On the Consistency, Expressiveness, and Precision of Partial Modeling Formalisms ☆

Ou Wei*,a,c, Arie Gurfinkel[b], Marsha Chechik[a]

[a]Department of Computer Science, University of Toronto, Canada
[b]Software Engineering Institute, Carnegie Mellon University, USA
[c]Nanjing University of Aeronautics and Astronautics, China

Abstract

Partial transition systems support abstract model checking of complex temporal properties by combining both over- and under-approximating abstractions into a single model. Over the years, three families of such modeling formalisms have emerged, represented by (1) Kripke Modal Transition Systems (KMTSs), with restrictions on necessary and possible behaviors; (2) Mixed Transition Systems (MixTSs), with relaxation on these restrictions; and (3) Generalized Kripke MTSs (GKMTSs), with hyper-transitions, respectively. In this paper, we investigate these formalisms based on two fundamental ways of using partial transition systems (PTSs) – as objects for abstracting concrete systems (and thus, a PTS is semantically consistent if it abstracts at least one concrete system) and as models for checking temporal properties (and thus, a PTS is logically consistent if it gives consistent interpretation to all temporal logic formulas). We study the connection between semantic and logical consistency of PTSs, compare the three families w.r.t. their expressive power (i.e., what can be modeled, what abstractions can be captured using them), and discuss the analysis power of these formalisms, i.e., the cost and precision of model checking.

Specifically, we identify a class of PTSs for which semantic and logical consistency coincide and define a necessary and sufficient structural condition to guarantee consistency. We also show that all three families of PTSs have the same expressive power (but do differ in succinctness). However, GKMTSs are more precise (i.e, can establish more properties) for model checking than the other two families. The direct use of GKMTSs in practice has been hampered by the difficulty of encoding them symbolically. We address this problem by developing a new semantics for temporal logic of PTSs that makes the MixTS family as precise for model checking as the GKMTS family. The outcome is a symbolic model checking algorithm that combines the efficient encoding of MixTSs with the model checking precision of GKMTSs. Our preliminary experiments indicate that the new algorithm is a good match for predicate-abstraction-based model checkers.

☆Preliminary version of some aspects of this paper has appeared in [32].

*Corresponding author

Email addresses: owei@cs.toronto.edu (Ou Wei), arie@cmu.edu (Arie Gurfinkel), chechik@cs.toronto.edu (Marsha Chechik)

Preprint submitted to Elsevier *September 15, 2010*

Key words: Partial transition systems, modal transition systems, 3-valued analysis, abstraction, model-checking.

1. Introduction

Abstraction is the key to scaling model checking to industrial-sized problems. Typically, a large (or infinite) concrete system is approximated by a smaller abstract system via: (a) abstracting the concrete states, (b) analyzing the resulting abstract system, and (c) lifting the result back to the concrete system. Two common abstraction schemes are *over-approximation* – the abstract system contains *more* behaviours than the concrete one, and *under-approximation* – the abstract system contains *less* behaviours than the concrete one. Over-approximation is sound for universal properties (e.g., absence of errors). Under-approximation is sound for existential properties (e.g., presence of errors).

Abstractions that are sound for arbitrary properties, such as full μ-calculus L_μ [23], must combine over- and under-approximation into a *single* model [25, 10]. This leads to transition systems (TSs) with two types of transitions, *may* and *must*, representing *possible* (or over-approximating), and *necessary* (or under-approximating) behaviours, respectively. We call such systems *partial*. A temporal property is interpreted over a partial TS in one of four ways: *true* or *false*, if the partial TS is precise enough to prove or disprove the property, *unknown*, if the TS is imprecise, and *inconsistent* otherwise.

There are three families of partial modeling formalisms identified in the literature:

1. *Kripke Modal Transition Systems* (KMTSs) [22] and their equivalent variants, *Modal Transition Systems (MTSs)* [25], *Partial Kripke Structures* (PKSs) [7], and *3-valued Kripke Structures* [8]. KMTSs require that every *must* transition is also a *may* transition. They were introduced as computational models for partial specifications of reactive systems [25] and then adapted for model checking [7, 22, 8].
2. *Mixed Transition Systems* (MixTSs) [10], and equivalently, *Belnap Transition Systems* [19]. MixTSs extend KMTSs by allowing *must only* transitions (i.e., transitions that are *must* but not *may*). MixTSs were introduced in [10] as abstract models for L_μ, and have been used for predicate abstraction and software model checking in [18].
3. *Generalized KMTSs* (GKMTSs) [29], and equivalently, *Abstract TSs* [13] and *Disjunctive MTSs* [26]. GKMTSs extend MixTSs by allowing *must hypertransitions*, (i.e., transitions into sets of states).

In this paper, we study these formalisms from two points of view: a semantic one, using partial TSs as objects for abstracting concrete systems, and a logical one, using partial TSs for temporal logic model checking. A partial TS is *semantically consistent* if it abstracts at least one concrete system. A partial TS is *logically consistent* if it gives consistent interpretation to all temporal logic formulas. For semantic consistency, we investigate partial transition systems for abstract model checking, where a partial transition system and its concrete refinement are related through the soundness relation of abstract and concrete states. The notion of semantic consistency in this setting (formally defined in Section 4) is slightly different from the notion of *implementability* where partial transition systems are used as specifications of a system's behavior. A discussion of this difference is given in Section 9. Specifically, in this paper we first study the connection between semantic and logical consistency of partial TSs. We then compare the expressive power of the formalisms, i.e., what abstractions can be captured using them. Finally, we discuss the analysis power of these formalisms, i.e., the cost and precision of model checking.

Consistency. Semantic consistency implies logical consistency but the converse is not true in general: Temporal logic is not expressive enough to detect all forms of inconsistency.

In this paper, we answer several questions about consistency: Is there a subclass of partial TSs for which semantic and logical consistency coincide? Do TSs outside of this subclass have additional expressive power? Is there a necessary and sufficient condition for ensuring consistency?

We show that there is a class of partial TSs for which semantic and logical consistency coincide. We call this class *monotone* because of the monotonicity condition we impose on the transition relation. The class of monotone TSs is as expressive as the class of all partial TSs. Thus, for every partial TS, there is an equivalent monotone one.

At a first glance, it may appear that a structural requirement "every *must* transition is also a *may* transition" is sufficient and necessary to guarantee both semantic and logical consistency. However, this is not the case. We show that for logical consistency, this requirement is sufficient but not necessary: weaker condition exists. For semantic consistency, the requirement is neither necessary nor sufficient. Instead, for monotone TSs, where semantic and logical consistency coincide, we define an alternative structural condition and show that it is both necessary and sufficient to guarantee consistency.

Expressive Power. We show that all three families of partial TSs, KMTSs, MixTSs, and GKMTSs, are equally expressive: for any partial TS M expressed in one formalism, there exists a partial TS M' in the other such that M and M' approximate the same set of concrete systems. That is, neither hyper-transitions nor restrictions on *may* and *must* transitions affect expressiveness. They do, however, affect the size of the models: GKMTSs and KMTSs can be converted to semantically equivalent MixTSs of (possibly exponentially) smaller or equal size. Dams and Namjoshi have shown that the three families of partial TSs are less expressive than tree automata [12]. We complete the picture by showing the expressive equivalence *between* these families.

Model Checking. We call a semantics of temporal logic *inductive* if it is defined inductively on the syntax of the logic. We refer to the typical inductive semantics of L_μ on partial TSs as the *Standard Inductive Semantics* (SIS). This is the semantics most widely used in other works on this subject as well as in practice. A GKMTS G can prove/disprove more properties under SIS than either a MixTS or KMTS obtained from G by semantics-preserving translation. However, while both MixTSs and KMTSs have been used in practical symbolic model checkers (e.g., [18, 20, 8]), the direct use of GKMTSs has been hampered by the difficulty of encoding hyper-transitions into BDDs. To address this problem, we develop a new semantics, called *reduced* (RIS), that is inductive (and tractable) but is more precise than SIS. We show that GKMTSs and MixTSs are equivalent with respect to RIS, and give an efficient symbolic model checking procedure for RIS. The outcome is an algorithm that combines the benefits of the efficient symbolic encoding of MixTSs with the model checking precision of GKMTSs.

To show the practicality of the above result, we develop a symbolic model checking algorithm with respect to RIS and apply it to MixTSs constructed using predicate abstraction. We evaluate our implementation empirically against a SIS-based algorithm.

The rest of the paper is organized as follows. Section 2 reviews the necessary background on partial TSs and abstraction. We define the notion of *monotone* partial TSs in Section 3. In Section 4, we investigate semantic and logical consistency of partial TSs. In Section 5, we prove that KMTSs, MixTSs and GKMTSs are equally expressive by developing semantics-preserving translations from GKMTSs to MixTSs, and from MixTSs to KMTSs. In Section 6, we introduce *Reduced Inductive Semantics* (RIS) for L_μ. In Section 7, we present a symbolic model checking algorithm with respect to RIS in the context of predicate abstraction. We report on our experience with this algorithm in Section 8. We discuss related work and research directions following our results in Section 9 and conclude the paper in Section 10.

2. Preliminaries

In this section, we review several modeling formalisms, and their use for abstraction.

2.1. Transition Systems

Definition 1 (Transition Systems). [10, 6, 22, 29] A *Generalized Kripke Modal Transition System* (GKMTS) is a tuple $M = \langle S, R^{\text{may}}, R^{\text{must}} \rangle$, where S is the statespace, and $R^{\text{may}} \subseteq S \times S$, $R^{\text{must}} \subseteq S \times 2^S$ are the *may* and *must* transition relations, respectively. A *Mixed TS* (MixTS) is a GKMTS such that $R^{\text{must}} \subseteq S \times S$. A *Kripke Modal TS* (KMTS) is a MixTS such that $R^{\text{must}} \subseteq R^{\text{may}}$. A *Boolean TS* (BTS) is a KMTS such that $R^{\text{may}} = R^{\text{must}}$.

For example, a MixTS M_1 is shown in Figure 1, a GKMTS G_2 is shown in Figure 2, and a KMTS K_1 is shown in Figure 3. In these figures, *must* and *may* transitions are indicated by solid and dashed edges, respectively. In this article, the statespace of a transition system corresponds to an abstract domain. In this case, a state is labeled by its abstract element. For example, state a_4 of M_1 in Figure 1 corresponds to an abstract element $x \leq 0$. A *transition system* (TS) is any of GKMTS, MixTS, KMTS, and BTS. A *partial transition system* (PTS) is any of GKMTS, MixTS, and KMTS.

We write $s \xrightarrow{\text{may}} t$ for $(s,t) \in R^{\text{may}}$, $s \xrightarrow{\text{must}} t$, and $s \xrightarrow{\text{must}} Q$ for $(s,t) \in R^{\text{must}}$ and $(s,Q) \in R^{\text{must}}$, respectively. Intuitively, *may* and *must* transitions represent possible and necessary behaviours, respectively. BTS differs from all other transitions systems in that in it all *may* and *must* transitions coincide. We say that BTS is a complete (or, a concrete) transition system. For simplicity, we only show a single transition relation when specifying a BTS.

Let AP be a set of atomic propositions. A literal l over AP is either an atomic proposition p or its negation $\neg p$. Let $Lit(AP)$ be a set of literals of AP, and S be a statespace. A *state labeling* is a function $L : S \to 2^{Lit(AP)}$ that assigns to each state s a set of literals that are true in s. A pair $\langle M, L\rangle$ of a TS M and a labeling L is called a *model*.

In this paper, we make a distinction between a "transition system" and a "model". Although the two are often used interchangeably in model-checking literature, formally, there is a difference. A *transition system* is built out of states and transitions. A *model* extends a transition system with an interpretation of atomic propositions. In our work, we find it convenient to talk about properties of a transition system, and then show that they hold in all corresponding models.

We use the following naming convention. Roman capital letters denote transition systems: M for a MixTS, K for a KMTS, G for a GKMTS, and B for a BTS. Subscripts indicate a particular transition system. For example, M_1 is a MixTS (see Figure 1), whereas G_2 is a GKMTS (see Figure 2). Script capital letters denote models: $\mathcal{M}$ for a MixTS model, $\mathcal{K}$ for a KMTS model, $\mathcal{G}$ for a GKMTS model, and $\mathcal{B}$ for a BTS model. Subscripts indicate a model corresponding to a particular transition system. For example, $\mathcal{M}_1$ is a model whose underlying transition system is the MixTS M_1 (see Figure 1). The letter L is used exclusively to indicate a labeling function of a model.

The modal μ-calculus [23] (L_μ) is the set of all formulas satisfying the following BNF grammar:

$$\varphi ::= p \mid Z \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \mu Z \cdot \varphi\,,$$

where p is an atomic proposition, and Z a fixpoint variable. Furthermore, Z in φ of the form $\mu Z \cdot \varphi$ must occur under the scope of an even number of negations. Additional operators are defined as abbreviations:

$$\begin{aligned} \varphi \vee \psi &\triangleq \neg(\neg\varphi \wedge \neg\psi) \\ \Box\varphi &\triangleq \neg\Diamond\neg\varphi \\ \nu Z \cdot \varphi(Z) &\triangleq \neg\mu Z \cdot \neg\varphi(\neg Z)\,. \end{aligned}$$

Let $\mathcal{M} = \langle M, L\rangle$ be a model, where $M = \langle S, R^{\text{may}}, R^{\text{must}}\rangle$, and φ be an L_μ formula. An *interpretation* (or *semantics*) of φ over $\mathcal{M}$, denoted by $\|\varphi\|^{\mathcal{M}}$, is a pair $\langle U, O\rangle$, where $U \subseteq S$ is a set of states that satisfy φ, and $O \subseteq S$ is the set of states that do not refute φ. Intuitively, U and O represent an *under*-approximation and an *over*-approxiamtion of the set of all the states that satisfy φ, respectively. For a state $s \in S$, we say that φ is *true* at s iff $s \in U \cap O$, φ is *false* at s iff $s \in S \setminus (U \cup O)$, φ is *unknown* at s iff $s \in O \setminus U$, and φ is *inconsistent* at s iff $s \in U \setminus O$. Alternatively (e.g., [29]), the semantics of φ over $\mathcal{M}$ can be defined by a pair of states $\langle U, D\rangle$, where U is the set of all states that satisfy φ (same as above), and D is the set of all states that refute φ. In this paper, we use the first approach to remain compatible with the partitioning of the transition relation into *must* and *may* transitions.

For a universe S, let e be a pair $\langle U, O\rangle$ with $U, O \subseteq S$. We write $\mathsf{U}(e)$ and $\mathsf{O}(e)$ to denote U and O, respectively, and $\overline{Q}$ for the complement of Q in S, i.e., $\overline{Q} = S \setminus Q$.

We write $\sim$ and $\sqcap$ for the operators defined below:

$$\sim\langle U, O\rangle \triangleq \langle \overline{O}, \overline{U}\rangle$$
$$\langle U_1, O_1\rangle \sqcap \langle U_2, O_2\rangle \triangleq \langle U_1 \cap U_2, O_1 \cap O_2\rangle\,.$$

A semantics of L_μ is *inductive* if it is defined inductively on the syntax of the logic. We refer to the commonly used (e.g., [10, 6, 22, 29, 19]) inductive semantics as the *Standard Inductive Semantics* (SIS). It is defined as follows

Definition 2 (Standard Inductive Semantics (SIS)). [10, 6, 22, 29, 19]. Let $\mathcal{M} = \langle M, L\rangle$ be a model, $M = \langle S, R^{\text{may}}, R^{\text{must}}\rangle$, *Var* a set of fixpoint variables, and $\sigma : \mathit{Var} \to 2^S \times 2^S$. The *standard inductive semantics (SIS)* of $\varphi \in L_\mu$ is:

$$\begin{array}{rcl}
||p||^{\mathcal{M}}_{i,\sigma} & \triangleq & \langle \{s \mid p \in L(s)\}, \{s \mid \neg p \notin L(s)\}\rangle \\
||\neg\varphi||^{\mathcal{M}}_{i,\sigma} & \triangleq & \sim||\varphi||^{\mathcal{M}}_{i,\sigma} \\
||\varphi \wedge \psi||^{\mathcal{M}}_{i,\sigma} & \triangleq & ||\varphi||^{\mathcal{M}}_{i,\sigma} \sqcap ||\psi||^{\mathcal{M}}_{i,\sigma} \\
||\Diamond\varphi||^{\mathcal{M}}_{i,\sigma} & \triangleq & \langle pre_{\mathsf{U}}(\mathsf{U}(||\varphi||^{\mathcal{M}}_{i,\sigma})), pre_{\mathsf{O}}(\mathsf{O}(||\varphi||^{\mathcal{M}}_{i,\sigma}))\rangle \\
||Z||^{\mathcal{M}}_{i,\sigma} & \triangleq & \sigma(Z) \\
||\mu Z \cdot \varphi||^{\mathcal{M}}_{i,\sigma} & \triangleq & \langle \mathsf{lfp}^{\subseteq}\left(\lambda Q \cdot \mathsf{U}(||\varphi||^{\mathcal{M}}_{i,\sigma[Z\mapsto Q]})\right), \mathsf{lfp}^{\subseteq}\left(\lambda Q \cdot \mathsf{O}(||\varphi||^{\mathcal{M}}_{i,\sigma[Z\mapsto Q]})\right)\rangle
\end{array}$$

where $Z \in \mathit{Var}$, lfp is the least fixpoint, and the *pre-image* operators pre_{U} and pre_{O} are defined as follows:

$$pre_{\mathsf{U}}(Q) \triangleq \begin{cases} \{s \mid \exists t \in Q \cdot s \xrightarrow{\text{must}} t\} & \text{if } M \text{ is a MixTS} \\ \{s \mid \exists U \subseteq Q \cdot s \xrightarrow{\text{must}} U\} & \text{if } M \text{ is a GKMTS} \end{cases}$$
$$pre_{\mathsf{O}}(Q) \triangleq \{s \mid \exists t \in Q \cdot s \xrightarrow{\text{may}} t\}$$

2.2. Partial Models and Abstraction

Abstraction Relation.. In this paper, we maintain an explicit connection between concrete and abstract statespaces. We define these formally below:

Definition 3. An *abstraction relation* is a structure $\langle C, \rho, S\rangle$, where C and S are arbitrary sets and $\rho \subseteq C \times S$ is a binary relation satisfying the "existence of best approximation" condition [9]:

$$\forall c \in C \cdot \exists s \in S \cdot (\rho(c, s) \wedge \forall s' \in S \cdot \rho(c, s') \Rightarrow \gamma(s') \supseteq \gamma(s))\,.$$

For an abstraction relation $\langle C, \rho, S\rangle$, we say that C is the *concrete* statespace (or domain), S is the *abstract* statespace (or domain), and ρ is the *soundness* relation, where $(c, s) \in \rho$ means that s *ρ-approximates* c. ρ induces a *concretization* function $\gamma(s) \triangleq \{c \mid (c, s) \in \rho\}$. That is, $\gamma(s)$ is the set of all concrete states approximated by s. We extend γ to a set $Q \subseteq S$ by letting $\gamma(Q) \triangleq \cup_{s \in Q}\gamma(s)$. γ induces an approximation ordering $\preceq_a$ on S defined as follows $s \preceq_a t \Leftrightarrow \gamma(s) \supseteq \gamma(t)$. That is, $s \preceq_a t$ if s is *less precise* (more approximate) than t. Following [9], we require that $\preceq_a$ is a partial order.

Finally, an *abstraction* function $\alpha : C \to S$ is defined to map each concrete element to its best approximation. The image of α is denoted by $\alpha[C] \triangleq \{\alpha(c) \mid c \in C\}$.

Note that it is common to assume that α and γ form a Galois connection between S and powerset of C. We prefer a more general setting, as described in [9], and do not make this assumption. Contrary to most of the work on Abstract Interpretation, we write $a \preceq_a b$ to mean that a is less precise than b, i.e., $\preceq_a$ is the "precision ordering". In other works on Abstract Interpretation, it is common to write $a \preceq b$ to mean that a is less abstract than b, i.e., $\preceq$ is the "abstraction ordering", which is the reverse of the precison ordering. We are aware of the confusion this causes, but follow the convention typical of the work on partial transition systems.

In the rest of the paper, we require that labeling of a concrete statespace is *complete*: for any concrete state $c \in C$ and any concrete labeling L, $p \in L(c) \Leftrightarrow \neg p \notin L(c)$.

An abstract state s is *consistent* iff $\gamma(s) \neq \emptyset$. We require that any state labeling function L over an abstract statespace is *locally consistent*, i.e., for any consistent abstract state s and proposition p, at most one of p and $\neg p$ belongs to $L(s)$. Furthermore, we require L to be *monotone* with respect to$\preceq_a$: $s_1 \preceq_a s_2 \Rightarrow L(s_1) \subseteq L(s_2)$. We say that p is true in s if $p \in L(s)$, and false if $\neg p \in L(s)$; otherwise, we say that p is unknown in s.

Abstract Domain of Predicate Abstraction. Let C be a concrete statespace, n be a natural number, and $P = \{p_1, \ldots, p_n\}$ be a set of quantifier-free first-order boolean predicates over C. A *monomial* is a conjunction of literals of P; a *minterm* is a monomial in which each variable p_i appears exactly once (either positively or negatively). We write $\text{Mon}(P)$ and $\text{MT}(P)$ for the set of all monomials and minterms of P, respectively. The set $\text{Mon}(P)$ is the domain of predicate abstraction. The abstraction relation $\langle C, \rho_P, \text{Mon}(P)\rangle$ is defined such that $(c, s) \in \rho_P$ iff $c \models s$, i.e., c satisfies all literals in s; the abstraction $\alpha_P(c) \triangleq (\bigwedge_{c\models p_i} p_i) \wedge (\bigwedge_{c\not\models p_i} \neg p_i)$; $\alpha_P[C] = \text{MT}(P)$; and the approximation ordering is reverse implication, $s \preceq_a t$ iff $s \Leftarrow t$.

Simulation and Approximation. An approximation relation is extended from a statespace to transition systems using the concept of *mixed simulation*.

Definition 4 (Mixed Simulation). [10] Let $M = \langle S, R^{\text{may}}, R^{\text{must}}\rangle$ and $M' = \langle S', R^{\text{may}'}, R^{\text{must}'}\rangle$ be two MixTSs. $H \subseteq S \times S'$ is a *mixed simulation* between M and M' iff for any $(s, s') \in H$, the following two conditions hold:

$$\begin{array}{ll} \text{(a)} & \forall t \in S \cdot \ s \xrightarrow{\text{may}} t \ \Rightarrow \exists t' \in S' \cdot s' \xrightarrow{\text{may}} t' \wedge (t, t') \in H \\ \text{(b)} & \forall t' \in S' \cdot s' \xrightarrow{\text{must}} t' \Rightarrow \exists t \in S \cdot \ s \xrightarrow{\text{must}} t \wedge (t, t') \in H \end{array}$$

We say that M' H*-simulates* M, written $M' \preceq_H M$.

Intuitively, M' simulates M whenever M' is less precise about its behaviour than M. This definition generalizes to GKMTSs (cf. [29]).

Let $\langle C, \rho, S\rangle$ be an abstraction relation. A partial TS M with statespace S *approximates* a concrete BTS B with statespace C iff the soundness relation ρ is a mixed simulation between M and B, i.e., $M \preceq_\rho B$. Equivalently, we say that B *refines* M. For a fixed TS M, the set of all BTSs that refine it is denoted by $\mathbb{C}[M]$.

Let L_M and L_B be the state labeling functions for S and C, respectively. We say that L_M *approximates* L_B, denoted $L_M \preceq_\rho L_B$, iff $\rho(c, s) \Rightarrow L_M(s) \subseteq L_B(c)$.

Definition 5 (Approximation Relation). [10] Let $\langle C, \rho, S\rangle$ be an abstraction relation, $\mathcal{M} = \langle M, L_M\rangle$ be a partial model over S, and $\mathcal{B} = \langle B, L_B\rangle$ be a concrete model over C. $\mathcal{M}$ *approximates* $\mathcal{B}$ iff $M \preceq_\rho B$, and $L_M \preceq_\rho L_B$. Equivalently, we say that $\mathcal{B}$ *refines* $\mathcal{M}$.

Since this paper investigates partial models from the perspective of abstract model checking, we define concrete refinements of a partial model with respect to a fixed mixed simulation relation, i.e., the abstraction relation. It is possible to consider concrete refinements of a partial model with respect to all the possible mixed simulations. We discuss this difference in Section 9.

Theorem 1. *[10] Let $\langle C, \rho, S\rangle$ be an abstraction relation, $\mathcal{B} = \langle B, L_B\rangle$ be a concrete model, where $B = \langle C, R\rangle$, and $\mathcal{M} = \langle M, L_M\rangle$ be a partial model, where $M = \langle S, R^{\text{may}}, R^{\text{must}}\rangle$. If $\mathcal{M} \preceq_\rho \mathcal{B}$, then, for any L_μ formula φ:*

$$\gamma(\mathsf{U}(\|\varphi\|_i^{\mathcal{M}})) \subseteq \mathsf{U}(\|\varphi\|_i^{\mathcal{B}}) \text{ , and } \qquad \gamma(\overline{\mathsf{O}(\|\varphi\|_i^{\mathcal{M}})}) \subseteq \overline{\mathsf{O}(\|\varphi\|_i^{\mathcal{B}})} \; .$$

That is, if $\mathcal{M}$ approximates $\mathcal{B}$ and φ is true/false in a state s of $\mathcal{M}$, then it is, respectively, true/false in all states of $\mathcal{B}$ approximated by s.

Let $\mathbb{C}[\mathcal{M}]$ be the set of all concrete refinements of $\mathcal{M}$. Intuitively, $\mathbb{C}[\mathcal{M}]$ is the semantic meaning of $\mathcal{M}$. An interpretation of L_μ with respect to the semantic meaning of a model is called *thorough*. Note that since we consider concretizations of $\mathcal{M}$ with respect to a fixed abstraction relation, the thorough semantics defined here is different from the original definition in [7], which is based on *all* possible concretizations of the given partial model.

Definition 6 (Thorough Semantics). Let $\langle C, \rho, S\rangle$ be an abstraction relation, and $\mathcal{M}$ be a partial model over an abstract statespace S. The *thorough* semantics of an L_μ formula φ over $\mathcal{M}$ is defined as $\|\varphi\|_t^{\mathcal{M}} = \langle U, O\rangle$, where

$$U = \{a \in S \mid \forall \mathcal{B} \in \mathbb{C}[\mathcal{M}] \cdot \gamma(a) \subseteq \mathsf{U}(\|\varphi\|_i^{\mathcal{B}})\}$$
$$O = \{a \in S \mid \exists \mathcal{B} \in \mathbb{C}[\mathcal{M}] \cdot (\gamma(a) \cap \mathsf{O}(\|\varphi\|_i^{\mathcal{B}})) \neq \emptyset\}$$

In order to compare different interpretations of L_μ, we introduce two ordering relations on the space $2^S \times 2^S$.

Definition 7 (Information and Semantics Orderings). Let $\langle C, \rho, S\rangle$ be an abstraction relation, and let $e_1 = \langle U_1, O_1\rangle$ and $e_2 = \langle U_2, O_2\rangle$ be two elements in $2^S \times 2^S$. e_1 is *less informative* than e_2, written $e_1 \preceq_i e_2$, iff

$$U_1 \subseteq U_2 \quad \text{and} \quad O_2 \subseteq O_1 \; .$$

e_1 is *semantically less precise* than e_2, written $e_1 \preceq_a e_2$, iff

$$\gamma(U_1) \subseteq \gamma(U_2) \quad \text{and} \quad \gamma(\overline{O_1}) \subseteq \gamma(\overline{O_2}) \; .$$

We say e_1 and e_2 are semantically equivalent, denoted $e_1 \equiv_a e_2$, iff $e_1 \preceq_a e_2$ and $e_2 \preceq_a e_1$. Note that we use the same notation $\preceq_a$ to denote the precision orderings, defined with respect to concretization, for both the elements in S and the ones in $2^S \times 2^S$.

Finally, we define *semantic equivalence* for partial models and TSs, and *expressive equivalence* for partial modeling formalisms as follows:

Definition 8 (Semantic Equivalence). Two partial models $\mathcal{M}$ and $\mathcal{M}'$ are *semantically equivalent*, if and only if they have the same set of concrete refinements, i.e., $\mathbb{C}[\mathcal{M}] = \mathbb{C}[\mathcal{M}']$. Similarly, two partial transition systems, M and M', are *semantically equivalent*, if and only if $\mathbb{C}[M] = \mathbb{C}[M']$.

Definition 9 (Expressive Equivalence). Two partial modeling formalisms are *expressively equivalent* if and only if for every transition system M from one formalism, there exists a transition system M' from the other, such that M and M' are semantically equivalent.

3. Monotone Partial Transition Systems

In this section, we define *monotone* partial TSs. We show that monotone partial TSs are expressively equivalent (in the sense of Definition 9) to their regular counterparts: for any partial TS there exists an equivalent monotone one, i.e., they approximate the same set of concrete systems. The monotonicity condition simply ensures that all information that can be derived from existing *may* and *must* transitions is made explicit in the TS. As we show in later sections, this condition allows us to perform local reasoning of partial TSs more effectively.

For simplicity, we present the results with respect to MixTSs. They can be easily adapted to GKMTSs as well. Throughout the section, we assume that γ, α, and $\preceq_a$ are interpreted with respect to a fixed an abstraction relation $\langle C, \rho, S\rangle$.

Definition 10. A MixTS $M = \langle S, R^{\text{may}}, R^{\text{must}}\rangle$ is *monotone* iff

(a) $\forall s, t_1, t_2 \in S \cdot t_2 \preceq_a t_1 \Rightarrow ((s, t_2) \in R^{\text{may}} \Rightarrow (s, t_1) \in R^{\text{may}}) \wedge$
$((s, t_1) \in R^{\text{must}} \Rightarrow (s, t_2) \in R^{\text{must}})$

(b) $\forall s_1, s_2, t \in S \cdot s_1 \preceq_a s_2 \Rightarrow ((s_2, t) \in R^{\text{may}} \Rightarrow (s_1, t) \in R^{\text{may}}) \wedge$
$((s_1, t) \in R^{\text{must}} \Rightarrow (s_2, t) \in R^{\text{must}})$

A model $\mathcal{M} = \langle M, L\rangle$ is *monotone* iff its MixTS component M is monotone.

Intuitively, a transition system is monotone if the information captured by its transition relation is monotone with respect to the approximation ordering $\preceq_a$ of its states. For example, let M be a transition system, s_1 and s_2 be two states of M such that $s_1 \preceq_a s_2$. (1) Suppose there is a *may* transition from s_2 to some other state t. The meaning of this transition is that any system that refines M can have a transition from a state in $\gamma(s_2)$ to a state in $\gamma(t)$. Recall that we assumed that $s_1 \preceq_a s_2$; hence, $\gamma(s_1) \supseteq \gamma(s_2)$. Thus, the same behavior is allowed from the states in $\gamma(s_1)$. For M to be monotone with this information, it must have a *may* transition from s_1 to t. (2) Similarly, suppose there is a *must* transition from s_1 to some other state t. Then, every state

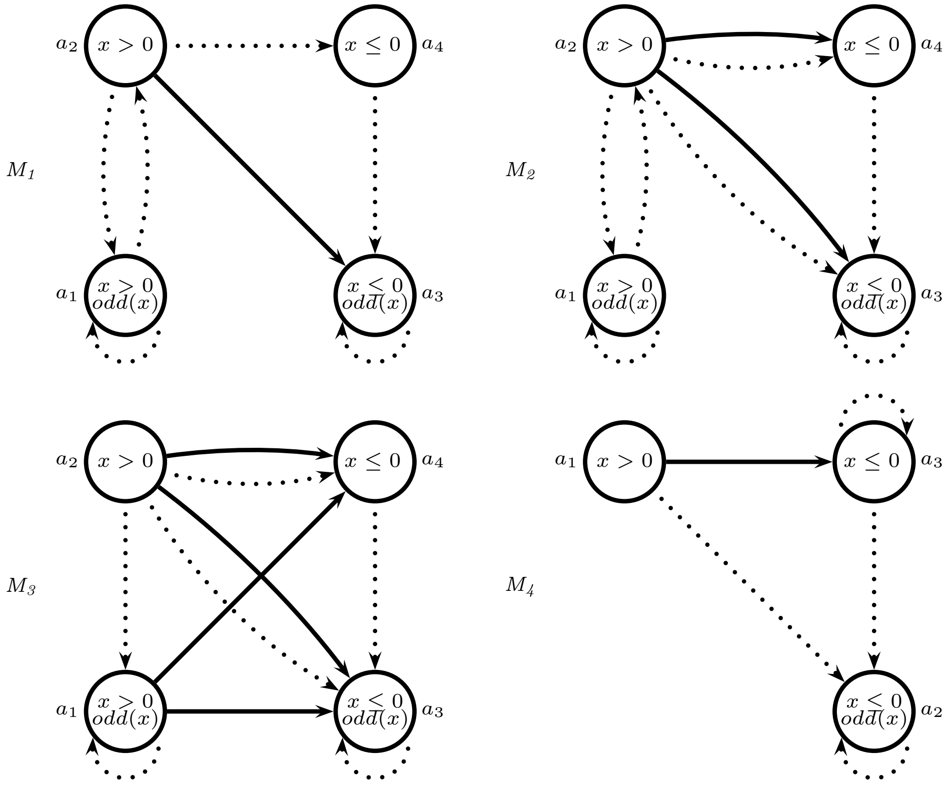

Figure 1: Four MixTSs: M_1, M_2, M_3, and M_4, where M_1 and M_4 are monotone. Solid and dashed lines represent *must* and *may* transitions, respectively.

in $\gamma(s_1)$ must have a transition to some state in $\gamma(t)$. Since $\gamma(s_1) \supseteq \gamma(s_2)$, the same is true for the states in $\gamma(s_2)$. Therefore, for M to be monotone with this information, it should have a *must* transition from s_2 to t.

For example, the MixTS M_3 shown in Figure 1 is monotone; the MixTS M_1 in the same figure is not monotone. For states a_1 and a_2: $a_2 \preceq_a a_1$ and $a_2 \xrightarrow{\text{must}} a_3$, but there is no *must* transition from a_1 to a_3; and for states a_3 and a_4: $a_4 \preceq_a a_3$ and $a_2 \xrightarrow{\text{may}} a_4$, but there is no *may* transition from a_2 to a_3.

In the rest of this section, we show that every partial TS (or model) can be translated into a semantically equivalent (in the sense of Definition 8) monotone one. We first define such translation for MixTSs. The translation consists of two steps: DSTT (destination translation) and SRCT (source translation) that produce a monotone transition system preserving the behaviors of the original one.

Definition 11 (Translation DSTT). Let $M = \langle S, R_M^{\text{may}}, R_M^{\text{must}} \rangle$ be a MixTS. The result of translation $\text{DSTT}(M)$ is a MixTS $N = \langle S, R_N^{\text{may}}, R_N^{\text{must}} \rangle$, such that

$$R_N^{\text{may}} \triangleq \{(a,b) \in S \times S \mid \exists b' \in S \cdot b' \preceq_a b \wedge (a,b') \in R_M^{\text{may}}\}$$
$$R_N^{\text{must}} \triangleq \{(a,b) \in S \times S \mid \exists b' \in S \cdot b \preceq_a b' \wedge (a,b') \in R_M^{\text{must}}\}$$

The translation $\textsc{DstT}$ checks the transition from each state in its input TS and adds missing transitions derived from the approximation ordering $\preceq_a$ over abstract states, ensuring that the result satisfies condition (a) of Definition 10. A *may* transition is added between states a and b if the source TS has a *may* transition between a and some state b' that is less precise than b. Similarly, a *must* transition between states a and b is added if the source TS has a *must* transition between a and some state b' that is more precise than b. For example, $\textsc{DstT}(M_1)$ results in the MixTS M_2: two new transitions are added, $a_2 \xrightarrow{\text{may}} a_3$ and $a_2 \xrightarrow{\text{must}} a_4$.

Lemma 1. *Let M be a MixTS, and $N = \textsc{DstT}(M)$. Then, N is a MixTS that satisfies condition (a) of Definition 10.*

Definition 12 (Translation SrcT). Let $M = \langle S, R_M^{\text{may}}, R_M^{\text{must}} \rangle$ be a MixTS. The result of the translation $\textsc{SrcT}(G)$ is a MixTS $N = \langle S, R_N^{\text{may}}, R_N^{\text{must}} \rangle$, such that

$$R_N^{\text{may}} \triangleq \{(a,b) \in S \times S \mid \forall a' \in S \cdot a' \preceq_a a \Rightarrow (a',b) \in R_M^{\text{may}}\}$$
$$R_N^{\text{must}} \triangleq \{(a,b) \in S \times S \mid \exists a' \in S \cdot a' \preceq_a a \wedge (a',b) \in R_M^{\text{must}}\}$$

The translation $\textsc{SrcT}$ ensures that its output, N, satisfies condition (b) of Definition 10. It guarantees that the transitions from more precise states are more defined: for each state a, it has a *must* transition to a state b in N if a less precise state a' already has a *must* transition to b in M; it has a *may* transition to b in N only when all the states that are less precise than it already have *may* transitions to b in M. For example, M_3 in Figure 1 is the result of $\textsc{SrcT}(M_2)$: because a_2 is less precise than a_1 and there are *must* transitions $a_2 \xrightarrow{\text{must}} a_3$ and $a_2 \xrightarrow{\text{must}} a_4$ in M_2, two *must* transitions $a_1 \xrightarrow{\text{must}} a_3$ and $a_1 \xrightarrow{\text{must}} a_4$ are added to M_3; on the other hand, the *may* transition $a_1 \xrightarrow{\text{may}} a_2$ is removed from M_3 because a_2 has no *may* transition to a_2 in M_2.

Lemma 2. *Let M be a MixTS, and $N = \textsc{SrcT}(M)$. Then, N is a MixTS that satisfies condition (b) of Definition 10.*

We define the monotone translation $\textsc{MonoT}$ be the composition of the translations for source and destination states: $\textsc{MonoT} \triangleq \textsc{SrcT} \circ \textsc{DstT}$. The following theorem shows that $\textsc{MonoT}$ translates a MixTS into an equivalent monotone one.

Theorem 2. *Let M be a MixTS, and $N = \textsc{MonoT}(M)$. Then, N is a monotone MisTS semantically equivalent to M.*

Proof. (1) Let $N_1 = \textsc{DstT}(M)$ and $N_2 = \textsc{SrcT}(N_1)$. According to Lemmas 1 and 2, N_1 and N_2 satisfy conditions (a) and (b) of Definition 10, respectively. To show that $\textsc{MonoT}(M)$ is monotone, we only need to show that N_2 also satisfies condition (a). Proof of this follows from the definition of $\textsc{SrcT}$.

(2) To prove that M and N_2 are semantically equivalent, we show that any concrete BTS $B = \langle C, R \rangle$ refines M iff it refines N. It is equivalent to showing that (i) the soundness relation $\rho \subseteq C \times S$ is a mixed simulation between B and M iff it is a mixed simulation between B and N_1; and (ii) ρ is a mixed simulation between B and N_1 iff it is a mixed simulation between B and N_2. This follows from the definitions

of DSTT and SRCT. □

The translation MONOT can also be used to convert a partial model into its monotone equivalent.

Corollary 1. *Let $\mathcal{M} = \langle M, L_M \rangle$ be a MixTS model, $N = \text{MONOT}(M)$, and $L_N = L_M$. Then the model $\mathcal{N} = \langle N, L_N \rangle$ is monotone and semantically equivalent to $\mathcal{M}$.*

In this section, we have shown that monotone partial TSs are as expressive as their "regular" counterparts. The monotone conditions make hidden transitions explicit, allowing us to do better local reasoning about partial TSs. This is illustrated in the following sections.

4. Consistency

There are two alternatives for defining consistency of a partial TS: either based on satisfaction of temporal logic formulas (*logical consistency*), or based on possible concrete refinements (*semantic consistency*). While semantic consistency implies logical consistency, the converse is not true. There exists a logically consistent TS that has no concrete refinements. In this section, we investigate these two notions, show when they coincide, and provide a new structural condition which is necessary and sufficient to ensure that a TS is consistent.

4.1. Logical and Semantic Consistency for Consistent Statespaces

Throughout this section, we assume a fixed abstraction relation $\langle C, \rho, S \rangle$. Furthermore, in this subsection, we assume that every state $a \in S$ is consistent, i.e., $\gamma(a) \neq \emptyset$. We extend our definitions to deal with inconsistent states in Section 4.2.

A model $\mathcal{M}$ is logically consistent over a consistent abstract statespace if and only if it gives a consistent interpretation, i.e., either *true*, *false*, or *unknown*, to every temporal formula.

Definition 13. A model $\mathcal{M}$ is *logically consistent* over a consistent abstract statespace iff for every $\varphi \in L_\mu$, $\mathsf{U}(\|\varphi\|_i) \subseteq \mathsf{O}(\|\varphi\|_i)$.

Logical consistency naturally extends from models to transition systems: a transition system M is logically consistent iff for any labeling function L the model $\langle M, L \rangle$ is logically consistent.

A transition system M is semantically consistent iff there exists at least one BTS that refines it:

Definition 14. A transition system M is *semantically consistent* iff $\mathbb{C}[M] \neq \emptyset$.

Semantic consistency extends naturally from transition systems to models. A model $\mathcal{M} = \langle M, L \rangle$ is semantically consistent iff the transition system M is semantically consistent. Because we require that the labeling function L be monotone with respect to $\preceq_a$, this is equivalent to requiring that the model $\mathcal{M}$ has a consistent refinement.

Semantic consistency implies logical consistency:

Theorem 3. *Every semantically consistent transition system is also logically consistent.*

PROOF. Let M be a consistent transition system. We show that M is logically consistent by contradiction.

Assume M is not logically consistent. Then, there exists a labeling function L and a temporal formula φ such that φ is inconsistent in some state of the model $\mathcal{M} = \langle M, L \rangle$. Formally, there exists a state a of M such that a is in $\mathsf{U}(\|\varphi\|_i^{\mathcal{M}}) \setminus \mathsf{O}(\|\varphi\|_i^{\mathcal{M}})$.

Let $\mathcal{B}$ be a concrete (BTS) model refining $\mathcal{M}$. Since $\mathcal{M}$ is semantically consistent, such $\mathcal{B}$ is guaranteed to exist. By Theorem 1, $\gamma(\mathsf{U}(\|\varphi\|_i^{\mathcal{M}})) \subseteq \mathsf{U}(\|\varphi\|_i^{\mathcal{B}})$, and $\gamma(\overline{\mathsf{O}(\|\varphi\|_i^{\mathcal{M}})}) \subseteq \overline{\mathsf{O}(\|\varphi\|_i^{\mathcal{B}})}$. Then, there exists a concrete state $c \in \gamma(a)$ such that $c \in \mathsf{U}(\|\varphi\|_i^{\mathcal{B}})$ and $c \in \overline{\mathsf{O}(\|\varphi\|_i^{\mathcal{B}})}$.

Since $\mathcal{B}$ is concrete, $\mathsf{U}(\|\varphi\|_i^{\mathcal{B}}) = \mathsf{O}(\|\varphi\|_i^{\mathcal{B}})$. Hence, $c \in \mathsf{U}(\|\varphi\|_i^{\mathcal{B}})$ and $c \in C \setminus \mathsf{U}(\|\varphi\|_i^{\mathcal{B}})$ — a contradiction. Thus, $\mathcal{M}$ is logically consistent. □

Interestingly, the converse of Theorem 3 is not true in general. We illustrate this on an example. Consider the MixTS M_2 in Figure 1. In M_2, every *must* transition is matched by a *may* transition, i.e., $R^{\text{must}} \subseteq R^{\text{may}}$. By [22, 13], $R^{\text{must}} \subseteq R^{\text{may}}$ is a sufficient condition for logical consistency. Therefore, M_2 is logically consistent. However, M_2 is not semantically consistent as we show using a proof by contradiction. Assume there is a BTS B that refines M_2. Let $c_1 : \langle x = 1 \rangle$ be a state of B; c_1 is approximated by both a_1 and a_2. Because B refines M_2, and M_2 has a *must* transition $a_2 \xrightarrow{\text{must}} a_3$, B has a transition from c_1 to a state approximated by a_3, say, $c_2 : \langle x = -1 \rangle$. Since M_2 approximates B, by the definition of mixed simulation (Definition 4), a_1 must have a *may* transition to a state that approximates c_2, i.e., either a_3 or a_4. There is no such *may* transition in M_2, contradicting the assumption. Thus, M_2 is not semantically consistent.

Below, we show that monotone MixTSs is a class of systems for which logical and semantic consistency coincide. Intuitively, the reason is that the approximation ordering, $\preceq_a$, of the statespace of monotone MixTSs is “pushed” down to its transitions. This gives rise to the following theorem:

Theorem 4. *Let M be a monotone MixTS $(S, R^{\text{must}}, R^{\text{may}})$, and assume that every state in S is consistent. Then, the following are equivalent:*

(a) M is semantically consistent (Definition 14),

(b) M is logically consistent (Definition 13),

(c) $\forall a, b_1 \in S \cdot a \xrightarrow{\text{must}} b_1 \Rightarrow \exists b_2 \in S \cdot b_1 \preceq_a b_2 \wedge a \xrightarrow{\text{may}} b_2$.

PROOF. We show that $(a) \Rightarrow (b)$, $(b) \Rightarrow (c)$, and $(c) \Rightarrow (a)$.

Part 1. $(a) \Rightarrow (b)$ The proof follows from Theorem 3.

Part 2. $(b) \Rightarrow (c)$ Let a and b_1 be two states in S such that $a \xrightarrow{\text{must}} b_1$ is a transition in R^{must}. We show that (i) there exists a labeling function L, and (ii) there exists a formula φ, such that φ is consistent in the state a of the model $\mathcal{M} = \langle M, L\rangle$ only if M has a transition $a \xrightarrow{\text{may}} b_2$ for some state b_2 that is more precise than b_1.

(i) To define L, we partition the statespace S into sets S_1, S_2, and S_3:

$$\begin{aligned}
S_1 &\triangleq \{s \in S \mid b_1 \preceq_a s\}\\
S_2 &\triangleq \{s \in S \mid \exists t \in S_1 \cdot s \preceq_a t\} \setminus S_1\\
S_3 &\triangleq S \setminus (S_1 \cup S_2)
\end{aligned}$$

S_1 is the set of all states that are more precise than b_1. S_2 is the set of all states that are not in S_1, but are less precise than some state in S_1. S_3 contains all states that are neither in S_1 nor S_2.

Let $AP = \{p\}$. L is defined as follows:

$$L(s) \triangleq \begin{cases} \{p\} & \text{if } s \in S_1\\ \{\} & \text{if } s \in S_2\\ \{\neg p\} & \text{if } s \in S_3. \end{cases}$$

L is consistent. We need to show that L is monotone, i.e., if $s \preceq_a t$ then $L(s) \subseteq L(t)$. Let s and t be two states such that $s \preceq_a t$. Then, either s and t belong to the same partition or $s \in S_2$ and $t \in S_1 \cup S_3$. In both cases, monotonicity follows trivially.

(ii) We define φ as the formula $\Diamond p$. Note that because of the must transition $a \xrightarrow{\text{must}} b_1$, a is in $\mathsf{U}(\|\Diamond p\|_i^{\mathcal{M}})$. And, because $\mathcal{M}$ is logically consistent, $a \in \mathsf{O}(\|\Diamond p\|_i^{\mathcal{M}})$ as well. We use this fact to show existence of b_2, needed for condition (c) of the theorem.

$$\begin{array}{ll}
& a \xrightarrow{\text{must}} b_1\\
\Rightarrow & (\text{by the definition of } L,\ \|p\|_i^{\mathcal{M}} = \langle S_1, S_1 \cup S_2\rangle)\\
& a \xrightarrow{\text{must}} b_1 \wedge b_1 \in \mathsf{U}(\|p\|_i^{\mathcal{M}})\\
\Rightarrow & (\text{by SIS of } \Diamond p)\\
& a \in \mathsf{U}(\|\Diamond p\|_i^{\mathcal{M}})\\
\Rightarrow & (\text{since } \mathcal{M} \text{ is logically consistent, } \Diamond p \text{ is consistent at } a)\\
& a \in \mathsf{O}(\|\Diamond p\|_i^{\mathcal{M}})\\
\Rightarrow & (\text{by SIS of } \Diamond p)\\
& \exists b_2 \in S_1 \cup S_2 \cdot a \xrightarrow{\text{may}} b_2\\
\Rightarrow & (\text{logic})\\
& (\exists b_2 \in S_1 \cdot a \xrightarrow{\text{may}} b_2) \vee (\exists b_2 \in S_2 \cdot a \xrightarrow{\text{may}} b_2)
\end{array}$$

In the first case, $b_2 \in S_1$. By definition of S_1, $b_1 \preceq_a b_2$. This fulfills condition (c) of the theorem.

In the second case, $b_2 \in S_2$.

$$\begin{array}{ll}
& \exists b_2 \in S_2 \cdot a \xrightarrow{\text{may}} b_2\\
\Rightarrow & (\text{by the definition of } S_2)\\
& \exists b_2 \in S_2 \cdot a \xrightarrow{\text{may}} b_2 \wedge \exists b' \in S_1 \cdot b_2 \preceq_a b'\\
\Rightarrow & (\text{by assumption, } M \text{ is monotone})\\
& \exists b' \in S_1 \cdot a \xrightarrow{\text{may}} b'
\end{array}$$

Hence, b' fulfills the condition (c) of the theorem.

Thus, if M is logically consistent, then

$$\forall a, b_1 \in S \cdot a \xrightarrow{\text{must}} b_1 \Rightarrow \exists b_2 \in S \cdot b_1 \preceq_a b_2 \wedge a \xrightarrow{\text{may}} b_2 \,.$$

Part 3. $(c) \Rightarrow (a)$ The proof proceeds by constructing a concrete BTS B that refines M. Let $\langle C, \rho, S \rangle$ be the abstraction relation and $\alpha : C \rightarrow S$ the corresponding abstraction function. Let B be a BTS $\langle C, R \rangle$, where

$$R \triangleq \{(c, d) \in C \times C \mid \exists b \in S \cdot (\alpha(c), b) \in R^{\text{may}} \wedge (d, b) \in \rho\}$$

We show that ρ is a mixed simulation relation between M and B, i.e., $M \preceq_\rho B$. Let $c \in C$, and $a \in S$ be two states such that $(c, a) \in \rho$. Recall that this implies that $a \preceq_a \alpha(c)$.

First, we show that ρ satisfies condition (a) of Definition 4. Let b be a state in M such that there is a must transition $a \xrightarrow{\text{must}} b$. Then,

$$\begin{array}{ll} & (a, b) \in R^{\text{must}} \\ \Rightarrow & \text{(by assumption, } M \text{ is monotone and } a \preceq_a \alpha(c)\text{)} \\ & (\alpha(c), b) \in R^{\text{must}} \\ \Rightarrow & \text{(by assumption of condition (c) of the theorem)} \\ & \exists b' \in S \cdot b \preceq_a b' \wedge (\alpha(c), b') \in R^{\text{may}} \\ \Rightarrow & \text{(by the definition of } B\text{)} \\ & \exists b' \in S \cdot \exists d \in C \cdot b \preceq_a b' \wedge (c, d) \in R \wedge (d, b') \in \rho \\ \Rightarrow & \text{(by monotonicity of } \rho\text{)} \\ & \exists d \in C \cdot (c, d) \in R \wedge (d, b) \in \rho \end{array}$$

Second, we show that ρ satisfies condition (b) of Definition 4. Let d be a state in B such that there is a transition $c \rightarrow d$. Then,

$$\begin{array}{ll} & (c, d) \in R \\ \Rightarrow & \text{(by the definition of } B\text{)} \\ & \exists b \in S \cdot (\alpha(c), b) \in R^{\text{may}} \wedge (d, b) \in \rho \\ \Rightarrow & \text{(by assumption, } M \text{ is monotone, and } a \preceq_a \alpha(c)\text{)} \\ & \exists b \in S \cdot (a, b) \in R^{\text{may}} \wedge (d, b) \in \rho \end{array}$$

Thus, we have constructed a BTS B and produced ρ which is a mixed mixed simulation between M and B. Hence, M is semantically consistent. □

In the rest of this section, we highlight some of the consequences of Theorem 4. First, note that Theorem 4 does not extend to monotone partial models! For example, consider a monotone MixTS M_3 in Figure 1. By Theorem 4, M_3 is inconsistent: there is a *must* transition $a_1 \xrightarrow{\text{must}} a_3$, but no *may* transition $a_1 \xrightarrow{\text{may}} a$ to a state a such that $a_3 \preceq_a a$. Let p be an atomic proposition: "x is a prime number". Let L_3 be a labeling function: for any state s of M_3, $L_3(s) = \emptyset$. That is, p is unknown at all the states in M_3. The model $\mathcal{M}_3 = \langle M_3, L_3 \rangle$ is semantically inconsistent. But, $\mathcal{M}_3$ is logically consistent – there does not exists a formula φ such that $\mathsf{U}(\|\varphi\|_i^{\mathcal{M}_3}) \setminus \mathsf{O}(\|\varphi\|_i^{\mathcal{M}_3}) \neq \emptyset$. Intuitively, the labeling function L_3 is too coarse to detect the inconsistency logically.

Second, part (c) of Theorem 4 gives a necessary and sufficient structural condition for a monotone MixTS to be consistent. Let us compare it with the previously known condition to ensure logical consistency [22, 13]:

$$\forall a, b \in S \cdot (a \xrightarrow{\text{must}} b) \Rightarrow (a \xrightarrow{\text{may}} b)\,.$$

Our new condition is weaker. Thus, there is a consistent monotone MixTS which has a *must* transition that is not a *may* transition. For example, consider the MixTS M_4 in Figure 1. Note that the *must* transition $a_1 \xrightarrow{\text{must}} a_3$ is not matched by any *may* transition. Let B be a BTS $(\mathbb{Z}, R)$, where $\mathbb{Z}$ is the set of integers, and R is defined as follows:

$$R \triangleq \{(x, x') \in \mathbb{Z} \times \mathbb{Z} \mid (x > 0 \wedge x' = -1) \vee (x \leq 0 \wedge x' = x - 2)\}\,.$$

B refines M_4. Thus, by definition, M_4 is semantically consistent. By Theorem 3, M_4 is logically consistent as well.

Third, by definition, a KMTS always satisfies condition (c) of Theorem 4. Existing work on KMTSs [22] often implicitly assumes that the abstract domain is flat (i.e., the abstract ordering $\preceq_a$ on S is discrete). This assumption ensures that every KMTS is monotone. For such TSs, semantic and logical consistency coincide. Yet the assumption about the flatness of the abstract domain is too restrictive. For example, it is not true in a typical application of predicate abstraction (e.g., in [18]). By looking at a wider range of transition systems and considering not only flat abstract domains, we have uncovered the subtle but important differences between logical and semantic consistency.

4.2. Logical and Semantic Consistency for Arbitrary Statespaces

In Section 4.1, we have assumed that the abstract statespace S does not contain any inconsistent states. That is, if a is in S, then its concretization $\gamma(a)$ is non-empty. We now lift this restriction, i.e., we aim to redefine (i) logical consistency, (ii) semantic consistency and (iii) the structural condition of Theorem 4.

(i) An inconsistent state does not abstract any concrete states, so a temporal formula can have any value in that state, including being both satisfied and refuted. We thus strengthen Definition 13 as follows:

Definition 15. A model $\mathcal{M}$ is *logically consistent* iff for every $\varphi \in L_\mu$,

$$a \in (\mathsf{U}(\|\varphi\|_i) \setminus \mathsf{O}(\|\varphi\|_i)) \Rightarrow \gamma(a) = \emptyset\,.$$

If the abstract statespace S has no inconsistent states, this definition reduces to Definition 13.

(ii) Semantic consistency does not need a new definition: a transition system is *semantically consistent* iff there is a BTS that refines it, independently of the structure of the abstract statespace.

(iii) We now need to strengthen the structural condition to match the new Definition 15. Specifically, we add the requirement that every *must* transition from a *consistent* state must be matched by a *may* transition into a *consistent* state.

Under these conditions, we now restate Theorem 4 to handle inconsistent states:

Theorem 5. *Let $M = \langle S, R^{\text{must}}, R^{\text{may}} \rangle$ be a monotone MixTS. Then, the following are equivalent:*

(a) M is semantically consistent (Definition 14),

(b) M is logically consistent (Definition 15),

(c) $$\begin{array}{l}\forall a, b_1 \in S \cdot (\gamma(a) \neq \emptyset \wedge a \xrightarrow{\text{must}} b_1) \Rightarrow \\ \qquad (\exists b_2 \in S \cdot b_1 \preceq_a b_2 \wedge \gamma(b_2) \neq \emptyset \wedge a \xrightarrow{\text{may}} b_2).\end{array}$$

In this section, we have investigated the connection between semantic and logical consistency of partial models. Semantic consistency is important for when partial TSs are used as objects for abstracting concrete TSs. Logical consistency is important when partial models are used to interpret temporal logic formulas. In the following two sections, we first compare the expressive power of the different TS formalisms, i.e., what can be modeled and what abstractions can be captured (Section 5). Second, we compare the analyzability of the formalisms, i.e., the cost and precision of model checking (Section 6).

5. Expressiveness

We show that GKMTSs, MixTSs, and KMTSs are expressively equivalent (in the sense of Definition 9). The equivalence of the three formalisms is proved by defining semantics-preserving translations from GKMTSs to MixTSs, and from MixTSs to KMTSs. Since GKMTSs syntactically subsume KMTSs, the translation from KMTSs to GKMTSs is basically an identity map.

5.1. GTOM*: Translation from GKMTSs to MixTSs*

We present the translation GTOM that converts a GKMTS into a semantically equivalent MixTS. First, we illustrate the translation on a GKMTS G_1 in Figure 2. G_1 is not a MixTS because of *must* hyper-transition $a_1 \xrightarrow{\text{must}} \{a_2, a_3\}$. This transition ensures that in every concrete BTS refining G_1, all states in $\gamma(a_1)$, i.e., those satisfying $(x \leq 0 \wedge even(x))$, must have a transition to a state in $\gamma(\{a_2, a_3\})$, i.e., satisfying $(x > 0)$. No single state of G_1 represents $(x > 0)$. Thus, this requirement can only be captured either by a hyper transition (as done in G_1), or by extending G_1 with a new state, say a_5, such that $\gamma(a_5) = (x > 0)$. In the latter case, the *must* hyper-transition $a_1 \xrightarrow{\text{must}} \{a_2, a_3\}$ can be replaced by (regular) *must* transition $a_1 \xrightarrow{\text{must}} a_5$. The result is a MixTS M_5 in Figure 2. Since a_5 replaces a "hyper-state" $\{a_2, a_3\}$, a_5 needs to preserve its *may* behaviours. This is done by adding $a_5 \xrightarrow{\text{may}} a_4$ and $a_5 \xrightarrow{\text{may}} a_2$ corresponding to $a_2 \xrightarrow{\text{may}} a_4$ and $a_3 \xrightarrow{\text{may}} a_2$, respectively. There are no outgoing *must* transitions from a_5 since the existing *must* transitions from a_2 and a_3 are sufficient. G_1

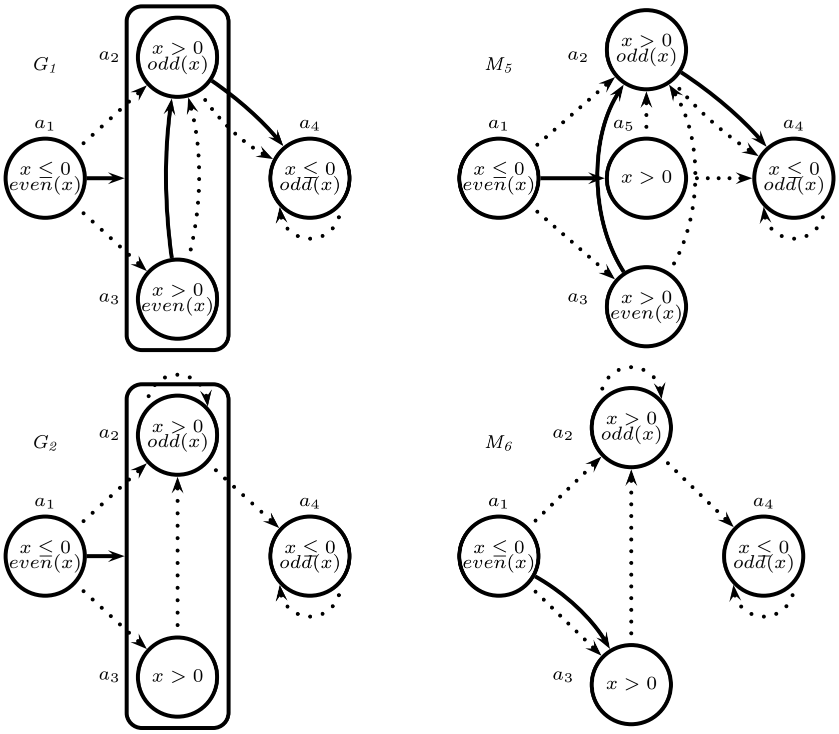

Figure 2: Two GKMTSs: G_1, G_2, and two MixTSs: M_5, M_6, where G_1 and G_2 are semantically equivalent to M_5 and M_6, respectively.

and M_5 are semantically equivalent: any BTS that refines G_1 also refines M_5, and vice versa.

In our example, a new state was added to encode a hyper-transition by a regular one. This isn't always necessary. For example, TSs G_2 and M_6 in Figure 2 are semantically equivalent. The hyper-transition $a_1 \xrightarrow{\text{must}} \{a_2, a_3\}$ is encoded by $a_1 \xrightarrow{\text{must}} a_3$ in M_6 since the hyper-state $\{a_2, a_3\}$ is equivalent to an existing state a_3, i.e., $\gamma(\{a_2, a_3\}) = \gamma(a_3) = (x > 0)$.

In summary, a GKMTS G is translated to a MixTS M in two steps: (i) every *must* hyper-transition $a \xrightarrow{\text{must}} U$ of G is replaced by a regular *must* transition $a \xrightarrow{\text{must}} b$, where b is a (possibly new) state such that $\gamma(b) = \gamma(U)$; (ii) *may* transitions are added for every state introduced in the first step, if any. We formalize this below.

Definition 16 (GTOM). Let $G = \langle S_G, R_G^{\text{may}}, R_G^{\text{must}} \rangle$ be a GKMTS. The translation

GTOM(G) is a MixTS $M = \langle S_M, R_M^{\text{must}}, R_M^{\text{may}} \rangle$, such that

$$\begin{aligned} S_M &\triangleq S_G \ \cup \ S^+ \\ S^+ &\triangleq \{a \mid \exists (s,U) \in R_G^{\text{must}} \cdot \gamma(a) = \gamma(U) \land (\forall t \in S_G \cdot \gamma(t) \neq \gamma(U))\} \\ R_M^{\text{may}} &\triangleq R_G^{\text{may}} \cup \{(a,b) \mid a \in S^+ \land b \in S_G \land \exists s \in S_G \cdot (s,b) \in R_G^{\text{may}} \land \gamma(s) \subseteq \gamma(a)\} \\ R_M^{\text{must}} &\triangleq \{(a,b) \mid a \in S_G \ \land \ b \in S_M \land \exists U \subseteq S_G \cdot (a,U) \in R_G^{\text{must}} \ \land \ \gamma(b) = \gamma(U)\} \end{aligned}$$

The theorem below shows that the translation GTOM is semantics-preserving.

Theorem 6. *Let G be a GKMTS, and $M =$ GTOM(G). Then, M is a MixTS, and G and M are semantically equivalent.*

PROOF. (1) According to the construction in Definition 16, every must hypertransition is replaced by a regular one. Therefore, M is a MixTS. (2) To prove that G and M are semantically equivalent, we show that any concrete BTS $B = \langle C, R \rangle$ refines G iff it refines M. It is equivalent to showing that the soundness relation $\rho_G \subseteq C \times S_G$ is a mixed simulation between B and G iff the soundness relation $\rho_M \subseteq C \times S_M$ is a mixed simulation between B and M. This follows from the construction of transition relations given in Definition 16. □

A corollary of Theorem 6 is that GKMTSs and MixTSs are equivalent with respect to thorough semantics. Let L_G be a labeling function for G. We extend the translation GTOM to a GKMTS model $\langle G, L_G \rangle$ such that GTOM$(\langle G, L_G \rangle) \triangleq \langle M, L_M \rangle$, where $M =$ GTOM(G), and L_M is a labeling function for S_M defined as follows:

$$L_M(a) \triangleq \begin{cases} L_G(a) & \text{if } a \in S_G \\ \bigcap_{\{s \in S_G \mid \gamma(s) \subseteq \gamma(a)\}} L_G(s) & \text{if } a \in S^+ \end{cases}$$

That is, if a is a state belonging to the original statespace S_G, the labels on a are the same as before. For a new state a added by the translation, since the concrete states approximated by a are the *union* of the ones approximated by a set of states in S_G, the labels on a are the literals that are true in all the concrete states; therefore, $L_M(a)$ is defined as the intersection of the labels on the states in S_G that are more precise than a.

Theorem 7. *The state labeling L_M above is well-defined and approximates the same labelings as L_G.*

PROOF. The proof follows immediately from the approximation defined for state labeling and construction of L_M. □

As a result, $\langle G, L_G \rangle$ and $\langle M, L_M \rangle$ satisfy the same properties under thorough semantics.

Corollary 2. *Let $\langle G, L_G \rangle$ be a GKMTS model and $\langle M, L_M \rangle =$ GTOM$(\langle G, L_G \rangle)$. Then, $\langle G, L_G \rangle$ and $\langle M, L_M \rangle$ are equivalent w.r.t. thorough semantics.*

Complexity. We show that the translation GTOM does not increase the size of the model. Let G be a GKMTS with the statespace S_G, and $M = \text{GTOM}(G)$. The size of G is at most $|S_G \times 2^{S_G}|$. Each new state added by GTOM corresponds to a subset of S_G, i.e., $|S^+| \leq |2^{S_G}|$. Furthermore, no transitions between the states in S^+ are added. Thus, the size of M is also at most $|S_G \times 2^{S_G}|$.

Sometimes GTOM can reduce a GKMTS exponentially. For example, assume that S_G is a disjunctive completion [9], i.e., for every subset U of S_G there exists an equivalent element s in S_G such that $\gamma(U) = \gamma(s)$. In this case, GTOM does not add any new states, i.e., $S^+ = \emptyset$. This makes the size of the output MixTSs be $|S_G \times S_G|$, which is exponentially smaller than that of the input GKMTS.

5.2. MTOK*: Translation from MixTSs to KMTSs*

We present the translation MTOK that converts a MixTS into a semantically equivalent KMTS. First, we illustrate the translation using a MixTS M_7 in Figure 3. M_7 is not a KMTS because of the two *must only* transitions, $a_1 \xrightarrow{\text{must}} a_2$ and $a_2 \xrightarrow{\text{must}} a_4$. One way to turn M_7 into a KMTS is to add *may* transitions $a_1 \xrightarrow{\text{may}} a_2$ and $a_2 \xrightarrow{\text{may}} a_4$, resulting in K_1 in Figure 3. This naive transformation is not semantics-preserving, i.e., K_1 and M_7 are not semantically equivalent. For example, the concrete system

$$\begin{aligned}((y > 0) \wedge (x > 0) \wedge odd(x) \wedge x' = x + 1 \wedge y' = y) \vee \\ ((x > 0) \wedge odd(x) \wedge x' = x \wedge y' = -1 \times x) \vee \\ ((x > 0) \wedge \neg odd(x) \wedge x' = x + 1 \wedge y' = -1 \times x)\end{aligned}$$

refines K_1, but not M_7: the transition $\langle x = 1, y = 1\rangle \rightarrow \langle x = 2, y = 1\rangle$ cannot be simulated by any *may* transition of M_7 from a_1.

The *must only* transition $a_1 \xrightarrow{\text{must}} a_2$ of M_7 ensures that in any concrete BTS refining M_7, all states in $\gamma(a_1)$, i.e., those satisfying $(x > 0 \wedge odd(x) \wedge y > 0)$, must have a transition to a state in $\gamma(a_2)$, i.e., satisfying $(x > 0)$. This is further restricted by the *may* transitions from a_1 that ensure that states in $\gamma(a_1)$ have transitions only to states in $\gamma(\{a_1, a_3\})$. Hence, in any BTS refining M_7, every state in $\gamma(a_1)$ must (and may) have a transition to a state in $\gamma(a_2) \cap \gamma(\{a_1, a_3\})$. That is, the restrictions posed by a *must only* transition from a_1 are further restricted by the set of all of the *may* transitions from a_1. In general, for abstract states $b_0, \ldots, b_k$, a *must only* transition $b_0 \xrightarrow{\text{must}} b_1$, and a set of *may* transitions $b_0 \xrightarrow{\text{may}} b_2,\ \ldots,\ b_0 \xrightarrow{\text{may}} b_k$ ensure that every state in $\gamma(b_0)$ has a transition to a state in $\gamma(b_1) \cap \gamma(\{b_2, \ldots, b_k\})$.

The *must only* transition $a_2 \xrightarrow{\text{must}} a_4$ in M_7 is equivalent to a pair of *may* and *must* transitions from a_2 to a_4, since $\gamma(a_4) \cap \gamma(\{a_1, a_2, a_3\}) = \gamma(a_4)$. The *must only* transition $a_1 \xrightarrow{\text{must}} a_2$ can be equivalently represented by (a) adding a new state a_5 such that $\gamma(a_5) = \gamma(a_2) \cap \gamma(\{a_1, a_3\}) = (x > 0 \wedge odd(x))$, and (b) adding a *must* and a *may* transition from a_1 to a_5. Moreover, since a_5 approximates some of the same states as a_2, i.e., $\gamma(a_5) \subseteq \gamma(a_2)$, a_5 inherits the transitions from a_2: $a_5 \xrightarrow{\text{may}} a_1$, $a_5 \xrightarrow{\text{may}} a_2$, $a_5 \xrightarrow{\text{may}} a_3$, $a_5 \xrightarrow{\text{must}} a_4$, $a_5 \xrightarrow{\text{may}} a_4$. The final result is the KMTS K_2 in Figure 3, which is semantically equivalent to M_7.

In summary, a MixTS M is translated to a KMTS K in two steps. First, every *must only* transition $a \xrightarrow{\text{must}} b$ of M is replaced by a pair of *must* and *may* transitions

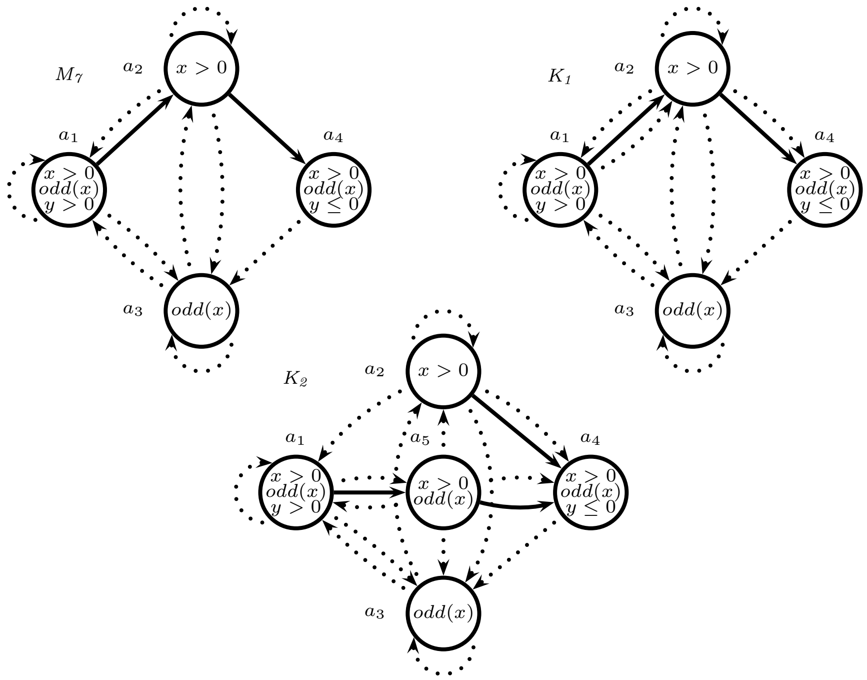

Figure 3: One MixTSs: M_7, and two KMTSs: K_1, K_2, where M_7 and K_4 are semantically equivalent.

$a \xrightarrow{\text{must}} \widehat{a \to b}$ and $a \xrightarrow{\text{may}} \widehat{a \to b}$, where $\widehat{a \to b}$ is a (possibly new) abstract state such that $\gamma(\widehat{a \to b}) = \gamma(b) \cap \gamma(R_M^{\text{may}}(a))$. Second, *may* and *must* transitions are added for all states introduced in the first step. We formalize this below.

Definition 17 (MTOK). Let $M = \langle S_M, R_M^{\text{may}}, R_M^{\text{must}} \rangle$ be a MixTS. The translation MTOK(M) is a KMTS $K = \langle S_K, R_K^{\text{may}}, R_K^{\text{must}} \rangle$, such that

$$\begin{aligned} S_K &\triangleq S_M \cup S^+ \\ R_K^{\text{may}} &\triangleq R_M^{\text{may}} \cup \text{REPL} \cup \text{IMAY} \cup \text{IMO} \\ R_K^{\text{must}} &\triangleq (R_M^{\text{must}} \cap R_M^{\text{may}}) \cup \text{REPL} \cup \text{IMUST} \cup \text{IMO}, \end{aligned}$$

where

$$\begin{aligned} S^+ \quad &\triangleq \{\widehat{a \to b} \mid \exists (a,b) \in (R_M^{\text{must}} \setminus R_M^{\text{may}}) \cdot \forall s \in S_M \cdot \gamma(s) \neq \gamma(\widehat{a \to b})\} \\ \text{REPL} \quad &\triangleq \{(a, \widehat{a \to b}) \mid \exists (a,b) \in (R_M^{\text{must}} \setminus R_M^{\text{may}})\} \\ \text{IMAY} \quad &\triangleq \{(\widehat{a \to b}, b') \mid \exists a, b, b' \in S_M \cdot \\ &\qquad\qquad (a,b) \in (R_M^{\text{must}} \setminus R_M^{\text{may}}) \wedge (b, b') \in R_M^{\text{may}} \wedge \widehat{a \to b} \in S^+\} \\ \text{IMUST} &\triangleq \{(\widehat{a \to b}, b') \mid \exists a, b, b' \in S_M \cdot \\ &\qquad (a,b) \in (R_M^{\text{must}} \setminus R_M^{\text{may}}) \wedge (b, b') \in (R_M^{\text{must}} \cap R_M^{\text{may}}) \wedge \widehat{a \to b} \in S^+\} \\ \text{IMO} \quad &\triangleq \{(\widehat{a \to b}, \widehat{b \to b'} \mid \exists a, b, b' \in S_M \cdot \\ &\qquad\qquad (a,b), (b,b') \in (R_M^{\text{must}} \setminus R_M^{\text{may}}) \wedge \widehat{a \to b} \in S^+\} \end{aligned}$$

In Definition 17, REPL denotes transitions that replace *must only* transitions, and IMAY, IMUST and IMO denote transitions from newly added states in S^+ that correspond to *may*, *must*, and *must only* transitions of the original system, respectively. In our example of MTOK(M_7), we have

$$\begin{array}{lcl} S^+ & = & \{a_5\}, \\ \text{REPL} & = & \{(a_1, a_5), (a_2, a_4)\}, \\ \text{IMUST} & = & \emptyset, \\ \text{IMO} & = & \{(a_5, a_4)\}, \\ \text{IMAY} & = & \{(a_5, a_1), (a_5, a_2), (a_5, a_3)\}. \end{array}$$

The result of the translation MTOK is a KMTS: every *must* transition is matched by a *may* transition.

Theorem 8. *Let M be a MixTS, and $K = \text{MTOK}(M)$. Then K is a KMTS, and M and K are semantically equivalent.*

PROOF. (1) The construction in Definition 17 ensures that every *must* transition in K is matched by a *may* transition. Therefore, K is a KMTS. (2) To prove that M and K are semantically equivalent, we show that for any concrete BTS $B = \langle C, R \rangle$, the soundness relation $\rho_M \subseteq C \times S_M$ is a mixed simulation between B and M iff the soundness relation $\rho_K \subseteq C \times S_K$ is a mixed simulation between B and K. This follows from the construction of transition relations in Definition 17. □

A corollary of Theorem 8 is that MixTSs and KMTSs are equivalent with respect to thorough semantics. Let L_M be a labeling function for M. We extend MTOK to $\langle M, L_M \rangle$ such that $\text{MTOK}(\langle M, L_M \rangle) \triangleq \langle K, L_K \rangle$, where $K = \text{MTOK}(M)$, and L_K is a labeling function for S_K defined as follows:

$$L_K(a) \triangleq \begin{cases} L_M(a) & \text{if } a \in S_M \\ \bigcup_{\{s \in S_M \mid \gamma(a) \subseteq \gamma(s)\}} L_M(s) & \text{if } a \in S^+ \end{cases}$$

In this case, if a is a new state added by the translation, the concrete states approximated by a correspond to the *intersection* of the concrete states approximated by a set of states in S_G; the labels on a are all the literals which are true on the concrete states. Therefore, $L_K(a)$ is defined as the union of the labels on the states in S_M that are less precise than a.

Theorem 9. *The state labeling L_K above is well-defined and approximates the same labelings as L_M.*

PROOF. The proof immediately follows from the approximation defined for state labeling and the construction of L_K. □

As a result, $\langle M, L_M \rangle$ and $\langle K, L_K \rangle$ satisfy the same properties under thorough semantics.

Corollary 3. *Let $\langle M, L_M \rangle$ be a MixTS model and $\langle K, L_K \rangle = \text{MTOK}(\langle M, L_M \rangle)$. Then, $\langle M, L_M \rangle$ and $\langle K, L_K \rangle$ are equivalent w.r.t. thorough semantics.*

Complexity. Let $M = \langle S_M, R_M^{\text{may}}, R_M^{\text{must}} \rangle$ be a MixTS, and K be a KMTS such that $K = \text{MtoK}(M)$. The size of M is bounded by $O(|S_M \times S_M|)$. In the worst case, the translation adds a new state for each *must only* transition in $R_M^{\text{must}} \setminus R_M^{\text{may}}$. Thus, the number of new states $|S^+|$ is bounded by $|S_M \times S_M|$, and $|K|$ is bounded by $O(|S_M \times S_M|^2)$.

MixTSs are more succinct than KMTSs: over a fixed statespace S, the set of MixTSs is more expressive than the set of KMTSs. This holds because S^+ may not be empty in some cases, i.e., new states have to be added by MtoK. The following theorem shows that if S is a powerset abstract domain [5], then MtoK does not add new states, and therefore, MixTS and KMTSs over S are equally expressive.

Theorem 10. *Let $\langle C, \rho, S \rangle$ be an abstraction relation. For any abstract state $a \in S$ and a subset $Q \subseteq S$, there exists a subset $V \subseteq S$ such that $\gamma(V) = \gamma(a) \cap \gamma(Q)$.*

Proof. Let $V \triangleq \{b \in S \mid \exists c \cdot c \in \gamma(a) \cap \gamma(Q) \wedge b = \alpha(c)\}$. The proof of $\gamma(V) \supseteq \gamma(a) \cap \gamma(Q)$ follows from the definition of V. To prove $\gamma(V) \subseteq \gamma(a) \cap \gamma(Q)$, we show that for each $b \in V$, $\gamma(b) \subseteq \gamma(a)$ and $\gamma(b) \subseteq \gamma(Q)$, which follows from the definition of abstraction function. $\square$

6. Reduced Inductive Semantics

GKMTSs and MixTSs are equally expressive: a GKMTS model and its equivalent MixTS model satisfy the same properties under thorough semantics. However, thorough model checking is expensive. In practice, model checking of partial models is done with respect to a more tractable inductive semantics, SIS. GKMTSs are more precise than MixTSs with respect to SIS: for any $\varphi \in L_\mu$, model checking φ in a GKMTS model $\mathcal{G}$ with respect to SIS is more precise than model checking it in the MixTS model $\mathcal{M} = \text{GtoM}(\mathcal{G})$. However, the direct use of GKMTSs in symbolic model checkers has been hampered by the difficulty of encoding hyper-transitions into BDDs. In this section, we propose a new semantics, called *reduced inductive semantics* (RIS), that is inductive while being strictly more precise than SIS. We show that GKMTSs and MixTSs are equivalent with respect to RIS. In Section 7, we give an efficient symbolic model checking procedure for computing RIS over MixTSs. This results in an algorithm that combines the benefits of the efficient symbolic encoding of MixTSs with the model checking precision of GKMTSs.

In Section 6.1, we illustrate the differences between GKMTSs and MixTSs with respect to SIS. We define RIS in Section 6.2, and show how to perform model checking with respect to RIS effectively in Section 6.3.

6.1. Example

Let p and q denote predicates $(x > 0)$ and $odd(x)$, respectively. Consider the model $\mathcal{G}_1 = \langle G_1, L_{G_1} \rangle$ in Figure 4, where G_1 is shown in Figure 2, and L_{G_1} is a labeling function that labels each abstract state as follows:

$$L_{G_1}(a_1) = \{\neg p, \neg q\} \qquad L_{G_1}(a_2) = \{p, q\}$$
$$L_{G_1}(a_3) = \{p, \neg q\} \qquad L_{G_1}(a_4) = \{\neg p, q\}\,.$$

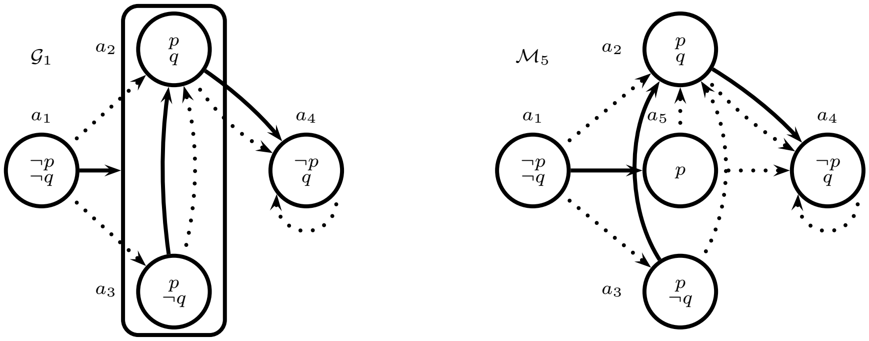

Figure 4: Two models: $\mathcal{G}_1$ and $\mathcal{M}_5$.

Let $\mathcal{M}_5 = \langle M_5, L_{M_5} \rangle = \text{GTOM}(\mathcal{G}_1)$ be the model obtained from $\mathcal{G}_1$ by GTOM. The model $\mathcal{M}_5$ is shown in Figure 4, its underlying transition system M_5 is shown in Figure 2, and

$$L_{M_5}(s) \triangleq \textbf{if } s = a_5 \textbf{ then } \{p\} \textbf{ else } L_{G_1}(s)\,.$$

Compare the value of $\varphi \triangleq \Diamond(q \vee \neg q)$ under SIS on $\mathcal{G}_1$ and $\mathcal{M}_5$:

$$\begin{array}{rcl} \|\varphi\|_i^{\mathcal{G}_1} & = & \langle \{a_1, a_2, a_3\}, \{a_1, a_2, a_3, a_4\} \rangle \\ \|\varphi\|_i^{\mathcal{M}_5} & = & \langle \{a_2, a_3\}, \{a_1, a_2, a_3, a_4, a_5\} \rangle \end{array}$$

According to $\mathcal{G}_1$, φ is true in all states corresponding to a_1. According to $\mathcal{M}_5$, the value of φ is unknown in exactly the same states. Since $\mathcal{M}_5 = \text{GTOM}(\mathcal{G}_1)$, $\mathcal{G}_1$ and $\mathcal{M}_5$ are semantically equivalent. Thus, although $\mathcal{M}_5$ and $\mathcal{G}_1$ are semantically equivalent, $\mathcal{M}_5$ is less precise than $\mathcal{G}_1$ for model checking with respect to SIS.

Let us reexamine the above example. First, there is no precision loss during the evaluation of $q \vee \neg q$:

$$\begin{array}{l} e_1 = \|q \vee \neg q\|_i^{\mathcal{G}_1} = \langle \{a_1, a_2, a_3, a_4\}, \{a_1, a_2, a_3, a_4\} \rangle \\ e_2 = \|q \vee \neg q\|_i^{\mathcal{M}_5} = \langle \{a_1, a_2, a_3, a_4\}, \{a_1, a_2, a_3, a_4, a_5\} \rangle \end{array} \qquad (\star)$$

Since $\gamma(\mathsf{U}(e_1)) = \gamma(\mathsf{U}(e_2))$ and $\gamma(\overline{\mathsf{O}(e_1)}) = \gamma(\overline{\mathsf{O}(e_2)}) = \gamma(\emptyset)$, $e_1 \equiv_a e_2$. However, there is a subtle difference between e_1 and e_2. In state a_5 of M_5, $q \vee \neg q$ is unknown even though it is true in both a_2 and a_3, and $\gamma(a_5) = \gamma(a_2) \cup \gamma(a_3)$. This minor imprecision is then magnified by the $\Diamond$ operator.

This loss of precision is not limited to tautologies. For example, a formula $\mu Z \cdot (\neg p \wedge q) \vee \Diamond Z$, i.e., $EF(\neg p \wedge q)$ in CTL, is true in state a_1 of $\mathcal{G}_1$, but is unknown in the same state of $\mathcal{M}_5$.

6.2. Reduced Inductive Semantics for Partial Models

In this section, we define the reduced inductive semantics (RIS). The new semantics is inductive and is *strictly more precise* than SIS. The key idea is to eliminate any local imprecision by using a special *reduction* operator, defined below:

Reduction Operator. Let $\langle C, \rho, S\rangle$ be an abstraction relation, and let $e, e' \in 2^S \times 2^S$. Recall that in the information order e is less than e', i.e., $e \preceq_i e'$, if $\mathsf{U}(e)$ is contained in $\mathsf{U}(e')$, and $\mathsf{O}(e)$ contains $\mathsf{O}(e')$. We define the *reduction* operator as follows:

$$\mathtt{RED}(\langle U, O\rangle) \triangleq \langle \mathtt{RED}_{\mathsf{U}}(U), \mathtt{RED}_{\mathsf{O}}(O)\rangle$$

where $\mathtt{RED}_{\mathsf{U}}(U) \triangleq \{s \mid \gamma(s) \subseteq \gamma(U)\}$ and $\mathtt{RED}_{\mathsf{O}}(O) \triangleq \{s \mid \gamma(s) \not\subseteq \gamma(\overline{O})\}$. Intuitively, for $e = \langle U, O\rangle$, $\mathtt{RED}(e)$ increases U and decreases O as much as possible without affecting the semantic meaning of e. That is, $\gamma(\mathtt{RED}_{\mathsf{U}}(U)) = \gamma(U)$ and $\gamma(\mathtt{RED}_{\mathsf{O}}(O)) = \gamma(O)$. Therefore, $\mathtt{RED}(e)$ is the largest element with respect to information ordering that is semantically equivalent to e, i.e., $\mathtt{RED}(e) \equiv_a e$.

For example, consider $\mathtt{RED}(e_2)$, where e_2 is as defined by ($\star$) above. Then,

$$e_3 = \mathtt{RED}(e_2) = \langle\{a_1, a_2, a_3, a_4, a_5\}, \{a_1, a_2, a_3, a_4, a_5\}\rangle \tag{$\star\star$}$$

e_3 differs from e_2 only in the addition of a_5 to $\mathsf{U}(e_3)$. Since $\gamma(\mathsf{U}(e_2)) = \gamma(\mathsf{U}(e_3))$ and $\gamma(\overline{\mathsf{O}(e_2)}) = \gamma(\overline{\mathsf{O}(e_3)})$, $e_2 \equiv_a e_3$; but e_3 is more informative since $\mathsf{U}(e_2) \subset \mathsf{U}(e_3)$.

An element $e = \langle U, O\rangle \in 2^S \times 2^S$ is *monotone* iff

$$s_1 \preceq_a s_2 \Rightarrow (s_1 \in U \Rightarrow s_2 \in U \wedge s_1 \notin O \Rightarrow s_2 \notin O)$$

That is, U and $\overline{O}$ are closed under more precise states. The monotonicity of elements is preserved under propositional operations: if e and e' are monotone elements, so are ${\sim}e$ and $e \sqcap e'$. Moreover, $\mathtt{RED}(e)$ is monotone for any e, and it acts homomorphically with respect to propositional operations on monotone elements. That is, let e and e' be monotone elements of $2^S \times 2^S$. Then, ${\sim}e \equiv_a {\sim}\mathtt{RED}(e)$, and $e \sqcap e' \equiv_a \mathtt{RED}(e) \sqcap \mathtt{RED}(e')$.

Reduced Inductive Semantics. RIS is defined by applying the RED operator before and after $\Diamond$ to prevent it from propagating imprecision.

Definition 18 (RIS). Let $\langle C, \rho, S\rangle$ be an abstraction relation, and let $\mathcal{M} = \langle M, L\rangle$ be a model, such that $M = \langle S, R^{\text{may}}, R^{\text{must}}\rangle$ and $\sigma : Var \rightarrow 2^S \times 2^S$. The *reduced inductive semantics* of $\varphi \in L_\mu$ is defined as follows:

$$\begin{array}{rcl}
||p||^{\mathcal{M}}_{r,\sigma} & \triangleq & \langle\{s \mid p \in L(s)\}, \{s \mid \neg p \notin L(s)\}\rangle \\
||\neg\varphi||^{\mathcal{M}}_{r,\sigma} & \triangleq & {\sim}||\varphi||^{\mathcal{M}}_{r,\sigma} \\
||\varphi \wedge \psi||^{\mathcal{M}}_{r,\sigma} & \triangleq & ||\varphi||^{\mathcal{M}}_{r,\sigma} \sqcap ||\psi||^{\mathcal{M}}_{r,\sigma} \\
||\Diamond\varphi||^{\mathcal{M}}_{r,\sigma} & \triangleq & \mathtt{RED}(\langle pre_{\mathsf{U}}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(||\varphi||^{\mathcal{M}}_{r,\sigma}))), pre_{\mathsf{O}}(\mathtt{RED}_{\mathsf{O}}(\mathsf{O}(||\varphi||^{\mathcal{M}}_{r,\sigma})))\rangle) \\
||Z||^{\mathcal{M}}_{r,\sigma} & \triangleq & \sigma(Z) \\
||\mu Z \cdot \varphi||^{\mathcal{M}}_{r,\sigma} & \triangleq & \langle \text{lfp}^{\sqsubseteq}\left(\lambda Q \cdot \mathsf{U}(||\varphi||^{\mathcal{M}}_{r,\sigma[Z \mapsto Q]})\right), \text{lfp}^{\sqsubseteq}\left(\lambda Q \cdot \mathsf{O}(||\varphi||^{\mathcal{M}}_{r,\sigma[Z \mapsto Q]})\right)\rangle
\end{array}$$

The only difference between RIS (Definition 18) and SIS (Definition 2) is the semantics of $\Diamond$, where the RED operator in RIS uses abstraction information to improve precision. Since we assume that a state labeling is monotone, applying RED to other operators as well does not improve precision.

We now show that RIS is sound.

Theorem 11. *Let $\langle C, \rho, S\rangle$ be an abstraction relation, $\mathcal{M} = \langle M, L_M\rangle$ be a partial model over S, and $\mathcal{B} = \langle B, L_B\rangle$ be a concrete model over C. If $\mathcal{M}$ approximates $\mathcal{B}$, then, for any L_μ formula φ,*

$$\gamma(\mathsf{U}(\|\varphi\|_r^{\mathcal{M}})) \subseteq \mathsf{U}(\|\varphi\|_r^{\mathcal{B}}) \qquad \textit{and} \qquad \gamma(\overline{\mathsf{O}(\|\varphi\|_r^{\mathcal{M}})}) \subseteq \overline{\mathsf{O}(\|\varphi\|_r^{\mathcal{B}})}\ .$$

Proof:
The only difference between RIS and SIS is the application of the RED operator before and after $\Diamond$. Since RED is semantics-preserving, the result holds following Theorem 1.
□

Returning to our running example, RIS of φ on $\mathcal{M}_5$ is computed as follows: RIS of q, $\neg q$, and $q \vee \neg q$ is the same as SIS. Thus, $\|q \vee \neg q\|_r^{\mathcal{M}_5} = e_2$. To compute $\Diamond$, recall from ($\star\star$) that $\mathtt{RED}(e_2) = e_3$; thus, $\|\varphi\|_r^{\mathcal{M}_5} = \langle\{a_1, a_2, a_3, a_5\}, \{a_1, a_2, a_3, a_4, a_5\}\rangle$. Hence, $\|\varphi\|_r^{\mathcal{M}_5}$ is more precise than $\|\varphi\|_i^{\mathcal{M}_1}$.

Theorem 12. *RIS is more precise than SIS: $\|\varphi\|_i \preceq_a \|\varphi\|_r$.*

PROOF. We begin by fixing an abstraction relation $\langle C, \rho, S\rangle$. The proof proceeds by structural induction on φ. For the base case, it is obvious that for any atomic proposition p, $\|p\|_i \equiv_a \|p\|_r$. In the following, we show the inductive case for $\Diamond\varphi$; the proofs of other cases are trivial.

We show that $||\varphi||_i \preceq_a ||\varphi||_r \Rightarrow ||\Diamond\varphi||_i \preceq_a ||\Diamond\varphi||_r$, which is equivalent to proving the following two statements:

$$(a)\ \ ||\varphi||_i \preceq_a ||\varphi||_r \Rightarrow \gamma(\mathsf{U}(||\Diamond\varphi||_i)) \subseteq \gamma(\mathsf{U}(||\Diamond\varphi||_r))$$

$$(b)\ \ ||\varphi||_i \preceq_a ||\varphi||_r \Rightarrow \gamma(\overline{\mathsf{O}(||\Diamond\varphi||_i)}) \subseteq \gamma(\overline{\mathsf{O}(||\Diamond\varphi||_r)})$$

The proof of (a) is as follows. First, note that for any two sets Q_1, Q_2, we have that

$$\gamma(Q_1) \subseteq \gamma(\mathtt{RED}_{\mathsf{U}}(Q_2)) \Rightarrow Q_1 \subseteq \mathtt{RED}_{\mathsf{U}}(Q_2) \qquad \textbf{(P1)}$$

This follows from the following derivation: suppose $Q_1 \not\subseteq \mathtt{RED}_{\mathsf{U}}(Q_2)$. Then there exists a state s such that $s \in Q_1$ and $s \notin \mathtt{RED}_{\mathsf{U}}(Q_2)$. By the definition of $\mathtt{RED}_{\mathsf{U}}$, $\gamma(s) \not\subseteq \gamma(Q_2)$; on the other hand, since $\gamma(Q_1) \subseteq \gamma(\mathtt{RED}_{\mathsf{U}}(Q_2)) = \gamma(Q_2)$, $\gamma(s) \subseteq \gamma(Q_2)$, reaching a contradition.

We then have the following:

$$\begin{array}{ll}
 & ||\varphi||_i \preceq_a ||\varphi||_r \\
\Rightarrow & \text{(by the definition of } \preceq_a\text{)} \\
 & \gamma(\mathsf{U}(||\varphi||_i) \subseteq \gamma(\mathsf{U}(||\varphi||_r)) \\
\Rightarrow & \text{(by the definition of } \mathtt{RED}_{\mathsf{U}}, \gamma(Q) = \gamma(\mathtt{RED}_{\mathsf{U}}(Q))\text{)} \\
 & \gamma(\mathsf{U}(||\varphi||_i) \subseteq \gamma(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(||\varphi||_r))) \\
\Rightarrow & \text{(by } \textbf{(P1)}\text{)} \\
 & \mathsf{U}(||\varphi||_i)) \subseteq \mathtt{RED}_{\mathsf{U}}(\mathsf{U}(||\varphi||_r)) \\
\Rightarrow & \text{(by monotonicity of } pre \text{)} \\
 & pre_{\mathsf{U}}(\mathsf{U}(||\varphi||_i)) \subseteq pre_{\mathsf{U}}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(||\varphi||_r))) \\
\Rightarrow & \text{(by monotonicity of } \gamma\text{)} \\
 & \gamma(pre_{\mathsf{U}}(\mathsf{U}(||\varphi||_i))) \subseteq \gamma(pre_{\mathsf{U}}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(||\varphi||_r)))) \\
\Rightarrow & \text{(by the definition of } \mathtt{RED}_{\mathsf{U}}, \gamma(Q) = \gamma(\mathtt{RED}_{\mathsf{U}}(Q))\text{)} \\
 & \gamma(pre_{\mathsf{U}}(\mathsf{U}(||\varphi||_i))) \subseteq \gamma(\mathtt{RED}_{\mathsf{U}}(pre_{\mathsf{U}}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(||\varphi||_r))))) \\
\Rightarrow & \text{(by the definitions of SIS and RIS)} \\
 & \gamma(\mathsf{U}(||\Diamond\varphi||_i)) \subseteq \gamma(\mathsf{U}(||\Diamond\varphi||_r))
\end{array}$$

Proof of (b) is dual of the one above. □

The previous example illustrates another important point: GKMTSs and MixTSs are equivalent with respect to RIS. For example, $\|\varphi\|_r^{\mathcal{M}_5}$ is equivalent to $\|\varphi\|_r^{\mathcal{G}_1}$. The following theorem formalizes this.

Theorem 13. *Let $\mathcal{G}$ be a GKMTS model, and $\mathcal{M} = \text{GTOM}(\mathcal{G})$. Then, $\mathcal{G}$ and $\mathcal{M}$ are equivalent with respect to RIS: $\forall\varphi \in L_\mu \cdot \|\varphi\|_r^{\mathcal{G}} \equiv_a \|\varphi\|_{r.}^{\mathcal{M}}$*

PROOF. We begin by fixing an abstraction relation $\langle C, \rho, S\rangle$. The proof proceeds by structural induction on φ. For the base case, according to the definition of L_M, $\|p\|_r^{\mathcal{G}} \equiv_a \|p\|_r^{\mathcal{M}}$ for any atomic proposition p. In the following, we show the inductive case for $\Diamond\varphi$; the proofs of the other cases are trivial.

We show that $\|\varphi\|_r^{\mathcal{G}} \equiv_a \|\varphi\|_r^{\mathcal{M}} \Rightarrow \|\Diamond\varphi\|_r^{\mathcal{G}} \equiv_a \|\Diamond\varphi\|_r^{\mathcal{M}}$, which is equivalent to proving the following two statements:

$$(a)\ \ \|\varphi\|_r^{\mathcal{G}} \equiv_a \|\varphi\|_r^{\mathcal{M}} \Rightarrow \gamma(\mathsf{U}(\|\Diamond\varphi\|_r^{\mathcal{G}})) = \gamma(\mathsf{U}(\|\Diamond\varphi\|_r^{\mathcal{M}}))$$

$$(b)\ \ \|\varphi\|_r^{\mathcal{G}} \equiv_a \|\varphi\|_r^{\mathcal{M}} \Rightarrow \gamma(\overline{\mathsf{O}(\|\Diamond\varphi\|_r^{\mathcal{G}})}) = \gamma(\overline{\mathsf{O}(\|\Diamond\varphi\|_r^{\mathcal{M}})})$$

The proof of (a) is as follows. First, note that for any concrete state c and a set of abstract states Q,

$$c \in \gamma(\mathtt{RED}_{\mathsf{U}}(Q)) \Leftrightarrow \exists a \in Q \cdot c \in \gamma(a) \qquad \textbf{(P2)}$$

We then have that, for any concrete state c,

$$
\begin{array}{ll}
 & c \in \gamma(\mathsf{U}(\|\Diamond\varphi\|_r^{\mathcal{G}})) \\
\Leftrightarrow & \text{(by the definition of RIS)} \\
 & c \in \gamma(\mathtt{RED}_{\mathsf{U}}(pre_{\mathsf{U}}^{\mathcal{G}}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}}))))) \\
\Leftrightarrow & \text{((}\Rightarrow\text{) let } a \text{ be the abstract state in } \textbf{(P2)}, \\
 & \ (\Leftarrow) \text{ since } \gamma(Q) = \gamma(\mathtt{RED}_{\mathsf{U}}(Q))) \\
 & c \in \gamma(a) \wedge a \in pre_{\mathsf{U}}^{\mathcal{G}}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}}))) \\
\Leftrightarrow & \text{(by the definition of } pre_{\mathsf{U}}\text{)} \\
 & c \in \gamma(a) \wedge \exists Q \subseteq \mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}})) \cdot R_{\mathcal{G}}^{\text{must}}(a, Q) \\
\Leftrightarrow & \text{(by the definition of G}\textsc{to}\text{M)} \\
 & c \in \gamma(a) \wedge \exists b \cdot \gamma(b) \subseteq \gamma(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}}))) \wedge R_{\mathcal{M}}^{\text{must}}(a, b) \\
\Leftrightarrow & \text{(since } \|\varphi\|_r^{\mathcal{G}} \equiv_a \|\varphi\|_r^{\mathcal{M}}, \gamma(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}})) = \gamma(\mathsf{U}(\|\varphi\|_r^{\mathcal{M}}))\text{)} \\
 & c \in \gamma(a) \wedge \exists b \cdot \gamma(b) \subseteq \gamma(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{M}}))) \wedge R_{\mathcal{M}}^{\text{must}}(a, b) \\
\Leftrightarrow & \text{(since } \gamma(Q) = \gamma(\mathtt{RED}_{\mathsf{U}}(Q))\text{, by the definition of } \mathtt{RED}_{\mathsf{U}}\text{)} \\
 & c \in \gamma(a) \wedge \exists b \in \mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{M}})) \cdot R_{\mathcal{M}}^{\text{must}}(a, b) \\
\Leftrightarrow & \text{(by the definition of } pre_{\mathsf{U}}\text{)} \\
 & c \in \gamma(a) \wedge a \in pre_{\mathsf{U}}^{\mathcal{M}}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{M}}))) \\
\Leftrightarrow & \ \text{((}\Rightarrow\text{) since } \gamma(Q) = \gamma(\mathtt{RED}_{\mathsf{U}}(Q)), \\
 & \ (\Leftarrow) \text{ let } a \text{ be the abstract state in } \textbf{(P2)}\text{)} \\
 & c \in \gamma(\mathtt{RED}_{\mathsf{U}}(pre_{\mathsf{U}}^{\mathcal{M}}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}}))))) \\
\Leftrightarrow & \text{(by the definition of RIS)} \\
 & c \in \gamma(\mathsf{U}(\|\Diamond\varphi\|_r^{\mathcal{M}}))
\end{array}
$$

The proof of (b) is similar to the one above, based on the observation that for any concrete state c and a set of abstract states Q, $c \in \gamma(\overline{\mathtt{RED}_{\mathsf{O}}(Q)}) \Leftrightarrow \exists a \in \overline{Q} \cdot c \in \gamma(a)$. $\Box$

Our new semantics RIS is both inductive and precise enough to make GKMTSs and MixTSs equivalent. However, the definition of the RED operator is based on concretization, γ. In practice, reasoning directly about concrete states may be undecidable or inefficient. We address this limitation next.

6.3. Reduced Inductive Semantics for Monotone Models

We study the reduction operator RED of RIS in the context of monotone models. As shown in Section 3, monotone models are as expressive as their regular counterparts. Furthermore, as shown in [19], monotone models are also more precise. That is, given an arbitrary model $\mathcal{M}$, there is a monotone model $\mathcal{M}'$ over the same statespace that is more precise than $\mathcal{M}$ under SIS. The following theorem implies that the same result also holds under RIS.

Theorem 14. *Let $\mathcal{M} = \langle M, L\rangle$ and $\mathcal{M}' = \langle M', L'\rangle$ be two partial models, where $M = \langle S, R^{\text{may}}, R^{\text{must}}\rangle$ and $M' = \langle S, R^{\text{may}\prime}, R^{\text{must}\prime}\rangle$ are two transition systems defined over the same abstract statespace S. Then, if $\mathcal{M}$ is less precise than $\mathcal{M}'$ under SIS, i.e., $\forall \varphi \in L_\mu \cdot \|\varphi\|_i^{\mathcal{M}} \preceq_a \|\varphi\|_i^{\mathcal{M}'}$, then, $\mathcal{M}$ is also less precise than $\mathcal{M}'$ under RIS.*

PROOF. The proof is by structural induction on φ. In particular, the inductive case for $\Diamond\varphi$ follows from the definition of RED and the monotonicity of the *preimage* operator.

□

Furthermore, models built by automated predicate abstraction [18] in practice are monotone by construction. Thus, restricting RED to monotone models is neither a theoretical nor a practical restriction.

Note that in any monotone model and any formula φ, $\|\varphi\|_r$ is a monotone element. This holds because of the monotonicity of the state labeling and the transition relation. For monotone elements, RED can be computed effectively, as we show below.

Let $\langle C, \rho, S\rangle$ be an abstraction relation, and $s \in S$ be a state. The *upset* of s is defined as

$$\uparrow s \triangleq \{t \in \alpha[C] \mid s \preceq_a t\}\,.$$

Thus, $\uparrow s$ is the set of all those states in $\alpha[C]$ that are more precise than s. For example, consider the abstraction relation $\langle \mathbb{Z}, \rho, S_1\rangle$, where S_1 be the statespace of M_5 shown in Figure 2. Recall that $\alpha[\mathbb{Z}]$ denotes the set of abstract states in S_1 that are best approximations of concrete states. Since every value of an integer variable x can be best approximated by a_1, a_2, a_3, or a_4, $\alpha[\mathbb{Z}] = \{a_1, a_2, a_3, a_4\}$. Furthermore, since a_2 and a_3 are more precise than a_5, we have that $\uparrow a_5 = \{a_2, a_3\}$. A state s and the upset $\uparrow s$ approximate the same set of concrete states, i.e., $\gamma(s) = \gamma(\uparrow s)$. For example, $\gamma(a_5) = \gamma(\uparrow a_5) = \gamma(\{a_1, a_2, a_3, a_4\}) = (x > 0)$.

The next theorem shows that for monotone elements of $2^S \times 2^S$ the upset operator lifts set inclusion from concrete to the abstract domain.

Theorem 15. *Let $\langle C, \rho, S\rangle$ be an abstraction relation, $e = \langle U, O\rangle$ be a monotone element of $2^S \times 2^S$, and $s \in S$ be a state. Then, $\gamma(s) \subseteq \gamma(U)$ iff $\uparrow s \subseteq U$ and $\gamma(s) \not\subseteq \gamma(\overline{O})$ iff $\uparrow s \not\subseteq \overline{O}$.*

PROOF. First, we show that $\gamma(s) \subseteq \gamma(U) \Leftrightarrow \uparrow s \subseteq U$. The ($\Leftarrow$) direction follows directly from the definition of γ. We prove the ($\Rightarrow$) direction by contradiction. Let C be the concrete statespace approximated by S. Suppose that $\uparrow s \not\subseteq U$. Then,

$$\begin{array}{ll}
 & \uparrow s \not\subseteq U \\
\Rightarrow & \exists a \in S \cdot a \in \uparrow s \wedge a \notin U \\
\Rightarrow & \text{(by the definition of } \uparrow s\text{, } a \in \alpha[S]\text{)} \\
 & \exists a \in S \cdot s \preceq_a a \wedge a \notin U \wedge \exists c \in C \cdot a = \alpha(c) \\
\Rightarrow & \text{(since } s \preceq_a a\text{, } \gamma(a) \subseteq \gamma(s)\text{; since } \gamma(s) \subseteq \gamma(U)\text{)} \\
 & \exists a \in S \cdot a \notin U \wedge \exists c \in C \cdot a = \alpha(c) \wedge c \in \gamma(U) \\
\Rightarrow & \text{(by the definition of } \gamma\text{)} \\
 & \exists a \in S \cdot a \notin U \wedge \exists c \in C \cdot a = \alpha(c) \wedge \exists b \in U \cdot c \in \gamma(b) \\
\Rightarrow & \text{(by the definition of } \alpha\text{)} \\
 & \exists a \in S \cdot a \notin U \wedge \exists b \in U \cdot b \preceq_a a \\
\Rightarrow & \text{(by monotonicity of } e\text{, } a \in U\text{)} \\
 & \exists a \in S \cdot a \notin U \wedge a \in U \\
\Rightarrow & \mathsf{false}
\end{array}$$

The proof of $\gamma(s) \not\subseteq \gamma(\overline{O}) \Leftrightarrow \uparrow s \not\subseteq \overline{O}$ is dual to the one above. □

We now define a new operator $\texttt{red}$ for monotone elements. Let $e = \langle U, O \rangle$ be a monotone element of $2^S \times 2^S$. $\texttt{red}$ is defined as

$$\texttt{red}(e) \triangleq \langle \texttt{red}_{\mathsf{U}}(U), \texttt{red}_{\mathsf{O}}(O) \rangle$$

where $\texttt{red}_{\mathsf{U}}(U) \triangleq \{s \mid \uparrow s \subseteq U\}$ and $\texttt{red}_{\mathsf{O}}(O) \triangleq \{s \mid \uparrow s \not\subseteq \overline{O})\}$. A corollary of Theorem 15 is that $\texttt{red}$ and $\texttt{RED}$ are equivalent.

Corollary 4. *Let $\langle C, \rho, S \rangle$ be an abstraction relation, and e be a monotone element in $2^S \times 2^S$. Then,* $\texttt{red}(e) = \texttt{RED}(e)$.

For example, the element e_2 defined in ($\star$) is monotone. We have that $\texttt{red}(\mathsf{U}(e_2)) = \{a_1, a_2, a_3\underline{, a_4, a_5}\}$ since $\uparrow a_5 = \{a_2, a_3\} \subseteq \mathsf{U}(e_2)$, and $\texttt{red}(\mathsf{O}(e_2))$ is the same as $\mathsf{O}(e_2)$ since $\overline{\mathsf{O}(e_2)}$ is empty. Therefore, $\texttt{red}(e_2)$ and $\texttt{RED}(e_2)$ are equal. Note that $\texttt{red}$ can be computed effectively since it does not reason about concrete elements directly.

In this section, we have introduced a new inductive semantics RIS, and shown that it is more precise than SIS, and that GKMTSs and MixTSs are equivalent with respect to RIS. RIS can be computed effectively on monotone models, which is not a limitation since monotone models are as expressive as their non-monotone counterparts.

7. Symbolic Model Checking of RIS using BDDs

In this section, we describe a symbolic algorithm RIS that implements the RIS semantics for *monotone* models constructed using predicate abstraction. These are the models used by some existing software model checkers, such as [20].

Our implementation is based on the following observations, which allow us to simply the encoding of computation results and transition systems.

Let $\langle C, \rho, S \rangle$ be an abstraction relation. Then, for any monotone element of $2^S \times 2^S$, there exists a *semantically equivalent* element in $2^{\alpha[S]} \times 2^{\alpha[S]}$. For example, the monotone element e_2 defined in ($\star$) is semantically equivalent to $\langle \{a_1, a_2, a_3, a_4\}, \{a_1, a_2, a_3, a_4\} \rangle$.

Theorem 16. *Let $\langle C, \rho, S \rangle$ be an abstraction relation, $e_1 = \langle U_1, O_1 \rangle$ be a monotone element of $2^S \times 2^S$, and $e_2 = \langle U_2, O_2 \rangle$ be in $2^{\alpha[C]} \times 2^{\alpha[C]}$. If $U_1 \cap \alpha[C] = U_2$ and $O_1 \cap \alpha[C] = O_2$, then $e_1 \equiv_a e_2$.*

PROOF. This is proved by showing that $\texttt{RED}(e_1) = \texttt{RED}(e_2)$; since $\texttt{RED}$ is semantics-preserving, the result holds. □

Recall that the RIS semantics uses the $\texttt{RED}$ operator to compute most precise elements with respect to information ordering without affecting semantic meaning. For two semantically equivalent elements e and e', $\texttt{RED}(e)$ is the same as $\texttt{RED}(e')$; moreover, $\texttt{RED}$ can be effectively computed over monotone models using the elements in $\alpha[C]$. Therefore, Theorem 16 allows us to restrict the algorithm to computing sets over $\alpha[C]$ instead of sets over S. The benefit of this restriction is that we can use fewer variables to encode computation results.

Furthermore, since the result of $\Diamond$ operator is contained in $\alpha[C]$, it only depends on transitions *from* the states of $\alpha[C]$. The following theorem shows that the transition relations can be simplified as well. Specifically, we only use the *may* transitions from $\alpha[C]$ to $\alpha[C]$ and the *must* transitions from $\alpha[C]$ to S. We apply $\texttt{RED}_{\mathsf{U}}$ over the states of $\alpha[C]$ before computing the result of the pre-image over *must* transitions to prevent it from propagating imprecision.

Theorem 17. *Let $\langle C, \rho, S\rangle$ be an abstraction relation, $M = \langle S, R^{\text{may}}, R^{\text{must}}\rangle$ be a monotone MixTS, and $e = \langle U, O\rangle$ be a monotone element of $2^S \times 2^S$. Let $\hat{U} \triangleq U \cap \alpha[S]$, $\hat{O} \triangleq O \cap \alpha[S]$, $\hat{R}^{\text{must}} \triangleq R^{\text{must}} \cap (\alpha[C] \times S)$, and $\hat{R}^{\text{may}} \triangleq R^{\text{may}} \cap (\alpha[S] \times \alpha[S])$. Then,*

$$\langle \mathit{pre}[R^{\text{must}}](\texttt{RED}_{\mathsf{U}}(U)), \mathit{pre}[R^{\text{may}}](\texttt{RED}_{\mathsf{O}}(O))\rangle \equiv_a \langle \mathit{pre}[\hat{R}^{\text{must}}](\texttt{RED}_{\mathsf{U}}(\hat{U})), \mathit{pre}[\hat{R}^{\text{may}}](\hat{O})\rangle$$

PROOF. By the definition of $\equiv_a$, the theorem is equivalent to proving the following results:

(a) $\gamma(\mathit{pre}[R^{\text{must}}](\texttt{RED}_{\mathsf{U}}(U))) = \gamma(\mathit{pre}[\hat{R}^{\text{must}}](\texttt{RED}_{\mathsf{U}}(\hat{U})))$

(b) $\gamma(\overline{\mathit{pre}[R^{\text{may}}](\texttt{RED}_{\mathsf{O}}(O))}) = \gamma(\overline{\mathit{pre}[\hat{R}^{\text{may}}](\hat{O})})$

(1) We first show that (a) holds. The proof of $\gamma(\mathit{pre}[R^{\text{must}}](\texttt{RED}_{\mathsf{U}}(U))) \subseteq \gamma(\mathit{pre}[\hat{R}_1](\texttt{RED}_{\mathsf{U}}(\hat{U})))$ is shown below. For any concrete state c,

$$\begin{array}{ll}
 & c \in \gamma(\mathit{pre}[R^{\text{must}}](\texttt{RED}_{\mathsf{U}}(U))) \\
\Rightarrow & \exists a \in S \cdot c \in \gamma(a) \wedge a \in \mathit{pre}[R^{\text{must}}](\texttt{RED}_{\mathsf{U}}(U)) \\
\Rightarrow & \text{(by the definition of } pre\text{)} \\
 & \exists a \in S \cdot c \in \gamma(a) \wedge \exists b \in \texttt{RED}_{\mathsf{U}}(U) \cdot R^{\text{must}}(a, b) \\
\Rightarrow & \text{(let } a' = \alpha(c)\text{; by the definition of } \alpha\text{)} \\
 & c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \exists a \in S \cdot a \preceq_a a' \wedge \exists b \in \texttt{RED}_{\mathsf{U}}(U) \cdot R^{\text{must}}(a, b) \\
\Rightarrow & \text{(by monotonicity of the transition relations)} \\
 & c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \exists b \in \texttt{RED}_{\mathsf{U}}(U) \cdot R^{\text{must}}(a', b) \\
\Rightarrow & \text{(by the definition of } \hat{R}^{\text{must}}\text{)} \\
 & c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \exists b \in \texttt{RED}_{\mathsf{U}}(U) \cdot \hat{R}^{\text{must}}(a', b) \\
\Rightarrow & \text{(since } e \text{ is a monotone element, } \gamma(U) = \gamma(\hat{U})\text{)} \\
 & c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \exists b \in \texttt{RED}_{\mathsf{U}}(\hat{U}) \cdot \hat{R}^{\text{must}}(a', b) \\
\Rightarrow & \text{(by the definition of } pre\text{)} \\
 & c \in \gamma(a') \wedge a' \in \mathit{pre}[\hat{R}^{\text{must}}](\texttt{RED}_{\mathsf{U}}(\hat{U})) \\
\Rightarrow & c \in \gamma(\mathit{pre}[\hat{R}^{\text{must}}](\texttt{RED}_{\mathsf{U}}(U)))
\end{array}$$

The proof of $\gamma(\mathit{pre}[R^{\text{must}}](\texttt{RED}_{\mathsf{U}}(U))) \supseteq \gamma(\mathit{pre}[\hat{R}^{\text{must}}](\texttt{RED}_{\mathsf{U}}(\hat{U})))$ follows from the definitions of $\hat{R}^{\text{must}}$ and $\hat{U}$.

(2) We now show that $\gamma(\overline{\mathit{pre}[R^{\text{may}}](\texttt{RED}_{\mathsf{O}}(O))}) = \gamma(\overline{\mathit{pre}[\hat{R}^{\text{may}}](\hat{O})})$. The proof of

$\gamma(\overline{\textit{pre}[R^{\text{may}}](\texttt{RED}_{\text{O}}(O))}) \subseteq \gamma(\overline{\textit{pre}[\hat{R}^{\text{may}}](\hat{O})})$ is shown below. For any concrete state c,

$$\begin{array}{ll}
 & c \in \gamma(\overline{\textit{pre}[R^{\text{may}}](\texttt{RED}_{\text{O}}(O))}) \\
\Rightarrow & \exists a \in S \cdot c \in \gamma(a) \wedge a \in \overline{\textit{pre}[R^{\text{may}}](\texttt{RED}_{\text{O}}(O))} \\
\Rightarrow & \text{(by the definition of } pre\text{)} \\
 & \exists a \in S \cdot c \in \gamma(a) \wedge R^{\text{may}}(a) \subseteq \overline{\texttt{RED}_{\text{O}}(O)} \\
\Rightarrow & \text{(let } a' = \alpha(c)\text{; by the definition of } \alpha\text{)} \\
 & c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \\
 & \exists a \in S \cdot a \preceq_a a' \wedge R^{\text{may}}(a) \subseteq \overline{\texttt{RED}_{\text{O}}(O)} \\
\Rightarrow & \text{(by monotonicity of the transition relations)} \\
 & c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \\
 & \exists a \in S \cdot R^{\text{may}}(a') \subseteq R^{\text{may}}(a) \subseteq \overline{\texttt{RED}_{\text{O}}(O)} \\
\Rightarrow & \text{(by the definition of } \hat{R}^{\text{may}}\text{, } R^{\text{may}}(a') \cap \alpha[S] = \hat{R}^{\text{may}}(a')\text{)} \\
 & c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \hat{R}^{\text{may}}(a') \subseteq (\overline{\texttt{RED}_{\text{O}}(O)} \cap \alpha[S]) \\
\Rightarrow & \text{(since } e \text{ is a monotone element, } \forall s \in \alpha[S] \cdot s \in \texttt{RED}_{\text{O}}(O) \Leftrightarrow s \in O \text{)} \\
 & c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \hat{R}^{\text{may}}(a') \subseteq (\overline{O} \cap \alpha[S]) \\
\Rightarrow & \text{(by the definition of } \hat{O}\text{)} \\
 & c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \hat{R}^{\text{may}}(a') \subseteq \alpha[S] \setminus \hat{O} \\
\Rightarrow & \text{(by the definition of } pre\text{)} \\
 & c \in \gamma(a') \wedge a' \in \gamma(\overline{\textit{pre}[\hat{R}^{\text{may}}](\hat{O})}) \\
\Rightarrow & c \in \gamma(\overline{\textit{pre}[\hat{R}^{\text{may}}](\hat{O})})
\end{array}$$

The proof of $\gamma(\overline{\textit{pre}[R^{\text{may}}](\texttt{RED}_{\text{O}}(O))}) \supseteq \gamma(\overline{\textit{pre}[\hat{R}^{\text{may}}](\hat{O})})$ is similar to the one above.
□

The algorithm RIS is shown in Figure 5. It uses BDDs to symbolically represent and manipulate sets of states and transition relations. Functions that are prefixed with "BDD" are the standard BDD operations, shown in Figure 6. The algorithm works recursively on the structure of the input formula φ. The fixpoints are computed as usual, by iterating until convergence. We describe the details of the implementation below.

Let C be a concrete statespace, and $P = \{p_1, \ldots, p_n\}$ be a set of n predicates over C. Recall that the abstraction relation of predicate abstraction is $\langle C, \rho_P, \text{Mon}(P)\rangle$, where Mon$(P)$ denotes the set of monomials over P. Furthermore, MT(P) denotes the set of minterms over P, and $\alpha[C] = \text{MT}(P)$. The input to the algorithm is a MixTS model $\langle M, L_M\rangle$, s.t. $M = (S, R^{\text{may}}, R^{\text{must}})$, $S = \text{Mon}(P)$, and $L_M(s) = Lit(s)$, and an L_μ property φ. Without loss of generality, by Theorem 17, we assume that the transition relations are restricted such that $R^{\text{may}} \subseteq \text{MT}(P) \times \text{MT}(P)$, and $R^{\text{must}} \subseteq \text{MT}(P) \times \text{Mon}(P)$.

The algorithm uses the following sets of BDD variables: $B = \{b_i \mid p_i \in P\}$ – the current state Boolean variables, $B' = \{b'_i \mid b_i \in B\}$ – the next state Boolean variables, $H = \{h_i \mid p_i \in P\}$ – the current state unknown variables, and $H' = \{h'_i \mid h_i \in H\}$ – the next state unknown variables. In what follows, we do not distinguish between the BDDs and the corresponding propositional formulas.

```
global var Rmay, Rmust : BDD
func RIS(Expr $\varphi$) : BDD
    match $\varphi$ with
        ATOMIC(p) : return ABSV(BDDVAR("p"),
                                BDDVAR("p"))
        $\neg\psi$ : return ABSNOT(RIS($\psi$))
        $\psi_1 \wedge \psi_2$ : return ABSAND(RIS($\psi_1$), RIS($\psi_2$))
        $\psi_1 \vee \psi_2$ : return ABSOR(RIS($\psi_1$), RIS($\psi_2$))
        $\Diamond\psi$ : return ABSPRE(Rmay, Rmust, RIS($\psi$))
        $\mu\psi$ : return RIS$_{\text{lfp}}$($\psi$)
        $\nu\psi$ : return RIS$_{\text{gfp}}$($\psi$)

func ABSV(BDD u, BDD o) : BDD
    sel := BDDVAR("sel")
    return BDDITE(sel, u, o)

func ABSO(BDD v) = v[0/sel]
func ABSU(BDD v) = v[1/sel]
func ABSAND(BDD v1, BDD v2) = BDDAND(v1, v2)
func ABSOR(BDD v1, BDD v2) = BDDOR(v1, v2)
func ABSEQ(BDD v1, BDD v2) = BDDEQ(v1, v2)

func ABSNOT(BDD v) : BDD
    o := ABSO(v), u := ABSU(v)
    return ABSV(BDDNOT(o), BDDNOT(u))

func ABSREDU(BDD v) : BDD
    if (BDDISCONST(v)) return v
    b := BDDROOTVAR(v), h := UVAR($b$)
    T := ABSREDU(v[1/b]), F := ABSREDU(v[0/b])
    tmp := BDDITE($b, T, F$)
    return BDDITE(h, BDDAND($T, F$), tmp)

func ABSPRE(BDD Rmay, BDD Rmust, BDD v) : BDD
    o := ABSO(v), u := ABSREDU(ABSU(v))
    return ABSV(BDDPRE(Rmust, u), BDDPRE(Rmay, o))
```

Figure 5: The RIS algorithm and its supporting functions.

Operation	Definition
BDDVAR(s)	returns a BDD node with the name s
BDDITE(f, g, h)	returns a BDD for $(\mathtt{f} \wedge \mathtt{g}) \vee (\neg\mathtt{f} \wedge \mathtt{h})$
BDDAND(f, g)	returns a BDD for conjunction $\mathtt{f} \wedge \mathtt{g}$
BDDOR(f, g)	returns a BDD for disjunction $\mathtt{f} \vee \mathtt{g}$
BDDEQ(f, g)	checks whether BDD f equals BDD g
BDDPRE(r, v)	returns a BDD for the preimage of v over r
BDDISCONST(f)	checks whether BDD f is a constant
BDDROOTVAR(f)	returns the top node of BDD f

Figure 6: Common BDD operations.

A set of minterms $X \subseteq \text{MT}(P)$ is encoded by a propositional formula over B, as usual. For example, let $P = \{p_1, p_2, p_3\}$. Then $b_1 \wedge \neg b_2$ encodes the set $\{p_1 \wedge \neg p_2 \wedge p_3, p_1 \wedge \neg p_2 \wedge \neg p_3\}$. A set of monomials $X \subseteq \text{Mon}(P)$ is encoded by a formula over $B \cup H$. Intuitively, for a monomial m, a variable h_i indicates whether p_i is present in m, and a variable b_i specifies the polarity of the occurrence. Formally, the encoding is

$$\bigvee_{m \in X} \left(\Big(\bigwedge_{p_i \in Lit(m)} \neg h_i \wedge b_i \Big) \wedge \Big(\bigwedge_{\neg p_i \in Lit(m)} \neg h_i \wedge \neg b_i \Big) \wedge \Big(\bigwedge_{p_i \in P \setminus Term(m)} h_i \Big) \right)$$

For example, $(\neg h_1 \wedge b_1) \wedge (\neg h_2 \wedge \neg b_2) \wedge h_3$ represents a singleton set $\{p_1 \wedge \neg p_2\}$.

An abstract value $e = \langle U, O \rangle$ is encoded in a single BDD by a formula $(\mathtt{sel} \wedge U) \vee (\neg\mathtt{sel} \wedge O)$, where sel is a designated BDD variable. This encoding is implemented by function ABSV. The U and O elements of value e are extracted using ABSU and ABSO, respectively. Abstract intersection (ABSAND), union (ABSOR), and equality (ABSEQ) are done using the corresponding BDD operations. Abstract negation (ABSNOT) is implemented following its definition in Section 2.

The may transition relation $R^{\text{may}} \subseteq \text{MT}(P) \times \text{MT}(P)$ is encoded by a formula over

$B \cup B'$ as usual. Similarly, the must relation $R^{\text{must}} \subseteq \text{MT}(P) \times \text{Mon}(P)$ is encoded by a formula over $B \cup B' \cup H'$, where the primed variables are used to encode the destination state. For example, a *must* transition from a state $(p_1 \wedge p_2 \wedge p_3)$ to a state $(p_1 \wedge \neg p_2)$ is represented by $(b_1 \wedge b_2 \wedge b_3) \wedge ((\neg h'_1 \wedge b'_1) \wedge (\neg h'_2 \wedge \neg b'_2) \wedge h'_3)$.

Function ABSREDU implements the $\texttt{red}_{\texttt{U}}$ reduction operator of Section 6.3. It takes a set of minterms as input, and returns a set of monomials for the computation of pre-image over *must* transitions. A monomial is added to the output iff its upset is contained in the input. The implementation of ABSREDU uses the following observation: let $Q \subseteq \text{MT}(P)$ be a set of minterms, and $a \in \text{Mon}(P)$. If $a \in \text{MT}(P)$, then $\uparrow a = \{a\}$, and $\uparrow a \subseteq Q \Leftrightarrow a \in Q$; otherwise, some predicate p is not present in a, and in this case $\uparrow a \subseteq Q$ iff $\uparrow(a \wedge p) \subseteq Q$ and $\uparrow(a \wedge \neg p) \subseteq Q$. For example, suppose $P = \{p_1, p_2, p_3\}$ and $Q = \{p_1 \wedge p_2 \wedge p_3, p_1 \wedge p_2 \wedge \neg p_3\}$. For the monomial $a = p_1 \wedge p_2$, we have that $\uparrow a \subseteq Q$ because $\uparrow(a \wedge p_3) = \uparrow(p_1 \wedge p_2 \wedge p_3) = \{p_1 \wedge p_2 \wedge p_3\} \subseteq Q$ and $\uparrow(a \wedge \neg p_3) = \uparrow(p_1 \wedge p_2 \wedge \neg p_3) = \{p_1 \wedge p_2 \wedge \neg p_3\} \subseteq Q$. Function ABSREDU applies this reasoning recursively on the input diagram, using function UVAR to find a variable $h_i \in H$ for each variable $b_i \in B$. Function ABSPRE implements the pre-image computation based on Theorem 17.

Theorem 18. *For a monotone MixTS $\mathcal{M}$ and $\varphi \in L_\mu$, algorithm* $\text{RIS}(\varphi)$ *in Figure 5 returns the symbolic representation of* $\|\varphi\|_r^{\mathcal{M}}$.

PROOF. The proof is by structural induction on φ. In particular, the base case follows from Theorem 16. The inductive case for boolean operations follows from the fact that $\texttt{RED}$ is semantics-preserving and acts homomorphically with respect to propositional operations on monotone elements. For the inductive case of $\Diamond\varphi$, Corollary 4 shows that $\texttt{RED}_{\texttt{U}}$ can be computed using $\texttt{red}_{\texttt{U}}$ implemented by ABSREDU, and Theorem 17 shows that $\Diamond$ can be computed over the simplified transition relations. □

The main difference between the symbolic implementations of SIS and our RIS is the extra ABSREDU operation in function ABSPRE (line 29 in Figure 5). ABSREDU is similar to existential quantification (BDDEXISTS) of BDDs, with one exception: BDDEXISTS uses BDDOR in each iteration, but ABSREDU uses one BDDAND and two BDDITE operations. Thus, ABSREDU has the same complexity as BDDEXISTS, and symbolic implementations of RIS and SIS also have the same complexity. This means that the extra precision of RIS comes "for free", without a penalty in complexity.

8. Experiments

To empirically evaluate the cost and performance of RIS versus SIS, we have implemented symbolic algorithms for computing both of them using the CUDD library [31], and analyzed reachability and non-termination properties over a realistic model. While our algorithm in Figure 5 can analyze any μ-calculus formula, our experiments considered just reachability and non-termination properties because of their practical interest.

We have conducted the experiments on instances of a template program $\texttt{Prog}_1$ shown in Figure 7(a). For a natural number n, an instance of $\texttt{Prog}_1$ uses n integer variables $\texttt{x}[0], \ldots, \texttt{x}[\texttt{n}-1]$ and consists of n blocks $B(i)$ shown in Figure 7(b), followed by a loop. An instance of $\texttt{Prog}_1$ for $n = 1$ is shown in Figure 7(c).

(a)

```
Prog1 (int n)
 int x[n]

 B(0)
 B(1)
  .
  .
  .
 B(n-2)
 B(n-1)

 L: while (x[n-1]<=0)
     x[n-1]=x[n-1]-1
 END:
```

(b)

```
B(int i)
 if (x[i]>5)
   x[i]=x[i]+1
 else if (x[i]>0)
   x[i]=x[i]+2
 else
   x[i]=x[i]-2

 while  (x[i]>0)
   if (odd(x[i]))
     x[i]=-1
    else
     x[i]=x[i]+1
```

(c)

```
L0: if (x[0]>5)
         x[0]=x[0]+1
      else if (x[0]>0)
         x[0]=x[0]+2
     else
        x[0]=x[0]-2

L1: while  (x[0]>0)
        if (odd(x[0]))
            x[0]=-1
        else
           x[0]=x[0]+1

L:  while  (x[0]<=0)
        x[0]=x[0]-1
END:
```

Figure 7: Template $\texttt{Prog}_1(n)$ for experiments: (a) the template, (b) definition of block $B(i)$, and (c) the instance $\texttt{Prog}_1(1)$.

We used the method of [18] to build an abstract MixTS using the set of predicates $\{\texttt{x}[0] > 0, \texttt{x}[1] > 0, \ldots, \texttt{x}[\texttt{n}-1] > 0\} \cup \{\texttt{odd}(\texttt{x}[0]), \texttt{odd}(\texttt{x}[1]), \ldots, \texttt{odd}(\texttt{x}[\texttt{n}-1])\}$. We model checked the following reachability (least fixed-point) and non-termination (greatest fixed-point) properties with respect to the standard and the reduced semantics:

$$\begin{aligned}
&\texttt{Prop}_1 &&: EF(pc = \texttt{L})\\
&\texttt{Prop}_2 &&: EG(pc \neq \texttt{END})\\
&\texttt{Prop}_3 &&: EG(pc \neq \texttt{END} \wedge (\texttt{x}[0] > 0 \vee \texttt{x}[1] > 0 \vee \cdots \vee \texttt{x}[\texttt{n}-1] > 0)),
\end{aligned}$$

where pc refers to program counter.

The template $\texttt{Prog}_1$ is based on an example from [29] that shows that using GKMTSs can improve the precision of model checking. For example, consider the instance $\texttt{Prog}_1(1)$ shown in Figure 7(c). A part of the corresponding abstract MixTS is shown in Figure 8. Here, the property $\texttt{Prop}_1$ is unknown in a_1 with respect to SIS. As shown in [29], the precision can be improved by adding a *must* hyper-transition $a_1 \xrightarrow{\text{must}} \{a_2, a_4\}$. We use this template to show that the same result can be also achieved using RIS.

For both SIS and RIS, we measure the size of the abstract models using the number of BDD nodes, the total analysis time, the number of iterations of the fixpoint computation, and the time spent in the ABSREDU operation for RIS. To compare the precision of the results, we consider two sets of initial states:

$$\begin{aligned}
\texttt{I}_1 &: (\texttt{x}[0] \leq 0 \wedge \texttt{x}[1] \leq 0 \wedge \cdots \wedge \texttt{x}[\texttt{n}-1] \leq 0)\\
\texttt{I}_2 &: (\texttt{x}[0] > 0 \wedge \texttt{x}[1] > 0 \wedge \cdots \wedge \texttt{x}[\texttt{n}-1] > 0)
\end{aligned}$$

and check whether conclusive results can be obtained over them.

The results are summarized in Figure 9. The top part of the table shows that RIS models enjoy significantly smaller encodings than their SIS counterparts, due to restricted transition relations (see Theorem 17). Note that the same simplification cannot

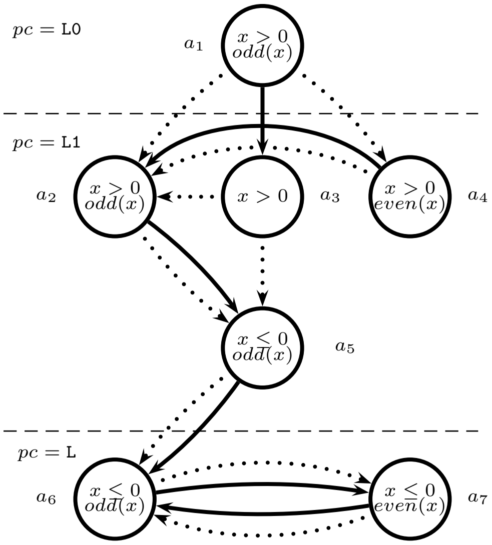

Figure 8: A partial view of a MixTS approximating $\mathtt{Prog}_1(1)$ from Figure 7(c).

	n	SIS				RIS				
Model Size	100	370,070				216,689				
	200	1,460,270				853,389				
	250	2,275,196				1,329,215				
Prop.	n	Analysis (sec.)	Iter.	$\mathtt{I}_1$	$\mathtt{I}_2$	Analysis (sec.)	AbsRedU (sec.)	Iter.	$\mathtt{I}_1$	$\mathtt{I}_2$
$\mathtt{Prop}_1$	100	2.20	301	T	U	3.60	0.74	401	T	T
	200	15.36	601			27.77	6.45	801		
	250	28.92	751			55.19	13.40	1001		
$\mathtt{Prop}_2$	100	3.60	203	T	U	0.03	$< 10^{-4}$	2	T	T
	200	27.16	403			0.12	$< 10^{-4}$	2		
	250	54.62	503			0.19	$< 10^{-4}$	2		
$\mathtt{Prop}_3$	100	33.96	400	F	F	21.24	4.5	400	F	F
	200	395.24	800			258.72	42.44	800		
	250	1108.67	1000			546.88	101.20	1000		

Figure 9: Experimental results for SIS and RIS over $\mathtt{Prog}_1$ (T, F and U denote *True*, *False* and *Unknown*, respectively).

be applied to SIS, since SIS does not use a reduction operator to compensate for the loss of precision over the states other than $\alpha[S]$. RIS is more precise than SIS: for the two sets of initial states, RIS produces conclusive results for both of them with respect to the three properties being checked, whereas SIS cannot decide whether $\mathtt{Prop}_1$ and $\mathtt{Prop}_2$ hold in $\mathtt{I}_2$. As expected, the extra precision of RIS does not cause a complexity penalty: the experiments show that the increases of the analysis time with respect to the size of the models for both RIS and SIS are comparable. In all of the cases, the time spent in AbsRedU, which represents the main difference between the two semantics, comprises roughly 20% - 25% of the total time.

	n	SIS				RIS				
Model Size	100	245,584				145,284				
	200	971,062				570,462				
	250	1,513,796				888,046				
Prop.	n	Analysis (sec.)	Iter.	$\mathtt{I}_1$	$\mathtt{I}_2$	Analysis (sec.)	AbsRedU (sec.)	Iter.	$\mathtt{I}_1$	$\mathtt{I}_2$
$\mathtt{Prop}_4$	100	0.48	603	U	T	0.27	$< 10^{-4}$	403	T	T
	200	2.15	1203			0.97	$< 10^{-4}$	803		
	250	3.46	1503			1.44	0.01	1003		

Figure 10: Experimental results for SIS and RIS over $\mathtt{Prog}_2$.

(a)

```
Prog2 (int n)
 int x[n]

 C(0)
 C(1)
  .
  .
  .
 C(n-2)
 C(n-1)
 END:
```

(b)

```
C(int i)
 if (nondet)
   x[i]=x[i]+1
   if (nondet)
     x[i]=x[i]+1
    else
     x[i]=x[i]*x[i]-10
 else
   x[i]=x[i]*x[i]-10

 if (x[i]>0)
   x[i]=x[i]+1
 else
   x[i]=x[i]-1
```

Figure 11: Template $\mathtt{Prog}_2(n)$ for experiments: (a) the template, and (b) definition of block $C(i)$.

Note that RIS and SIS may require different numbers of iterations of fixpoint computation: in the above experiments, RIS required more iterations than SIS for the reachability property $\mathtt{Prop}_1$, but fewer iterations than SIS for the non-termination property $\mathtt{Prop}_2$. These differences are determined by the structure of the model and by the fixpoint type (least or greatest) being computed.

As another example, we checked a reachability property on instances of the template $\mathtt{Prog}_2$ shown in Figure 11(a).

Each instance is abstracted using the set of predicates $\{\mathtt{x}[0] > 0, \mathtt{x}[1] > 0, \ldots, \mathtt{x}[\mathtt{n}-1] > 0\}$. The property checked was $\mathtt{Prop}_4 : EF(pc = \mathtt{END})$. The result of model checking was evaluated on the same initial sets of states, $\mathtt{I}_1$ and $\mathtt{I}_2$. The results are summarized in Figure 10. In this case, while still more precise, RIS requires fewer iterations than SIS.

These experiments suggest that using the more precise RIS semantics may improve the overall performance of model checking, making it a possible alternative to SIS in practice. We leave further investigation along this direction for future work.

9. Related Work and Discussion

Consistency. In this paper, we investigated partial TSs and models from the perspective of abstract model checking. Partial TSs are also used as specifications of a system's behavior [25, 24]. In this case, semantic consistency is replaced by implementability. A partial transition system M is *implementable* iff there exists a BTS B that refines M through some mixed simulation. Such a BTS is called *an implementation*. There is a subtle, but crucial, difference between implementability and semantic consistency as defined in this paper. We assume that the statespace of an abstract transition system is an abstract domain, and that it is related to the concrete domain by a given soundness relation ρ. In our case, a partial TS M is semantically consistent iff there exists a BTS that refines M via this ρ. On the other hand, the definition of implementability leaves the choice of the mixed simulation relation open. Thus, semantic consistency is stronger than implementability.

For example, the MixTS M_2 in Figure 1 is not semantically consistent. It is, however, implementable. Let B be a BTS $(\mathbb{Z}, R)$, where $\mathbb{Z}$ is the set of integers, and R is defined as follows:

$$\begin{aligned} R \triangleq\ & \{(x,x') \mid (x > 0 \wedge odd(x) \wedge x' = 2)\} \cup \\ & \{(x,x') \mid (x > 0 \wedge even(x) \wedge x' = -3)\} \cup \\ & \{(x,x') \mid (x > 0 \wedge even(x) \wedge x' = -2)\} \cup \\ & \{(x,x') \mid (x < 0 \wedge x' = -3)\} \quad . \end{aligned}$$

Then, B refines M_2 through the following mixed simulation relation:

$$\{(c,a_1) \mid c > 0 \wedge odd(c)\} \cup \{(c,a_2) \mid c > 0 \wedge even(c)\} \cup$$
$$\{(c,a_3) \mid c \leq 0 \wedge odd(c)\} \cup \{(c,a_4) \mid c \leq 0 \wedge even(c)\}$$

Note that in this case, no concrete state in B is approximated by both a_1 and a_2. Therefore, the source of inconsistency discussed in Section 4 does not exist.

In [21], Huth et al. provided the *mix condition* (MC) on MixTSs to ensure implementability. A MixTS $M = \langle S, R^{\text{may}}, R^{\text{must}} \rangle$ satisfies the mix condition iff for all $(a,b) \in R^{\text{must}}$, there exists some $b' \in S$ such that b' refines b, and $(a,b') \in R^{\text{must}} \cap R^{\text{may}}$. For example, the MixTS M_2 in Figure 1 satisfies this condition, whereas M_4 does not. However, M_2 is semantically inconsistent, and M_4 is consistent. Therefore, MC is neither sufficient nor necessary for semantic consistency.

The complexity of deciding implementability of a partial TS is EXPTIME-complete [4, 3, 1]. On the other hand, semantic consistency can be decided in time polynomial in the size of the system; this is immediate from Theorem 5. This result is not surprising since semantic consistency is stronger than implementability.

Huth et al. showed that the KMTS models are logically consistent [22]. To ensure logical consistency of GKMTSs, de Alfaro et al. defined the condition that requires that every destination of a *must* hyper-transition intersects with the destination of a *may* transition from the same state [13]. This can be viewed as an analogue of the condition $R^{\text{must}} \subseteq R^{\text{may}}$ required by KMTSs. In this paper, we showed that such a condition is not necessary for logical consistency. We fixed this problem by defining a relaxed structural condition which captures both logical consistency and semantic consistency of partial models.

Partial model consistency does not have to be based on mixed simulation. For example, a partial model may be built for abstract model checking of temporal logic properties without the next operator, e.g., as described in [24]. Exploring connections between semantic and logical consistency in this case and providing algorithms for deciding them are interesting questions which we leave for future work.

Expressiveness. The work of Godefroid and Jagadeesan [15], and Gurfinkel and Chechik [17] showed that the models in the KMTS family have the same expressive power and are equally precise for SIS. Dams and Namjoshi [12] showed that the three families considered in this paper are subsumed by tree automata. We completed the picture by proving that the three families are equivalent as well. Specifically, we showed that KMTSs, MixTSs and GKMTSs are relatively complete (in the sense of [12]) with one another.

We did not consider Hyper TSs (HTSs) [30] which allow for both *must* and *may* hyper-transitions. As pointed out in [30], *may* hyper-transitions can be eliminated by increasing the abstract statespace, making HTSs exactly as expressive as GKMTSs.

Our results bring forth several interesting research directions. Since the three modeling formalisms are equally expressive, it would be interesting to study how to relate the results of model checking with respect to thorough semantics for one formalism, e.g., for KMTSs [7, 16], to the ones for another formalism. Another direction is formalizing our translations within the abstract interpretation framework using Galois connections [9].

Reduced Inductive Semantics. Our reduction operator RED is an instance of normalization from Abstract Interpretation [9]. There it is often used to provide a canonical representation of equivalent abstract properties. The symbolic implementation ABSREDU is similar to the semantic minimization of 3-valued propositional formulas [28].

Regarding the ability to improve model checking results, the reduction operator is similar to the focus and defocus operations defined in [11]. According to the definition of RED, a formula holds in an abstract state a if (i) $\gamma(a)$ can be split into (i.e., focused) different parts approximated by more precise states than a, and the formula holds in each of these states, or (ii) $\gamma(a)$ can be covered (i.e., defocused) by a set approximated by a state less precise than a, and the formula holds in it. In particular, if the partial model is monotone, then the reduction operator resembles the focus operation only.

For a partial modeling formalism, the ability to support the monotonic abstraction refinement framework allows us to define a best model over an abstract statespace such that model checking on it is more precise than on other models over the statespace. In the context of SIS, as shown in [29], KMTSs is inappropriate for monotonic abstraction refinement — extra *may* transitions required by the condition $R^{\text{must}} \subseteq R^{\text{may}}$ introduce a loss of precision, and therefore, a best KMTS model over an abstract statespace may not exist. However, this is not a problem for MixTSs [10, 19] which support monotonic abstraction refinement by allowing must-only transitions. GKMTSs achieve the same goal by using *must* hyper-transitions [29], which essentially ensure that no extra *may* transitions are added. Theorem 14 shows that our new inductive semantics, RIS, preserves the precision order of partials models with respect to SIS. Therefore, the best abstract model for SIS is also the best one for RIS, and both MixTSs and GKMTSs still support monotonic abstraction refinement under RIS.

In this paper, we use a notion of *thorough semantics* with respect to a *fixed* mixed simulation (i.e., soundness) relation: by Definition 6, a formula φ is true in a model $\mathcal{M}$ with respect to thorough semantics if and only if it is true in all conretizations of $\mathcal{M}$ with respect to a fixed soundness relation ρ. In contrast, in the original definition of Bruns and Godefroid [7], φ is true in $\mathcal{M}$ under thorough semantics if and only if it is true in all concrete structures that mix-simulate $\mathcal{M}$. Thus, our definition is more restrictive (i.e., it considers fewer concrete structures), but is more appropriate in the context of software model-checking where the soundness relation is fixed a priori. We leave further investigations of model-checking complexity and other properties of our definition to future work.

For the original definition of thorough semantics, Godefroid and Huth investigated self-minimizing temporal formulas whose inductive and thorough semantics coincide [14]. Through a semantic minimization process, every L_μ formula can be transformed into an equivalent formula that is self-minimizing, but may be exponentially larger than the original one. Several results along this line, based on the comparison of SIS and thorough semantics, have been reported, e.g., [14, 17, 27, 2]. In this paper, we have used a reduction operator to improve precision of inductive semantics based on the exploration of the approximation ordering over the abstract domain. Our approach is orthogonal to semantic minimization. For example, consider the model $\mathcal{M}_5$ defined in Section 6.1 (its transition system M_5 is shown in Figure 2) and the formula $\psi \triangleq EF(\neg p \wedge q)$, where p and q denote predicates $(x > 0)$ and $odd(x)$, respectively. ψ is self-minimizing. However, its value in a_1 is unknown under SIS, but is true under RIS. We leave further investigation of the relation between RIS and semantic minimization of temporal logic formulas for future work.

We have shown that symbolic model checking of RIS and SIS have the same complexity. An interesting question left for future study is whether there exists an inductive semantics that is more precise than RIS, and whether it can be symbolically model checked with the same complexity as RIS.

10. Conclusion

Several types of partial transition systems (PTSs) have been developed over the years to support abstract model checking of complex temporal formulas. Some were claimed to be more precise; some had a more efficient decision procedure; others were more succinct. In this paper, we have studied these PTSs, partitioned into three families – KMTSs, MixTSs and GKMTSs. We have compared them with respect to two fundamental ways of using PTSs: as objects for abstracting concrete systems, and as models for checking temporal properties.

Specifically, we studied the connection between semantic and logical consistency of TSs, which is necessary to ensure meaningful abstract model checking. We showed that these notions are not equivalent. However, we proved that they coincide for monotone PTSs and provided an effective structural condition which is necessary and sufficient to guarantee consistency.

We have also compared the expressive power of the three families of PTSs w.r.t. their ability to capture abstractions. We showed, by defining semantics-preserving transformations between the formalisms, that while there are structural differences, all three formalisms are equally expressive. Thus, neither hyper-transitions nor restrictions on *may* and *must* transitions affect expressiveness. They do, of course, affect the succinctness of the resulting TSs.

We then turned to looking at the power of these formalisms w.r.t. the cost and precision of model checking. We have introduced a new inductive semantics, RIS, for PTSs and showed not only that it is more precise than the standard semantics, SIS, but also that model-checking under this semantics for MixTSs and GKMTSs has the same results. We have further described a symbolic implementation of model checking with respect to RIS. The outcome is an algorithm that combines the efficient symbolic encoding of MixTSs with the model checking precision of GKMTSs. The symbolic algorithm was evaluated empirically, and our preliminary experiments suggest that RIS should be a good alternative to SIS for predicate abstraction-based model checkers. We leave further experimental comparisons between the two semantics for future work.

We hope that the results of our investigation help eliminate the confusion about the expressive power of the different partial transition systems and enable their increasing usage as underlying formalisms for abstract model checking.

Acknowledgments. We thank Sagar Chaki, Orna Grumberg, and Yael Meller for comments on the paper; and Laurie Lugrin for her help with the experiments.

References

[1] Antonik, A., September 2008. Decision Problems for Partial Specifications: Empirical and Worst-Case Complexity. Ph.D. thesis, Imperial College, London, UK.

[2] Antonik, A., Huth, M., 2006. Efficient Patterns for Model Checking Partial State Spaces in CTL *intersection* LTL. Electronic Notes in Theoretical Computer Science 158, 41–57.

[3] Antonik, A., Huth, M., Larsen, K. G., Nyman, U., Wasowski, A., March 2008. Complexity of Decision Problems for Mixed and Modal Specifications. In: Proceedings of the 11th International Conference of Foundations of Software Science and Computational Structures (FOSSACS'08). Vol. 4962 of LNCS. pp. 112–126.

[4] Antonik, A., Huth, M., Larsen, K. G., Nyman, U., Wasowski, A., 2009. EXPTIME-complete Decision Problems for Modal and Mixed Specifications. Electronic Notes in Theoretical Computer Science 242 (1), 19–33.

[5] Bagnara, R., Hill, P., Zaffanella, E., 2006. Widening Operators for Powerset Domains. International Journal on Software Tools for Technology Transfer (STTT) 8 (4-5), 449–466.

[6] Bruns, G., Godefroid, P., July 1999. Model Checking Partial State Spaces with 3-Valued Temporal Logics. In: Proceedings of the 11th International Conference on Computer-Aided Verification (CAV'99). Vol. 1633 of LNCS. pp. 274–287.

[7] Bruns, G., Godefroid, P., August 2000. Generalized Model Checking: Reasoning about Partial State Spaces. In: Proceedings of the 11th International Conference on Concurrency Theory (CONCUR'00). Vol. 1877 of LNCS. pp. 168–182.

[8] Chechik, M., Devereux, B., Easterbrook, S., Gurfinkel, A., 2003. Multi-Valued Symbolic Model-Checking. ACM Transactions on Software Engineering and Methodology (TOSEM) 12 (4), 1–38.

[9] Cousot, P., Cousot, R., 1992. Abstract Interpretation Frameworks. Jornal of Logic and Computation 2 (4), 511–547.

[10] Dams, D., Gerth, R., Grumberg, O., 1997. Abstract Interpretation of Reactive Systems. ACM Transactions on Programming Languages and Systems (TOPLAS) 2 (19), 253–291.

[11] Dams, D., Namjoshi, K. S., July 2004. The Existence of Finite Abstractions for Branching Time Model Checking. In: Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS'04). pp. 335–344.

[12] Dams, D., Namjoshi, K. S., January 2005. Automata as Abstractions. In: Proceedings of the 6th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'05). Vol. 3385 of LNCS. pp. 216–232.

[13] de Alfaro, L., Godefroid, P., Jagadeesan, R., July 2004. Three-Valued Abstractions of Games: Uncertainty, but with Precision. In: Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS'04). pp. 170–179.

[14] Godefroid, P., Huth, M., June 2005. Model Checking v.s. Generalized Model Checking: Semantic Minimizations for Temporal Logics. In: Proceedings of the 20th IEEE Symposium on Logic in Computer Science (LICS'05). pp. 158–167.

[15] Godefroid, P., Jagadeesan, R., January 2003. On the Expressiveness of 3-Valued Models. In: Proceedings of the 4th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'03). Vol. 2575 of LNCS. pp. 206–222.

[16] Godefroid, P., Piterman, N., January 2009. LTL Generalized Model Checking Revisited. In: Proceedings of the 10th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'09). Vol. 5403 of LNCS. pp. 89–104.

[17] Gurfinkel, A., Chechik, M., October 2005. How Thorough is Thorough Enough. In: Proceedings of 13th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'05). Vol. 3725 of LNCS. pp. 65–80.

[18] Gurfinkel, A., Chechik, M., March 2006. Why Waste a Perfectly Good Abstraction? In: Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06). Vol. 3920 of LNCS. pp. 212–226.

[19] Gurfinkel, A., Wei, O., Chechik, M., January 2006. Systematic Construction of Abstractions for Model-Checking. In: Proceedings of the 7th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'06). Vol. 3855 of LNCS. pp. 381–397.

[20] Gurfinkel, A., Wei, O., Chechik, M., August 2006. YASM: A Software Model-Checker for Verification and Refutation. In: Proceedings of the 18th International Conference on Computer-Aided Verification (CAV'06). Vol. 4144 of LNCS. pp. 170–174.

[21] Huth, M., Jagadeesan, R., Schmidt, D., 2004. A Domain Equation for Refinement of Partial Systems. Mathematical Structures in Computer Science 14 (4), 469–505.

[22] Huth, M., Jagadeesan, R., Schmidt, D. A., April 2001. Modal Transition Systems: A Foundation for Three-Valued Program Analysis. In: Proceedings of 10th European Symposium on Programming (ESOP). Vol. 2028 of LNCS. pp. 155–169.

[23] Kozen, D., 1983. Results on the Propositional μ-calculus. Theoretical Computer Science 27, 334–354.

[24] Larsen, K. G., Nyman, U., Wasowski, A., September 2007. On Modal Refinement and Consistency. In: Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07). Vol. 4703 of LNCS. pp. 105–119.

[25] Larsen, K. G., Thomsen, B., July 1988. A Modal Process Logic. In: Proceedings of the 3rd Annual Symposium on Logic in Computer Science (LICS '88). pp. 203–210.

[26] Larsen, P., July 1991. The Expressive Power of Implicit Specifications. In: Proceedings of the 18th International Colloquium on Automata, Languages and Programming (ICALP'91). Vol. 510 of LNCS. pp. 204–216.

[27] Nejati, S., Gheorghiu, M., Chechik, M., November 2006. Thorough Checking Revisited. In: Proceedings of the 6th International Conference on Formal Methods in Computer-Aided Design (FMCAD'06). pp. 106–116.

[28] Reps, T. W., Loginov, A., Sagiv, S., July 2002. Semantic Minimization of 3-Valued Propositional Formulae. In: Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS'02). pp. 40–54.

[29] Shoham, S., Grumberg, O., March 2004. Monotonic Abstraction-Refinement for CTL. In: Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04). Vol. 2988 of LNCS. pp. 546–560.

[30] Shoham, S., Grumberg, O., August 2006. 3-Valued Abstraction: More Precision at Less Cost. In: Proceedings of the 21th IEEE Symposium on Logic in Computer Science (LICS'06). pp. 399–410.

[31] Somenzi, F., 2001. CUDD: CU Decision Diagram Package Release.

[32] Wei, O., Gurfinkel, A., Chechik, M., January 2009. Mixed Transition Systems Revisited. In: Proceedings of the 10th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'09). Vol. 5403 of LNCS. pp. 349–365.