 2, $reduce_i(V_1, ..., V_n) = \bigsqcup\{a_i(S) \mid S \in \bigcap_{i=1}^n c_{\models_i}(V_i)\}$. Since $S^* \in \bigcap_{i=1}^n c_{\models_i}(V_i)$, we have $reduce_i(V_1, ..., V_n) \sqsupseteq a_i(S^*) = V_i$. By Lemma 3, $V_i \sqsupseteq reduce_i(V_1, ..., V_n)$. Since $\sqsupseteq$ is a partial order, it is antisymmetric, therefore, $reduce_i(V_1, ..., V_n) \sqsupseteq V_i$ and $V_i \sqsupseteq reduce_i(V_1, ..., V_n)$ imply $reduce_i(V_1, ..., V_n) = V_i$. Note that for this part of the proof we used a generic partial order $\sqsupseteq$, and did not have to use the assumption that reduction is defined w.r.t. abstraction functions lifted to sets of behaviors and the set-theoretic order $\supseteq$.

(2) $\rightarrow$ (1):

$$
\begin{aligned}
reduce_i(V_1, ..., V_n) &\overset{\text{by definition}}{=} a_{\models_i}\big(\bigcap_{i=1}^n c_{\models_i}(V_i)\big) \\
&\overset{\text{by Lemma 2}}{=} \bigsqcup\{a_i(S) \mid S \in \bigcap_{i=1}^n c_{\models_i}(V_i)\} \\
&\overset{\text{because } \sqsupseteq \text{ is } \supseteq}{=} \bigcup\{a_i(S) \mid S \in \bigcap_{i=1}^n c_{\models_i}(V_i)\} \\
&\overset{\text{because } a_i \text{ distributes over } \cup}{=} a_i(\bigcup\{S \mid S \in \bigcap_{i=1}^n c_{\models_i}(V_i)\}).
\end{aligned}
$$

As by our assumption $V_1, ..., V_n$ are mutually irreducible, we have $V_i = reduce_i(V_1, ..., V_n) = a_i(\bigcup\{S \mid S \in \bigcap_{i=1}^n c_{\models_i}(V_i)\})$ for all i, and so $\bigcup\{S \mid S \in \bigcap_{i=1}^n c_{\models_i}(V_i)\}$ is a witness to the consistency of $V_1, ..., V_n$ w.r.t. $=$. □

A system $S \subseteq \mathcal{U}$ is called *view definable* w.r.t. $\models_1, ..., \models_n$ if there exist views $V_1 \subseteq \mathcal{D}_1, ..., V_n \subseteq \mathcal{D}_n$, such that $c_{\models_1}(V_1) \cap \cdots \cap c_{\models_n}(V_n) = \{S\}$. In the example of 3D objects, with 2D projections, the empty object $S = \{\}$ is view definable, as it is defined by the empty views. Similarly, all objects $S_{i,j,k} = \{(i, j, k)\}$ are view definable. Note that a general cube is not view definable, as there are other objects (e.g., a hollow cube) which have the same 2D projections.

3.10 Verification and synthesis problems related to views

View conformance checking: given (concrete representation of) system S, view V, and a certain conformance relation, does V conform to S?

View synthesis: given system S and abstraction function a, synthesize (concrete representation of) $a(S)$. Alternatively, given S and conformance relation $\models$, construct smallest view V such that $V \models S$, that is, construct $a_{\models}(S)$.

View consistency checking: given views $V_1, ..., V_n$ and conformance relations $\models_1, ..., \models_n$, check whether $V_1, ..., V_n$ are consistent w.r.t. $\models_1, ..., \models_n$.

System synthesis from views: given consistent views $V_1, ..., V_n$ and conformance relations $\models_1, ..., \models_n$, construct a system S such that for all i, $V_i \models_i S$.

View reduction: given views $V_1, ..., V_n$ compute $reduce_i(V_1, V_2, ..., V_n)$ for given i.

4 Discrete Systems

Our goal in the rest of this paper is to instantiate the view framework developed in Section Section 3. We instantiate it for a class of discrete systems, and we also provide answers to some of the corresponding algorithmic problems.

We will consider finite-state discrete systems. The state space of such a system can be represented by a set of boolean variables, X, resulting in 2^n potential states, where $n = |X|$ is the size of X. A *state* s over X is a *valuation over* X, i.e., a function $s : X \to \mathbb{B}$, where $\mathbb{B} := \{0, 1\}$ is the set of booleans. For convenience, we sometimes consider other finite domains with the understanding that they can be encoded as booleans. A *behavior over* X is a finite or infinite sequence of states over X, $\sigma = s_0 s_1 s_2 \cdots$. $\mathcal{U}(X)$ denotes the set of all possible behaviors over X.

Semantically, a discrete system S over X is a set of behaviors over X, i.e., $S \subseteq \mathcal{U}(X)$. For computation, we need a concrete representation of discrete systems. We will start with a simple representation where all system variables are observable. We will then discuss limitations of this representation and consider an extension where the system can also have internal (unobservable) variables in addition to the observable ones.

4.1 Fully-observable discrete systems

A *fully-observable* discrete system (FOS) is represented concretely by a triple (X, θ, ϕ). X is the (finite) set of (boolean) variables. All variables in X are considered observable. θ is a boolean expression over X, characterizing the set of initial states of the system. Given state s, we write $\theta(s)$ to denote the fact that s satisfies θ, i.e., s is an initial state. ϕ is a boolean expression over $X \cup X'$, where X' is the set of primed copies of variables in X, $X' := \{x' \mid x \in X\}$, representing the next state variables, as usual. ϕ characterizes pairs of states (s, s'), each representing a transition of S, i.e., a move from state s to state s'. 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 system (X, θ, ϕ) is a finite or infinite sequence of states over X, $\sigma = s_0 s_1 s_2 \cdots$, such that $\theta(s_0)$ and $\forall i : \phi(s_i, s_{i+1})$, i.e., s_0 is an initial state and there is a transition from each s_i to s_{i+1} (if the latter exists). A state s is *reachable* if there is a finite behavior $s_0 s_1 \cdots s_n$, such that $s = s_n$.

We sometimes use $S = (X, \theta, \phi)$ to denote the concrete (syntactic) representation of discrete system S, and $[\![S]\!]$ to denote its semantics, i.e., its set of behaviors.

4.2 Projection (variable hiding)

Projection, or variable hiding, is a natural operation on systems, which can also serve as a basic abstraction function for views, as we shall see below. Here, we define projection and motivate the introduction of internal variables in the concrete representation of discrete systems.

Let s be a state over a set of variables X. Given subset $Y \subseteq X$, the projection function h_Y projects s onto the set of variables Y, that is, h_Y hides from s all variables in $X \setminus Y$. $h_Y(s)$ is defined to be the new state s' over Y, that is, the function $s' : Y \to \mathbb{B}$, such that $s'(x) = s(x)$ for all $x \in Y$.

Projection can be lifted to behaviors in the standard way. If $\sigma = s_0 s_1 s_2 \cdots$ is a behavior over X, then $h_Y(\sigma)$ is a behavior over Y defined by $h_Y(\sigma) := h_Y(s_0)h_Y(s_1)h_Y(s_2)\cdots$. Projection can also be lifted to systems. If S is a discrete system over X then $h_Y([\![S]\!]) := \{h_Y(\sigma) \mid \sigma \in [\![S]\!]\}$. We also denote $h_Y(s)$, $h_Y(\sigma)$, and $h_Y([\![S]\!])$, by $s[Y]$, $\sigma[Y]$, and $[\![S]\!][Y]$, respectively.

4.3 Non-closure properties

4.3.1 Non-closure under projection

The projection $h_Y([\![S]\!])$ is defined semantically, as a set of behaviors. It is natural to ask whether the syntactic representation of discrete systems is closed under projection. That is, is it true that for any $S = (X, \theta, \phi)$,

and $Y \subseteq X$, there exists $S' = (Y, \theta', \phi')$, such that $[\![S']\!] = h_Y([\![S]\!])$? This is not generally true:

Lemma 5. *There exists a FOS $S = (X, \theta, \phi)$, and $Y \subseteq X$, such that there is no FOS $S' = (Y, \theta', \phi')$, such that $[\![S']\!] = h_Y([\![S]\!])$.*

Proof. Consider the finite-state system $S = (\{x, y\}, x = 0 \wedge y = \mathit{true}, (x' = (x + 1) \bmod 5) \wedge (y' \leftrightarrow (x' = 0)))$, where $x \in \{0, 1, 2, 3, 4\}$ and $y \in \mathbb{B}$. Let $Y = \{y\}$. Then $h_Y([\![S]\!]) = \{y_0 y_1 \cdots \mid \forall i : y_i \Leftrightarrow i \bmod 5 = 0\}$. We claim that there is no $S' = (Y, \theta', \phi')$ such that $[\![S']\!] = h_Y([\![S]\!])$. The reason is that S' needs to count modulo five in order to produce the correct output. But S' has only one boolean variable y. □

As it turns out, we can check whether closure under projection holds for a given system: see Theorem 2 in Section 5.2.

4.3.2 Non-closure under union

Lemma 6. *Fully-observable systems over a set of variables X are not closed under union, i.e., there exist $S_1 = (X, \theta_1, \phi_1)$, $S_2 = (X, \theta_2, \phi_2)$ such that there is no $S = (X, \theta, \phi)$ such that $[\![S]\!] = [\![S_1]\!] \cup [\![S_2]\!]$.*

Proof. Consider as an example $S_1 = (\{x\}, \theta_1 = x, \phi_1 = x \wedge \neg x')$ and $S_2 = (\{x\}, \theta_2 = \neg x, \phi_1 = \neg x \wedge x')$. Both systems allow exactly one transition, from $x \mapsto \mathit{true}$ to $x \mapsto \mathit{false}$ and vice versa. A system that represents the union of S_1 and S_2 needs to include both transitions. Then, however, it also includes arbitrarily long behaviors alternating between $x \mapsto \mathit{true}$ and $x \mapsto \mathit{false}$. □

4.4 Discrete systems with internal variables

The above non-closure properties motivate us to study, in addition to fully-observable discrete systems, a generalization which extends them with a set of *internal*, *unobservable* state variables. Most practical modeling languages also allow the construction of models with both internal and observable state variables.

Accordingly, we extend the definition of a discrete system to be in general a tuple (X, Z, θ, ϕ), where X, Z are disjoint (finite) sets of variables. X models the observable and Z the internal variables. θ is a boolean expression over $X \cup Z$ and ϕ is a boolean expression over $X \cup Z \cup X' \cup Z'$. In such a system, we need to distinguish between behaviors, and observable behaviors. A behavior of a system $S = (X, Z, \theta, \phi)$ is a finite or infinite sequence σ over $X \cup Z$, defined as above. The observable behavior corresponding to σ is $h_X(\sigma)$, which is a behavior over X. From now on, $[\![S]\!]$ denotes the set of all behaviors (over $X \cup Z$) of S, and $[\![S]\!]_o$ denotes the set of observable behaviors (over X) of S.

Note that we allow Z to be empty. In that case, the system has no internal variables, i.e., it is a FOS. We will continue to represent a FOS by a triple $S = (X, \theta, \phi)$. A FOS S satisfies $[\![S]\!] = [\![S]\!]_o$.

4.5 Closure properties

We have already shown (Lemma 6) that FOS are not closed under union. They are however closed under intersection:

Lemma 7. *Given two FOS $S_1 = (X, \theta_1, \phi_1)$ and $S_2 = (X, \theta_2, \phi_2)$, a FOS S such that $[\![S]\!] = [\![S_1]\!] \cap [\![S_2]\!]$ is $S_1 \wedge S_2 = (X, \theta_1 \wedge \theta_2, \phi_1 \wedge \phi_2)$.*

Proof. Let $\sigma = s_0 \dots s_n$ be a behavior in $[\![S_1]\!] \cap [\![S_2]\!]$. Then $\theta_1(s_0)$ and $\theta_2(s_0)$ and $\phi_1(s_i, s_{i+1})$ and $\phi_2(s_i, s_{i+1})$ for all $i = 1, \dots, n-1$. Clearly, this is the case if and only if $(\theta_1 \wedge \theta_2)(s_0)$ and $(\phi_1 \wedge \phi_2)(s_i, s_{i+1})$ for all $i = 1, \dots, n-1$. Thus, $\sigma \in [\![S_1]\!] \cap [\![S_2]\!]$ if and only if $\sigma \in [\![S_1 \wedge S_2]\!]$. □

General discrete systems (with internal variables) are closed under intersection, union, as well as projection.

Lemma 8. *Let $S_1 = (X, Z_1, \theta_1, \phi_1)$ and $S_2 = (X, Z_2, \theta_2, \phi_2)$ be two systems, such that $Z_1 \cap Z_2 = \emptyset$. Let $Y \subseteq X$ and let z be a fresh variable not in $X \cup Z_1 \cup Z_2$. Let:*

$$S_\cap = (X, Z_1 \cup Z_2, \theta_1 \wedge \theta_2, \phi_1 \wedge \phi_2),$$
$$S_\cup = \big(X, Z_1 \cup Z_2 \cup \{z\}, (\theta_1 \wedge z) \vee (\theta_2 \wedge \neg z).(z \rightarrow \phi_1 \wedge z') \wedge (\neg z \rightarrow \phi_2 \wedge \neg z')\big),$$
$$S_\cup = \big(X, Z_1 \cup Z_2 \cup \{z_1, z_2\}, (z_1 \rightarrow \theta_1) \wedge (z_2 \rightarrow \theta_2).(z_1 \rightarrow \phi_1) \wedge (z_2 \rightarrow \phi_2) \wedge (\bigwedge_{i=1,2} z_i \leftrightarrow z_i')\big),$$
$$S_h = (Y, Z_1 \cup (X \setminus Y), \theta_1, \phi_1).$$

Then, $[\![S_\cap]\!]_o = [\![S_1]\!]_o \cap [\![S_2]\!]_o$, $[\![S_\cup]\!]_o = [\![S_1]\!]_o \cup [\![S_2]\!]_o$, and $[\![S_h]\!]_o = h_Y([\![S_1]\!]_o)$.

Proof. From Lemma 8 we know that $[\![S_\cap]\!] = [\![S_1]\!] \cap [\![S_2]\!]$. The observable behavior of $S_\cap$ is thus $[\![S_\cap]\!]_o = h_X([\![S_\cap]\!]) = h_X([\![S_1]\!] \cap [\![S_2]\!]) = h_X([\![S_1]\!]) \cap h_X([\![S_2]\!]) = [\![S_1]\!]_o \cap [\![S_2]\!]_o$.

To show that $[\![S_\cup]\!]_o \subseteq [\![S_1]\!]_o \cup [\![S_2]\!]_o$, consider an observable behavior σ_o of $S_\cup$. Then, there must be a behavior $\sigma = s_0 \dots s_n$ of $S_\cup$ with $\sigma_o = h_X(\sigma)$. By construction, either, $s_0(z) = \dots = s_n(z) = \mathit{true}$, or $s_0(z) = \dots = s_n(z) = \mathit{false}$. In the former case, $h_{X \cup Z_1}(\sigma)$ is a behavior of S_1, and thus $h_X(h_{X \cup Z_1}(\sigma)) = h_X(\sigma) = \sigma_o$ is an observable behavior of S_1. In the latter case, $h_X(h_{X \cup Z_2}(\sigma)) = h_X(\sigma) = \sigma_o$ is an observable behavior of S_2.

To show that $[\![S_\cup]\!]_o \supseteq [\![S_1]\!]_o \cup [\![S_2]\!]_o$, consider an observable behavior $\sigma_{o,1}$ of S_1. Then, there must be a behavior $\sigma_1 = s_1 \dots s_n$ of S_1 with $h_X(\sigma_1) = \sigma_{o,1}$. Extending all of the states of σ_1 with the mapping $z \mapsto \mathit{true}$ yields a behavior σ of $S_\cup$. Clearly, $h_X(\sigma) = \sigma_{o,1}$, so $\sigma_{o,1}$ is an observable behavior of $S_\cup$. A similar argument applies to observable behaviors of S_2.

Clearly, the behaviors of S_1 and S_h are equal. Consider a behavior σ_o in $h_Y([\![S_1]\!]_o)$. There must be a behavior σ of S_1 with $\sigma_o = h_Y(h_X(\sigma))$. As σ is also a behavior of S_h, $\sigma_o = h_Y(h_X(\sigma)) = h_Y(\sigma)$ is also an observable behavior of S_h. Consider a behavior σ_o in $[\![S_1]\!]_o$. There must be a behavior σ of S_h with $\sigma_o = h_Y(\sigma)$. As σ is also a behavior of S_1, $\sigma_o = h_Y(\sigma) = h_Y(h_X(\sigma))$ is also in $h_Y([\![S_1]\!]_o)$. □

5 Views of Finite-State Discrete Systems

Having defined discrete systems, we now turn to instantiating the view framework for such systems.

5.1 Discrete views, view domains, and abstraction functions

Discrete views are finite-state discrete systems. They are represented in general by tuples of the form (X, Z, θ, ϕ), and when $Z = \emptyset$, by triples of the form (X, θ, ϕ).

In this paper, we will study projection as the abstraction function for the discrete view framework. That is, a system will be a discrete system S over a set of observable variables X, and therefore the domain of system behaviors will be $\mathcal{U} = \mathcal{U}(X)$. A view will be a discrete system V over a subset of observable variables $Y \subseteq X$. Therefore, the view domain of V is $\mathcal{D} = \mathcal{U}(Y)$. Note that both S and V may have (each their own) internal variables.

Let $S = (X, Z, \theta, \phi)$ be a discrete system, $V = (Y, W, \theta', \phi')$ be a discrete view, with $Y \subseteq X$, and $\sqsupseteq$ be one of the orders $=$, $\subseteq$, or $\supseteq$. To make notation lighter, we will write $V \sqsupseteq h_Y(S)$ instead of $[\![V]\!]_o \sqsupseteq h_Y([\![S]\!])$. Note, that $h_Y([\![S]\!]) = h_Y([\![S]\!]_o)$. More generally, when comparing systems or views, we compare them w.r.t. their observable behaviors. For instance, when writing $V_1 \sqsupseteq V_2$, we mean $[\![V_1]\!]_o \sqsupseteq [\![V_2]\!]_o$.

5.2 Least and greatest fully-observable views

Let S be a discrete system over set of observable variables X. Given a set $Y \subseteq X$, one might ask whether there is a "canonical" view V of S w.r.t. Y. Clearly, if we allow V to have internal variables, the answer is yes: it suffices to turn all variables in $X \setminus Y$ into internal variables in V. Then, by Lemma 8, V represents precisely the projection of S to Y, i.e., it is a complete view, it satisfies $V = h_Y(S)$, and therefore trivially

also $V \supseteq h_Y(S)$ and $V \subseteq h_Y(S)$. Note that this is true independently of whether S has internal variables or not.

In this section we study the question for the case where we forbid V from having internal variables, i.e., we restrict views to be fully-observable. As FOS are not closed under projection, there are systems that have no complete fully-observable view. On the other hand, there can be multiple views V over Y such that $V \supseteq h_Y(S)$ or $V \subseteq h_Y(S)$. In particular, $(Y, \mathit{true}, \mathit{true}) \supseteq h_Y(S)$ and $(Y, \mathit{false}, \mathit{false}) \subseteq h_Y(S)$, for any S and Y. Thus, the question arises, whether there is a *least* fully-observable view $lv(S,Y)$ of S with $lv(S,Y) \supseteq h_Y(S)$, such that for any fully-observable view V' with $V' \supseteq h_Y(S)$, we have $V' \supseteq lv(S,Y)$. Similarly, one may ask whether there is a *greatest* fully-observable view $gv(S,Y)$ w.r.t. $\subseteq_{h_Y}$. These questions are closely related to whether views are closed under intersection and union. In particular, we can use closure under intersection to show that a least view always exists. A greatest view, on the other hand, does not necessarily exist.

Theorem 1. *Let $S = (X, Z, \theta, \phi)$ be any discrete system and let $Y \subseteq X$. Let ψ_S characterize the set of reachable states of S. Then the FOS*

$$(Y, \theta_Y = \exists (X \cup Z) \setminus Y : \theta, \phi_Y = \exists (X \cup Z) \setminus Y : \psi_S \wedge \exists (X' \cup Z') \setminus Y' : \phi)$$

is the unique fully-observable least view $lv(S,Y)$, that is, $lv(S,Y) \supseteq h_Y(S)$, and for any fully-observable view V' over Y with $V' \supseteq h_Y(S)$, we have $V' \supseteq lv(S,Y)$.

Proof. To see that $lv(S,Y)$ is a view of S for partial order $\supseteq$, consider an arbitrary behavior $\sigma = s_1 \dots s_n$ of S. Then, $\theta(s_1)$ and for all $i = 1, \dots, n-1$, we have $\psi_S(s_i)$ and $\phi(s_i, s_{i+1})$. We need to show that $h_Y(\sigma) = h_Y(s_1) \dots h_Y(s_n)$ is a behavior of $lv(S,Y)$. $\theta(s_1)$ implies $\theta_Y(h_Y(s_1))$, and so $h_Y(s_1)$ is an initial state of $lv(S,Y)$. Similarly, $\psi_S(s_i) \wedge \phi(s_i, s_{i+1})$ implies $\phi_Y(h_Y(s_i), h_Y(s_{i+1}))$, and so $h_Y(s_1) \dots h_Y(s_n)$ is a behavior of $lv(S,Y)$, and $lv(S,Y)$ is a view of S for partial order $\supseteq$.

It remains to show that $lv(S,Y)$ is the least view of S. Assume V with $[\![V]\!] \not\supseteq [\![lv(S,Y)]\!]$ is another view of S. Then, there is a behavior $\sigma = t_1 \dots t_n$ of $lv(S,Y)$ that is not a behavior of V. Let $t_1 \dots t_i$ be the shortest prefix of σ that is in $lv(S,Y)$ but not in V. Then, (t_{i-1}, t_i) is not a transition of V. As (t_{i-1}, t_i) is a transition of $lv(S,Y)$, there must be a reachable state s_{i-1} of S with $h_Y(s_{i-1}) = t_{i-1}$ and a state s_i of S with $\phi(s_{i-1}, s_i)$ and $h_Y(s_i) = t_i$. As s_{i-1} is reachable, there must be a sequence ending in s_{i-1}, s_i in S. The projection of this sequence cannot be in V, as $(h_Y(s_{i-1}), h_Y(s_i))$ is not a transition in V, and thus V is not a view of S. □

As Lemma 5 shows, the projection of a system cannot generally be represented as a fully-observable view. As it turns out, we can effectively check whether this is the case for a given system S, by checking whether the least view of S conforms to S w.r.t. $=$.

Theorem 2. *Given discrete system S over X and $Y \subseteq X$, there exists a fully-observable view V over Y with $V = h_Y(S)$ iff $[\![lv(S,Y)]\!] = h_Y([\![S]\!])$.*

Proof. Clearly, if $[\![lv(S,Y)]\!] = h_Y([\![S]\!])$ then $lv(S,Y)$ is a view V with $V = h_Y(S)$. For the opposite direction, we must show that if $lv(S,Y) \neq h_Y(S)$, then there is no view V with $V = h_Y(S)$. Assume for a contradiction that $V = h_Y(S)$, but not $lv(S,Y) = h_Y(S)$. Then $[\![lv(S,Y)]\!] \supsetneq h_Y(S) = [\![V]\!] \supseteq h_Y(S)$, contradicting the fact that $lv(S,Y)$ is the least view of S for partial order $\supseteq$.

The least view $lv(S,Y)$ can be computed as shown in Theorem 1. Given $lv(S,Y)$ it is sufficient to check whether $lv(S,Y) \subseteq h_Y(S)$. This is shown to be decidable in Theorem 4 in the following section. □

Theorem 2 implies that it is decidable to check whether a system admits a fully-observable complete view V.

Theorem 3. *There is a discrete system S over X and a subset $Y \subseteq X$ for which there is no unique greatest fully-observable view $gv(S,Y)$ with $gv(S,Y) \subseteq h_Y(S)$, such that for any fully-observable view V' with $V' \subseteq h_Y(S)$, we have $V' \subseteq gv(S,Y)$.*

Proof. Consider the FOS $S = (\{x, y\}, \theta = (x \wedge y) \vee (\neg x \wedge \neg y), \phi = (x \wedge \neg x' \wedge y \wedge y') \vee (\neg x \wedge x' \wedge \neg y \wedge \neg y'))$. The FOS S_1 and S_2 from the proof of Lemma 6 are both views of S for $Y = \{x\}$, yet they are incomparable and there is no FOS view conforming to S w.r.t. $\subseteq$ that is greater than both of them as their union is not a view of S. □

5.3 View conformance checking for discrete systems and views

Problem 1. *Given discrete system $S = (X, Z, \theta, \phi)$, discrete view $V = (Y, W, \theta_V, \phi_V)$, where $Y \subseteq X$ and $Z \cap W = \emptyset$, and partial order $\sqsupseteq \in \{=, \subseteq, \supseteq\}$, check whether $V \sqsupseteq h_Y(S)$.*

Problem 2. *Given discrete systems $S_1 = (X, Z_1, \theta_1, \phi_1)$ and $S_2 = (X, Z_2, \theta_2, \phi_2)$, where $Z_1 \cap Z_2 = \emptyset$, and partial order $\sqsupseteq \in \{=, \subseteq, \supseteq\}$, check whether $[\![S_1]\!]_o \sqsupseteq [\![S_2]\!]_o$.*

Theorem 4. *Problem 1 can be reduced to Problem 2 in polynomial time. Problem 2 is in PSPACE.*

Proof. For the first part of the theorem, observe that discrete systems are closed under projection. An instance of Problem 1 can be transformed into an instance of Problem 2, simply by shifting the variables $X \setminus Y$ of S from the observable to the internal variables.

For the second part of the theorem, we limit our attention to the case $\sqsupseteq = \subseteq$, as the other two cases then follow trivially. Problem 2 can be reduced to the finite state automaton inequivalence problem, which is known to be in PSPACE [9]. As discrete systems are closed under union, we construct a system $S_\cup$, with $[\![S_\cup]\!]_o = [\![S_1]\!]_o \cup [\![S_2]\!]_o$. Then $[\![S_\cup]\!]_o = [\![S_2]\!]_o$ iff $[\![S_1]\!]_o \subseteq [\![S_2]\!]_o$. From $S_\cup$ and S_2 we can construct NFAs $M_\cup$ and M_2 that accept a sequence σ iff σ is an observable behavior of $S_\cup$ and S_2, respectively. □

Theorem 5. *Problem 1 is in P for partial order $\sqsupseteq = \supseteq$ if the discrete view V is a FOS.*

Proof. First, notice that if $Y \subseteq X$, then $V = (Y, \theta_V, \phi_V)$ is a view of $S = (X, Z, \theta, \phi)$ if and only if it is a view of the fully-observable system $S' = (X \cup Z, \theta, \phi)$. This is because $h_Y(S) = h_Y(S')$. Thus, in the following, we will assume S to be a FOS with $S = (X, \theta, \phi)$.

Let ψ_S denote the reachable states of S. ψ_S can, e.g., be computed incrementally using BDDs. Let $Z := X \setminus Y$ and $Z' := X' \setminus Y'$. Then, $V \supseteq_{h_Y} S$, if and only if the following two conditions hold, which can be effectively checked:

1. $\forall Y, Z : \theta(Y, Z) \rightarrow \theta_V(Y) \equiv \forall s : \theta(s) \rightarrow \theta_V(h_Y(s))$, and
2. $\forall Y, Z, Y', Z' : \psi_S(Y, Z) \rightarrow (\phi((Y, Z), (Y', Z')) \rightarrow \phi_V(Y, Y')) \equiv \forall s, s' : \psi_S(s) \rightarrow (\phi(s, s') \rightarrow \phi_V(h_Y(s), h_Y(s')))$.

We need to show that Conditions 1 and 2 from above hold, if and only if $V \supseteq_{h_Y} S$.

Let us first show that Conditions 1 and 2 imply $V \supseteq_{h_Y} S$:
We show this by induction over the length n of behaviors σ of S.
Base case: let $\sigma = s_0 \in [\![S]\!]$ be any behavior of length 1 of S. Then $\theta(s_0)$ must hold, which, by Condition 1 implies $\theta_V(h(s_0))$, which implies that $h(s_0) \in [\![V]\!]$.
Induction step: let $\sigma = s_0 s_1 \cdots s_{n-1} s_n \in [\![S]\!]$ be a sequence of length $n+1$. As S is by definition prefix-closed, $s_0 s_1 \cdots s_{n-1}$ is also in S. By the induction hypothesis, we know that $h(s_0)h(s_1) \cdots h(s_{n-1})$ is in $[\![V]\!]$. As $\sigma \in S$, s_{n-1} is reachable, thus $\psi_S(s_{n-1})$ holds. Thus, we can apply Condition 2, and deduce from the fact that $\phi(s_{n-1}, s_n)$, that $\phi_V(h(s_{n-1}), h(s_n))$. This in turn implies that $h(s_0)h(s_1) \cdots h(s_{n-1})h(s_n)$ is a behavior of V.

Now, let us show the opposite direction, i.e., that $V \supseteq_{h_Y} S$ implies Conditions 1 and 2. We show this by contraposition. Assume Condition 1 does not hold. Then, there is a valuation vY of Y and a valuation vZ of Z, such that $\theta(vYvZ)$ holds (where $vYvZ$ is the valuation that agrees with vY on Y and with vZ on Z), but $\theta_V(vY)$ does not. Clearly, $h(vYvZ) = vY$. So, $vYvZ \in [\![S]\!]$, but $h(vYvZ) \notin [\![V]\!]$, which implies that $V \supseteq_{h_Y} S$ does not hold. Now assume that Condition 2 does not hold. This implies that there are valuations vY, vZ and vY', vZ', such that $\psi_S(vYvZ)$ and $\phi(vYvZ, vY'vZ')$ hold, but $\phi_V(vY, vY')$ does not. As $vYvZ$ is thus reachable, there must be a behavior $s_0 \cdots (vYvZ) \in [\![S]\!]$. By $\phi(vYvZ, vY'vZ')$, we also have that $s_0 \cdots (vYvZ)(vY'vZ') \in [\![S]\!]$. Yet, because $\phi_V(vY, vY')$ does not hold, $h(s_0) \cdots h(vYvZ)h(vY'vZ') \notin [\![V]\!]$, which concludes the proof. □

Theorem 6. *Problem 1 is PSPACE-hard even if the discrete view V is fully-observable for $|Y| \geq 1$ and partial orders $=, \subseteq$. Problem 1 is also PSPACE-hard for $|Y| \geq 1$ and partial order $\supseteq$ if V is not restricted to be fully-observable.*

Proof. In [13], it is shown that checking the universality of non-deterministic finite automata (NFA), having the property that all states are final, is PSPACE-hard for alphabets of size at least 2.

We reduce this problem in polynomial time to Problem 1 for $|Y| \geq 1$, V being fully-observable, and partial orders $=$ and $\subseteq$. NFAs considered in [13] are quintuples $M = (Q, \Sigma, \delta, I, F)$, where Q is a finite set of states; Σ is a finite alphabet; $\delta : Q \times \Sigma \to 2^Q$ is the transition function, δ is naturally extended to sets of states and words; $I \subseteq Q$ is the set of initial states; and F is the set of final states, in this case $F = Q$. A word $w \in \Sigma^*$ is accepted by NFA M, if $\delta(I, w) \cap F \neq \emptyset$. The *universality problem* is to check whether every word $w \in \Sigma^*$ is accepted by a given NFA M, or not.

For a given NFA $M = (Q, \Sigma, \delta, I, F)$, we construct a discrete system S_M, such that $w = t_1 \dots t_n \in \Sigma^+$ is accepted by M if and only if there exists a behavior $\sigma = s_1 \dots s_n$ of S_M with $h_Y(\sigma) = h_Y(s_1) \dots h_Y(s_n) = e_\Sigma(t_1) \dots e_\Sigma(t_n) = e_\Sigma(w)$, where e_Σ is a function encoding the alphabet of M using the boolean variables Y_M of S_M. Then, M is universal if and only if $h_Y(S) = e_\Sigma(\Sigma)^+$, where $e_\Sigma(\Sigma)$ is the image of e_Σ, i.e., all valuations of Y that encode a letter in Σ. Note that ϵ is trivially accepted by M if the set of initial states I is non-empty, as all states, including the initial states, are by definition final.

The construction of $S_M = (X_M, \theta_M, \phi_M)$ is akin to the transformation of a Mealy machine into a Moore machine: a state of S_M corresponds to a state of M and a letter in Σ consumed in the previous transition of M. The set of variables X_M of S is the disjoint union of a set of variables Y_M that encodes the outputs Σ of M, and a set of variables Z_M that encodes the states Q of M. Due to the binary encoding, we have $|Y_M| = \lceil \log_2 |\Sigma| \rceil \geq 1$ and $|Z_M| = \lceil \log_2 |Q| \rceil$. Let $e_Q : Q \to (Z_M \to \mathbb{B})$ and $e_\Sigma : \Sigma \to (Y_M \to \mathbb{B})$ denote the encoding functions, and e_Q^{-1} and e_Σ^{-1} their inverses.

Then, the transition relation ϕ_M and the initial states θ_M of S_M are defined as follows:

$$\phi_M(s, s') := e_Q^{-1}(s'[Z]) \in \delta(e_Q^{-1}(s[Z]), e_\Sigma^{-1}(s'[Y])),$$
$$\theta_M(s') := \exists q \in I : e_Q^{-1}(s'[Z]) \in \delta(q, e_\Sigma^{-1}(s'[Y])),$$

for all s, s' for which $e_Q^{-1}(s[Z]), e_Q^{-1}(s'[Z]), e_\Sigma^{-1}(s[Y])$, and $e_\Sigma^{-1}(s'[Y])$ are defined. Expressions for ϕ_M and θ_M can be generated in $O(|\Sigma| \cdot |\delta|)$ by traversing all transitions of M and adding appropriate disjuncts. The factor of $|\Sigma|$ stems from the fact that each state of M is duplicated up to Σ times to account for the label of an incoming transition.

Assume $w = t_1 \dots t_n$ is accepted by M. Then, there exist $q_1 \dots q_n q_{n+1} \in Q$ with $q_1 \in I$ and $q_{i+1} \in \delta(q_i, t_i)$ for all $i = 1, \dots, n$. By construction, $e_Q(q_2) e_\Sigma(t_1) \dots e_Q(q_{n+1}) e_\Sigma(t_n) \in [\![S]\!]$ and thus $e_\Sigma(t_1) \dots e_\Sigma(t_n) \in h_Y(S)$. Similarly, if $e_\Sigma(t_1) \dots e_\Sigma(t_n) \in h_Y(S)$, then $t_1 \dots t_n$ is accepted by M.

To check whether S_M is "universal", we can construct a fully-observable view $V = (Y, \theta_V, \phi_V)$ with $\theta_V(s) := \exists t \in \Sigma : e_\Sigma(t) = s$ and $\phi_V(s, s') = \exists t : e_\Sigma(t) = s \wedge \exists t' \in \Sigma : e_\Sigma(t') = s'$. Clearly, $[\![V]\!] = e_\Sigma(\Sigma)^+$. Then, $V =_{h_Y} S_M$ if and only if $[\![V]\!] = e_\Sigma(\Sigma)^+ = h_Y(S_M)$, which is the case if and only if M is universal. The same holds for $V \subseteq_{h_Y} S_M$, as $h_Y(S_M)$ cannot be strictly larger than $[\![V]\!]$ by construction, and so $V \subseteq_{h_Y} S_M$ if and only if $V =_{h_Y} S_M$. This concludes the proof of the first part of the claim.

It remains to show that it is also PSPACE-hard to check whether $V \supseteq_{h_Y} S$ if V is not restricted to be fully-observable. To see this, we exchange the roles of the view and the system from above, and construct, for a given NFA M, a view V_M and a "universal" system S, such that $V_M \supseteq_{h_Y} S$ if and only if M is universal. Let $V_M = (Y_M, Z_M, \theta_M, \phi_M)$, where all of its components are defined as in the proof of the first part of the claim, above. Also, let $S = (Y_M, \theta \equiv \exists t \in \Sigma : e_\Sigma(t) = s, \phi_V \equiv \exists t : e_\Sigma(t) = s \wedge \exists t' \in \Sigma : e_\Sigma(t') = s')$. Then, the observable behaviors $[\![S]\!]_o$ of S correspond to all non-empty words Σ^+, and the observable behaviors of V_M correspond to all words accepted by M. Thus $V_M \supseteq_{h_Y} S$ if and only if M is universal. □

5.4 View consistency checking for discrete systems and views

Problem 3. *Given partial order $\sqsupseteq \in \{=, \subseteq, \supseteq\}$ and discrete views $V_1, ..., V_n$, with $V_i = (Y_i, W_i, \theta_i, \phi_i)$ for $i = 1, ..., n$, check whether there exists discrete system $S = (X, Z, \theta, \phi)$, with $X \supseteq Y_i$ for all i, such that*

$V_i \sqsupseteq_{h_{Y_i}} S$ *for all* i.

Problem 3 asks to check whether a given number of views are consistent w.r.t. projection as abstraction function and a given partial order among $=, \subseteq, \supseteq$. Note that we can assume without loss of generality that the witness system has set of observable variables $X = \bigcup_{i=1}^{n} Y_i$, as any extra variables could be made internal.

Problem 3 is trivially solved by the "all" system $\theta = \phi = \mathit{true}$ for $\subseteq$ and by the "empty" system $\theta = \phi = \mathit{false}$ for $\supseteq$. For $=$, if we restrict the witness system to be a FOS, then Problem 3 is trivially decidable as there are only finitely many systems with $X = \bigcup_{i=1}^{n} Y_i$. Clearly, this is not very efficient. Theorems 7-9 (which also apply to general discrete systems, non necessarily FOS) provide a non-brute-force method.

Theorem 7. *For a set of views* $V_1, \ldots, V_n$*, with* $V_i = (Y_i, W_i, \theta_i, \phi_i)$ *for all* i*, there always exists a computable unique greatest witness system* $gw(V_1, \ldots, V_n) = (X, Z, \theta, \phi)$*, with* $X = \bigcup_{i=1}^{n} Y_i$*, w.r.t. partial order* $\supseteq$*.*

Proof. First, observe that $S_i = (X, W_i, \theta_i, \phi_i)$ is the unique greatest witness system for V_i for systems with the set of variables X, i.e., $V_i \sqsupseteq_{h_{Y_i}} S_i$ and for all $S = (X, W, \theta, \phi)$ such that $V_i \sqsupseteq_{h_{Y_i}} S$, we have $[\![S_i]\!] \supseteq [\![S]\!]$. In fact, $V_i =_{h_{Y_i}} S_i$. Given two views V_i, V_j, the unique greatest witness system for both views is $S_{i,j} = (X, W_i \cup W_j, \theta_i \wedge \theta_j, \phi_i \wedge \phi_j)$, whose behaviors are exactly the intersection of the behaviors of S_i and S_j (see Lemma 8). Adding any behavior to $S_{i,j}$ would violate either $V_i \sqsupseteq_{h_{Y_i}} S_{i,j}$ or $V_j \sqsupseteq_{h_{Y_i}} S_{i,j}$ Generalizing the above, $S_\wedge = (X_\wedge = \bigcup_{i=1}^{n} Y_i, Z_\wedge = \bigcup_{i=1}^{n} W_i, \theta_\wedge = \bigwedge_{i=1}^{n} \theta_i, \phi_\wedge = \bigwedge_{i=1}^{n} \phi_i)$ is the unique greatest witness system for the set of views $V_1, \ldots, V_n$. □

Theorem 8. *Consistency with respect to* $=$ *holds if and only if the greatest witness system* $gw(V_1, \ldots, V_n)$ *derived in Theorem 7 is a witness with respect to* $=$*.*

Proof. Clearly, if $gw(V_1, \ldots, V_n)$ is a witness for $=$, then consistency with $=$ holds. So, assume, $gw(V_1, \ldots, V_n)$ is no witness for $=$. We need to show that no witness for $=$ exists, i.e., that consistency with $=$ does not hold. Assume for a contradiction S is a witness for $=$. As $gw(V_1, \ldots, V_n)$ is no witness for $=$ but S is, there must be a V_i, such that $V_i = h_{Y_i}(S) \supsetneq h_{Y_i}(gw(V_1, \ldots, V_n))$. Due to the monotonicity of h_{Y_i}, this implies that $S \not\subseteq gw(V_1, \ldots, V_n)$. However, $V_i =_{h_{Y_i}} S$ implies $V_i \sqsupseteq_{h_{Y_i}} S$. Thus, S is also a witness for $\supseteq$, contradicting the assumption that $gw(V_1, \ldots, V_n)$ is the greatest such witness. □

Theorem 9. *Problem 3 is PSPACE-complete for partial order* $=$*.*

Proof. Checking consistency is in PSPACE, because it reduces to checking conformance of the greatest witness system with all of the views, which is shown to be decidable in PSPACE in the proof of Theorem 4.

For the PSPACE-hardness, we reduce conformance checking, shown to be PSPACE-hard in Theorem 6, to consistency checking: Let $V = (Y, W, \theta_V, \phi_V)$ and $S = (X, Z, \theta, \phi)$, with $Y \subseteq X$ and $W \cap Z = \emptyset$. Then, $[\![V]\!]_o = h_Y([\![S]\!]_o)$ if and only if V and S are consistent w.r.t. $=$, i.e., there exists a witness system S' with $[\![S]\!]_o = h_X([\![S']\!]_o)$ and $[\![V]\!]_o = h_Y([\![S']\!]_o)$. Clearly, if $[\![V]\!]_o = h_Y([\![S]\!]_o)$, then S is a witness system for consistency of V and S w.r.t. $=$, as $h_X([\![S]\!]_o) = [\![S]\!]_o$. For the other direction, assume $S' = (X', Z', \theta', \phi')$ is a witness system for consistency of V and S w.r.t. $=$. Then, $h_X([\![S']\!]_o) = [\![S]\!]_o$. As $[\![V]\!]_o = h_Y([\![S']\!]_o) = h_Y(h_X([\![S']\!]_o) = h_Y([\![S]\!]_o)$, S is also a witness for consistency. □

Theorem 10. *There are discrete views* $V_1, \ldots, V_n$*, with* $V_i = (Y_i, W_i, \theta_i, \phi_i)$ *for all* i*, for which there is no unique least witness system* $lw(V_1, \ldots, V_n) = (X, Z, \theta, \phi)$*, with* $X = \bigcup_{i=1}^{n} Y_i$*, w.r.t. partial order* $\subseteq$*.*

Proof. Consider the following two views $V_x = (\{x\}, \theta_x = x, \phi_x = \mathit{true})$ and $V_y = (\{y\}, \theta_y = y, \phi_y = \mathit{true})$. We provide two witness systems S_1, S_2, both consistent with V_x, V_y, such that their intersection is not consistent with V_x and V_y, which proves that there is no unique least witness system for V_x, V_y w.r.t. $\subseteq$:

$$S_1 = (\{x, y\}, \theta_1 = x \wedge y, \phi_1 = (x \Leftrightarrow y) \wedge (x' \Leftrightarrow y'))$$
$$S_2 = (\{x, y\}, \theta_2 = x \wedge y, \phi_2 = x' \vee y')$$

In every behavior of S_1, x and y take the same value, whereas in S_2, x and y are never both *false*. In their intersection $S_\cap = (\{x, y\}, \theta_1 \wedge \theta_2, \phi_1 \wedge \phi_2)$, neither x nor y can thus ever be *false*. So $S_\cap$ is neither consistent with V_x nor with V_y. □

5.5 View reduction for discrete systems and views

Problem 4. *Given partial order $\sqsupseteq \in \{=, \subseteq, \supseteq\}$ and discrete views $V_1, ..., V_n$, with $V_i = (Y_i, W_i, \theta_i, \phi_i)$ for $i = 1, ..., n$, compute $reduce_i(V_1, ..., V_n)$ for all $i = 1, ..., n$.*

Theorem 11. *For partial order $\supseteq$, Problem 4 is solved by the projection of the greatest witness system to the observable variables of the respective view: let $gw(V_1, \ldots, V_n) = (X, Z, \theta, \phi)$, with $X = \bigcup_{i=1}^n Y_i$, be the greatest witness system to the consistency of $V_1, ..., V_n$ w.r.t. partial order $\supseteq$. Then:*

$$reduce_i(V_1, ..., V_n) = (Y_i, Z \cup (X \setminus Y_i), \theta, \phi).$$

Proof. Observe that projection is distributive, i.e., $h_Y(\bigcup_{i=1}^n [\![S_i]\!]_o) = \bigcup_{i=1}^n h_Y([\![S_i]\!]_o)$.

$$\begin{array}{lll} reduce_i(V_1, ..., V_n) & & \\ & \overset{\text{Def. } reduce_i}{=} & h_{Y_i}(\bigcap_{i=1}^n c_{\models_i}(V_i)) \\ & \overset{\text{Lemma 2}}{=} & \bigsqcup\{h_{Y_i}(S) \mid S \in \bigcap_{i=1}^n c_{\models_i}(V_i)\} \\ & \overset{\sqsupseteq = \supseteq}{=} & \bigcup\{h_{Y_i}(S) \mid S \in \bigcap_{i=1}^n c_{\models_i}(V_i)\} \\ & \overset{h_{Y_i} \text{ is distributive}}{=} & h_{Y_i}(\bigcup\{S \mid S \in \bigcap_{i=1}^n c_{\models_i}(V_i)\}) \\ & \overset{\text{Theorem 7}}{=} & h_{Y_i}(gw(V_1, \ldots, V_n)) \\ & \overset{\text{Lemma 7}}{=} & (Y_i, Z \cup (X \setminus Y_i), \theta, \phi). \end{array}$$

□

For partial order $\subseteq$, Problem 4 is often trivial. Specifically, if the sets of observable variables of all views are incomparable, then no information can be transferred from one view to another:

Theorem 12. *Let $V_1, ..., V_n$ be discrete views with $V_i = (Y_i, W_i, \theta_i, \phi_i)$. Assume $Y_i \setminus Y_j \neq \emptyset$ for all i, j. Then, assuming $\sqsupseteq$ is $\subseteq$, the following holds for all i:*

$$reduce_i(V_1, ..., V_n) = V_i.$$

Proof. From Lemma 3 we know that the set of witnesses to the consistency of views $V_1, ..., V_n$ is invariant under reduction, i.e., for all i, $\bigcap_{i=1}^n c_{\models_i}(reduce_i(V_1, V_2, ..., V_n)) = \bigcap_{i=1}^n c_{\models_i}(V_i)$. In the same lemma, $reduce_i$ is also shown to be reductive. For partial order $\sqsupseteq = \subseteq$ this means that $reduce_i(V_1, ..., V_n) \supseteq V_i$. We show that in the setting considered here, $reduce_i(V_1, ..., V_n) \supsetneq V_i$ would violate the invariance property of reduction mentioned above, which then immediately implies the theorem.

So for a contradiction, assume for some i there is a behavior $\sigma = s_1 s_2 \ldots \in reduce_i(V_1, ..., V_n) \setminus V_i$. We will construct a system S that is a witness to the consistency of $V_1, ..., V_n$, yet $\sigma \notin h_{Y_i}(S)$. So S is *not* a witness to the consistency of $reduce_1(V_1, V_2, ..., V_n), ..., reduce_n(V_1, V_2, ..., V_n)$. S is constructed as the union of witness systems S_i for V_i and S_j for each $V_j, j \neq i$ as follows:

$$S_i = (\bigcup_{i=1}^n Y_i, W_i, \theta_{V_i}, \phi_{V_i}),$$

$$S_j = (\bigcup_{i=1}^n Y_i, W_j, \theta_{V_j} \wedge \bigwedge_{y \in Y_i \setminus Y_j} y \neq s_1(y), \phi_{V_j}).$$

By construction, $V_i = h_{Y_i}(S_i)$ and $V_j = h_{Y_j}(S_j)$ for all $j \neq i$, as the initial state of S_j is only constrained on variables that are irrelevant to V_j. By monotonicity of projection, this implies $V_k \subseteq h_{Y_k}(S_i \cup \bigcup_{j \neq i} S_j)$ for all $k = 1...n$. At the same time, $\sigma \notin h_{Y_i}(S_i \cup \bigcup_{j \neq i} S_j)$, as $\sigma \notin h_{Y_i}(S_i)$ by our initial assumption, and $\sigma \notin h_{Y_i}(S_j)$ for all $j \neq i$ by construction. □

Conjecture 1. *Let $V_1, ..., V_n$ be discrete views with $V_i = (Y_i, W_i, \theta_{V_i}, \phi_{V_i})$ and let $\sqsupseteq = \supseteq$. Let $I_i = \{j \mid Y_j \supseteq Y_i\}$ and let z_i be a fresh variable for every i. Then,*

$$reduce_i(V_1, ..., V_n) = \left(Y_i, \left(\bigcup_{j \in I_i} (Y_j \cup W_j \cup \{z_j\}) \right) \setminus Y_i, \bigwedge_{j \in I_i} z_j \rightarrow \theta_{V_j}, \left(\bigwedge_{j \in I_i} z_j \rightarrow \theta_{V_j} \right) \wedge \left(\bigwedge_{j \in I_i} z_j \leftrightarrow z'_j \right) \right).$$

Intuition: If $Y_j \not\supseteq Y_i$, then V_i cannot be reduced at all based on V_j (we would have to "invent" the missing pieces). Otherwise, we can add to V_i the projection of the observable behaviors of V_j to the observable variables of V_i.

6 Discussion

MVM is not a new topic, and terms such as "view" and "viewpoint" often appear in system engineering literature, including standards such as ISO 42010 [12]. Despite this fact, and the fact that MVM is a crucial concern in system design, an accepted mathematical framework for reasoning about views has so far been lacking. This is especially true for *behavioral* views, that is, views describing the dynamic behavior of the system, as opposed to its static structure. Behavioral views are the main focus of our work.

Discrete behavioral views could also be captured in a temporal logic formalism such as LTL. View consistency could then be defined as satisfiability of the conjunction $\phi_1 \wedge \cdots \wedge \phi_n$, where each ϕ_i is a view (possibly over a different set of variables). This definition is however weaker than our definition of strict consistency (w.r.t. $=$). Satisfiability of $\phi_1 \wedge \cdots \wedge \phi_n$ is equivalent to checking that the intersection of the inverse projections of views is non-empty, which, as we explained earlier, is a necessary but not sufficient condition for strict consistency.

The same fundamental difference exists between our framework and view consistency as formulated in the context of interface theories, where a special type of interface conjunction is used [11] (called "fusion" in [2] and "shared refinement" in [7, 17]).

Behavioral abstractions/views are also the topic of [15, 16]. Their framework is close to ours, in the sense that it also uses abstraction functions to map behaviors between different levels of abstraction (or between systems and views). The focus of both [15, 16] is to ease the verification task in a heterogeneous (e.g., both discrete and continuous) setting. Our main focus is checking view consistency. The notion of "heterogeneous consistency" [15] is different from our notion of view consistency. The notion of "conjunctive implication" [15] is also different, as views which have an empty intersection of their inverse projections trivially satisfy conjunctive implication, yet these views are in our framework inconsistent. Problems such as view consistency checking are not considered in [15, 16].

Consistency between architectural views, which capture structural but not behavioral aspects of a system, is studied in [3]. Consistency problems are also studied in [8] using a static, logic-based framework. Procedures such as join and normalization in relational databases also relate to notions of static consistency.

An extensive survey of different approaches for multi-view modeling can be found in [14]. [14] also gives a partial and preliminary formalization, but does not discuss algorithmic problems. [4] discusses an informal methodology for selecting formalisms, languages, and tools based on viewpoint considerations. A survey of trends in multi-paradigm modeling can be found in [1]. Trends and visions in multi-view modeling are also the topic of [18]. The latter paper also discusses pragmatics of MVM in the context of the Ptolemy tool. However formal aspects of MVM and algorithmic problems such as checking consistency are not discussed.

Implicitly, MVM is supported by multi-modeling languages such as UML, SysML, and AADL. For instance, AADL defines separate "behavior and error annexes" and having separate models in these annexes can result in inconsistencies. But capabilities such as conformance or consistency checking are typically not provided by the tools implementing these standards. Architectural consistency notions in a UML-like framework are studied in [6].

This work is a first step toward a formal and algorithm-supported framework for multi-view modeling. A natural direction for future work is to study algorithmic problems such as consistency checking in a heterogeneous setting. Although the framework of Section 3 is general enough to capture heterogeneity, in

this paper we restricted our attention to algorithmic MVM problems for discrete systems, as we feel that we first need a solid understanding of MVM in this simpler case.

Other directions for future work including investigating other types of abstraction functions, generalizing the methods developed in Section 5, e.g., so that $\subseteq, =, \supseteq$ can be arbitrarily combined, and studying algorithmic problems related to orthogonality.

References

[1] V. Amaral, C. Hardebolle, G. Karsai, L. Lengyel, and T. Levendovszky. Recent advances in multi-paradigm modeling. In *MODELS*, pages 220–224. Springer, 2010.

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

[3] A. Bhave, B.H. Krogh, D. Garlan, and B. Schmerl. View consistency in architectures for cyber-physical systems. In *ICCPS 2011*, pages 151 –160, 2011.

[4] D. Broman, E.A. Lee, S. Tripakis, and M. Törngren. Viewpoints, Formalisms, Languages, and Tools for Cyber-Physical Systems. In *MPM*, 2012.

[5] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *POPL*, pages 269–282. ACM, 1979.

[6] R. M. Dijkman. *Consistency in Multi-Viewpoint Architectural Design*. PhD thesis, University of Twente, 2006.

[7] L. Doyen, T. Henzinger, B. Jobstmann, and T. Petrov. Interface theories with component reuse. In *EMSOFT*, pages 79–88, 2008.

[8] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency handling in multiperspective specifications. *IEEE TSE*, 20(8):569–578, 1994.

[9] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[10] P. Granger. Improving the results of static analyses programs by local decreasing iteration. In *FSTTCS*, pages 68–79, London, UK, UK, 1992. Springer-Verlag.

[11] T. A. Henzinger and D. Nickovic. Independent implementability of viewpoints. In *Monterey Workshop*, volume 7539 of *LNCS*, pages 380–395. Springer, 2012.

[12] 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.

[13] J.-Y. Kao, N. Rampersad, and J. Shallit. On NFAs where all states are final, initial, or both. *Theoretical Computer Science*, 410(4749):5010 – 5021, 2009.

[14] M. Persson, M. Törngren, et al. A Characterization of Integrated Multi-View Modeling for Embedded Systems. In *EMSOFT*, 2013.

[15] A. Rajhans and B. H. Krogh. Heterogeneous verification of cyber-physical systems using behavior relations. In *HSCC '12*, pages 35–44. ACM, 2012.

[16] A. Rajhans and B. H. Krogh. Compositional heterogeneous abstraction. In *HSCC '13*, pages 253–262. ACM, 2013.

[17] S. Tripakis, B. Lickly, T. A. Henzinger, and E. A. Lee. A theory of synchronous relational interfaces. *ACM Trans. on Progr. Lang. and Sys. (TOPLAS)*, 33(4), July 2011.

[18] R. von Hanxleden, E. A. Lee, C. Motika, and H. Fuhrmann. Multi-view modeling and pragmatics in 2020. In *17th Intl. Monterey Workshop*, LNCS, 2012.