Mixed Transition Systems Revisited

Ou Wei*, Arie Gurfinkel†, and Marsha Chechik*
*University of Toronto. {owei,chechik}@cs.toronto.edu
†Software Engineering Institute, Carnegie Mellon University. arie@sei.cmu.edu

***Abstract*—A variety of partial modeling formalisms, aimed to capture and reason about abstractions, have been proposed. Some, e.g., Kripke Modal Transition Systems (KMTSs) put strong restrictions on necessary and possible behaviours. Some, e.g., Mixed Transition Systems (MixTSs), relax these restrictions. Yet others, e.g., Generalized Kripke MTSs (GKMTSs), allow hyper-transitions.**

In this paper, we aim to understand trade-offs between these formalisms w.r.t. their applicability to symbolic model-checking. We establish that these formalisms have the same expressive power while differing in succinctness. We also measure the analyzability of these formalisms, measured as the precision of computing compositional semantics of temporal logic formulas. We show that the standard compositional semantics is not preserved between equivalent GKMTSs and MixTSs, and introduce a novel semantics, called *reduced*, which remains compositional while being both more precise than the standard one and preserved by the semantic equivalence.

We also present a symbolic algorithm to compute the reduced semantics for MixTS models built via predicate abstractions and report on our experience using it in practice.

I. 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 abstracting the concrete states, analyzing the resulting abstract system, and lifting the result back to the concrete system. The interpretation of the result depends on the type of property being checked, and the type of the abstraction used. The two common abstraction schemes are *over-approximation* – the abstract system contains *more* behaviours than the concrete one and thus preserves universal properties, and *under-approximation* – the abstract system contains *less* behaviours than the concrete one and thus preserves existential properties. Preserving arbitrary properties (e.g., full μ-calculus, L_μ, [15]) requires combining over- and under-approximation into a single model [5, 16]. This is done via using two types of transitions, *may* and *must*, representing *possible* (or over-approximating), and *necessary* (or under-approximating) behaviours, respectively. We refer to such models as *partial*. Temporal properties over partial models are interpreted using the 3-valued semantics: a property can be either true, false, or *unknown*.

A variety of partial modeling formalisms have been developed, forming three separate families. The first is *Kripke Modal Transition Systems* (KMTSs) [14] and their equivalent variants, *Modal TSs* [16], *Partial Kripke Structures* (PKSs) [2], and *3-valued KSs* [3]. It requires that every *must* transition is also a *may* transition. KMTSs were first introduced as computational models for partial specifications of reactive systems [16], and later adapted for model-checking [2, 3, 14]. The second family is Mixed Transition Systems (MixTSs) [5], or, equivalently, Belnap TSs [12]. It places no restrictions on the relationship between *may* and *must* transitions and thus extends KMTSs. MixTSs where introduced in [5] as abstract models for L_μ, and have been combined with predicate abstraction and software model-checking in [11]. The third family is *Generalized KMTSs* (GKMTSs) [20], or, equivalently, *Abstract TSs* [7] and *Disjunctive MTSs* [17]. It extends MixTSs by allowing *must hyper-transitions*, i.e., transitions into sets of states. Both MixTSs and KMTSs have been used in practical symbolic model-checkers (e.g., [3, 11, 13]), while the direct use of GKMTSs has been hampered by the difficulty of compactly encoding hyper-transitions into BDDs.

In this paper, we compare the three families w.r.t. their suitability as the "right" formalism for symbolic model-checking of partial models. Our basis of comparison is (i) the expressive power of the formalisms (i.e., what can be modeled, what abstraction can be captured) (ii) analyzability of the formalisms (i.e., the cost and precision of evaluating temporal logic).

We show that MixTSs, KMTSs and GKMTSs are equally expressive: for any partial model M expressed in one formalism, there exists a partial model M' in the other s.t. M and M' approximate the same set of concrete systems. Thus, neither hyper-transitions nor restrictions on *may* and *must* transitions affect expressiveness. They do, however, affect the size of the models: a GKMTS can be modeled by a MixTS of smaller or equal size (the reduction can be exponential), and MixTSs are more succinct than KMTSs. Dams and Namjoshi have showed that all of the above partial models are subsumed by tree automata [6]. Our work completes the picture by showing the expressive equivalence *between* those formalisms.

A semantics of temporal logic is called *compositional* if it is defined inductively on the syntax of the logic. We refer to the typical compositional semantics of L_μ on partial models as *standard* (SCS). We show that GKMTSs are more precise than MixTSs (and, hence, KMTSs), w.r.t. SCS. That is, a GKMTS can prove/disprove more properties under SCS than a MixTS obtained by a semantics-preserving translation. This is significant since in practice partial models are evaluated w.r.t. compositional semantics. We propose a novel alternative semantics, called *reduced* (RCS), which remains compositional (and tractable) and is more precise than SCS. We show that GKMTSs and MixTSs are equivalent w.r.t. RCS. Thus, we argue that MixTSs offer a more compact and more versatile alternative to GKMTSs, supporting efficient symbolic

compositional model checking.

To show the practical impact of the above result, we present a symbolic algorithm to compute the reduced semantics of MixTS models constructed using predicate abstraction. We describe our implementation and evaluate it empirically against the standard compositional semantics.

The rest of the paper is organized as follows. Sec. II reviews the necessary background on partial models and abstraction. In Sec. IV, we show that KMTSs, MixTSs and GKMTSs are equally expressive by developing semantics-preserving translations from GKMTSs to MixTSs, and from MixTSs to KMTSs. In Sec. V, we introduce a new *reduced* compositional semantics (RCS) for L_μ. In Sec. VI, we present a symbolic algorithm to compute RCS in the context of predicate abstraction, and report on our experience with this algorithm in Sec. VII. We conclude the paper in Sec. VIII with the summary of the paper, and comparison with related work.

II. Preliminaries

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

A. Complete and Partial Models

A statespace of a *partial* transition system is a tuple $\langle S, \preceq_S \rangle$, where S is a set of states, and $\preceq_S$ is a partial order on S. Intuitively, $s_1 \preceq_S s_2$ means that s_1 is less informative (more partial) than s_2.

Def. 1 (Partial TSs) *[1, 5, 14, 20] A* Generalized Kripke Modal Transition System *(GKMTS) is a tuple* $M = \langle \langle S, \preceq_S \rangle, R^{\text{may}}, R^{\text{must}} \rangle$, *where* $\langle S, \preceq_S \rangle$ *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 s.t.* $R^{\text{must}} \subseteq S \times S$. *A* Kripke Modal TS *(KMTS) is a MixTS s.t.* $R^{\text{must}} \subseteq R^{\text{may}}$. *A* Boolean TS *(BTS) is a KMTS s.t.* $R^{\text{may}} = R^{\text{must}}$.

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. For example, a BTS is *complete* (i.e., not partial) since every *may* behaviour is also a *must* behaviour.

Let AP be a set of atomic propositions, $Lit(AP)$ be a set of literals of AP, and S be a statespace. A *state labeling* is a function $L : S \to 2^{AP}$ that assigns to each state s a set of literals that are true in s. A TS M together with a labeling L, written $\langle M, L \rangle$, is called a *model*. L is defined over literals. Thus, if $p \in L(s)$, we say that p is true in s; if $\neg p \in L(s)$ — p is false in s; otherwise, the value of p is *unknown*. We require that a state labeling is *locally consistent*, i.e., at most one of p and $\neg p$ belongs to $L(s)$; and *monotone* w.r.t. $\preceq_S$, i.e., $s_1 \preceq_S s_2 \Rightarrow L(s_1) \subseteq L(s_2)$.

In this paper, we use the modal μ-calculus [15] (L_μ) as our temporal logic. It is defined as the set of all formulas satisfying the following grammar:

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

where p is an atomic proposition, and Z a fixpoint variable. Furthermore, Z in $\mu Z \cdot \varphi(Z)$ must occur under the scope of an even number of negations. Additional operations are defined as abbreviations: $\varphi \vee \psi \triangleq \neg(\neg\varphi \wedge \neg\psi)$, $\Box\varphi \triangleq \neg\Diamond\neg\varphi$, $\nu Z \cdot \varphi(Z) \triangleq \neg\mu Z \cdot \neg\varphi(\neg Z)$.

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 $\|\varphi\|^{\mathcal{M}}$, is given by a pair $\langle U, O \rangle$, where $U, O \subseteq S$. Intuitively, U is the set of states that satisfy φ, and O is the set of states that are "not known to refute" φ. Thus, φ is true in U, false in $S \setminus O$ and unknown in $O \setminus U$. We call U and O the *under-* and the *over-approximation* of φ, respectively.

The semantics of L_μ is called *compositional* if it is inductive on the syntax of the logic. We refer to the commonly used compositional semantics as *standard* (SCS). In the definition, we use the following notation. Let $e = \langle U, O \rangle$. We write $\mathsf{U}(e)$ and $\mathsf{O}(e)$ to denote U and O, respectively. We use operators $\sqcap$ and $\sim$ defined as follows: $\sim\langle U, O \rangle \triangleq \langle \overline{O}, \overline{U} \rangle$, and $\langle U_1, O_1 \rangle \sqcap \langle U_2, O_2 \rangle \triangleq \langle U_1 \cap U_2, O_1 \cap O_2 \rangle$.

Def. 2 (SCS) *[1, 5, 12, 14, 20]. Let* $\mathcal{M} = \langle M, L_M \rangle$ *be a model,* $M = \langle S, R^{\text{may}}, R^{\text{must}} \rangle$, Var *a set of fixpoint variables, and* $\sigma : Var \to 2^S \times 2^S$. *The* standard compositional semantics (SCS) *of* $\varphi \in L_\mu$ *is:*

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

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

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

$$pre_{\mathsf{O}}(Q) \triangleq \{s \mid \exists t \in Q \cdot s \xrightarrow{\text{may}} t\}$$

B. Partial Models and Abstraction

A *concrete* statespace C is a set of states. An *abstract* statespace approximating C is a set of states S together with a *soundness* relation $\rho : C \times S$, where $(c, s) \in \rho$ means that s *ρ-approximates* c. ρ induces a *concretization* function $\gamma(s) \triangleq \{c \mid (c, s) \in \rho\}$, and an *approximation* ordering $\preceq_a \subseteq S \times S$ defined as $s \preceq_a t \Leftrightarrow \gamma(s) \supseteq \gamma(t)$. That is, $\gamma(s)$ is the set of all concrete states approximated by s, and $s \preceq_a t$ if s is *less precise* (more approximate) than t. For a set $Q \subseteq S$, we define $\gamma(Q) \triangleq \cup_{s \in Q} \gamma(s)$. Following [4], we require that $\preceq$ be an partial order, and that S satisfy "the existence of a best approximation":

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

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

In our examples, we often use the abstract domain of predicate abstraction. Let $P = \{p_1, \ldots, p_n\}$ be a set of n predicates. A conjunction of literals of P is called a *monomial*; a monomial in which each variable p_i appears once (either

positively or negatively) is called a *minterm*. We write $\text{Mon}(P)$ and $\text{MT}(P)$ for the set of all monomials and minterms of P, respectively. The domain of predicate abstraction is the set $\text{Mon}(P)$. The soundness relation ρ_P is defined s.t. $(c, s) \in \rho_P$ iff c satisfies all predicates in s, i.e., $c \models s$; the abstraction $\alpha_P(c) \triangleq (\bigwedge_{c \models p_i} p_i) \wedge (\bigwedge_{c \not\models p_i} \neg p_i)$; $\alpha_P[\text{Mon}(P)] = \text{MT}(P)$; and the approximation ordering is reverse implication.

The approximation relation is extended from a statespace to transition systems using the concept of *mixed simulation*.

Def. 3 (Mixed Sim.) *[5] Let $M_1 = \langle S_1, R_1^{may}, R_1^{must}\rangle$ and $M_2 = \langle S_2, R_2^{may}, R_2^{must}\rangle$ be two MixTSs. $H \subseteq S_1 \times S_2$ is a* mixed simulation *between M_1 and M_2 if for any $(s_1, s_2) \in H$, the following two conditions hold:*

$$\exists t_1 \in S_1 \cdot s_1 \xrightarrow{\text{may}} t_1 \Rightarrow \exists t_2 \in S_2 \cdot s_2 \xrightarrow{\text{may}} t_2 \wedge (t_1, t_2) \in H$$

$$\exists t_2 \in S_2 \cdot s_2 \xrightarrow{\text{must}} t_2 \Rightarrow \exists t_1 \in S_1 \cdot s_1 \xrightarrow{\text{must}} t_1 \wedge (t_1, t_2) \in H$$

*In this case, we say M_2 H-*simulates *M_1, written $M_2 \preceq_H M_1$.*

Intuitively, M_2 simulates M_1 whenever M_2 is less precise about its behaviour than M_1. This definition generalizes to GKMTSs (c.f., [20]).

Let C and S be a concrete and abstract statespaces, respectively, and $\rho \subseteq C \times S$ be the soundness relation. A partial TS M over S *approximates* a BTS B over C (or, equivalently B *refines* M) iff M ρ-simulates B, $M \preceq_\rho B$. Let L_M and L_B be state-labellings for S and C, respectively. L_M *approximates* L_B, denoted $L_M \preceq_\rho L_B$, iff $\rho(c, s) \Rightarrow L_M(s) \subseteq L_B(c)$. A partial model $\mathcal{M} = \langle M, L_M\rangle$ *approximates* a concrete model $\mathcal{B} = \langle B, L_B\rangle$ (or, equivalently, $\mathcal{B}$ *refines* $\mathcal{M}$) iff $M \preceq_\rho B$, and $L_M \preceq_\rho L_B$.

Theorem 1 *[5] Let $\mathcal{B} = \langle B, L_B\rangle$ be a concrete model that refines a partial model $\mathcal{M} = \langle M, L_M\rangle$, and $\varphi \in L_\mu$. Then, $\gamma(\mathsf{U}(\|\varphi\|_c^{\mathcal{M}})) \subseteq \mathsf{U}(\|\varphi\|_c^{\mathcal{B}})$, and $\mathsf{O}(\|\varphi\|_c^{\mathcal{B}}) \subseteq \gamma(\mathsf{O}(\|\varphi\|_c^{\mathcal{M}}))$.*

That is, if φ is true (false) at a state a of $\mathcal{M}$, then it is true (false) at all states $\gamma(a)$ of $\mathcal{B}$.

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 based on the semantic meaning of a partial model was introduced in [2] as *thorough semantics*. It is defined as follows: $\|\varphi\|_t^{\mathcal{M}} = \langle U, O\rangle$ iff $a \in U \Leftrightarrow \forall \mathcal{B} \in \mathbb{C}[\mathcal{M}] \cdot \gamma(a) \subseteq \mathsf{U}(\|\varphi\|_c^{\mathcal{B}})$, and $a \notin O \Leftrightarrow \forall \mathcal{B} \in \mathbb{C}[\mathcal{M}] \cdot \gamma(a) \subseteq \mathsf{U}(\|\neg\varphi\|_c^{\mathcal{B}})$.

To compare different interpretations of L_μ, we introduce two ordering relations on $2^S \times 2^S$. Let $e_1 = \langle U_1, O_1\rangle$ and $e_2 = \langle U_2, O_2\rangle$. We say that e_1 is *less informative* than e_2, written $e_1 \preceq_i e_2$ iff $U_1 \subseteq U_2$ and $O_2 \subseteq O_1$. We say that e_1 is *semantically less precise* than e_2, written $e_1 \preceq_a e_2$, iff $\gamma(U_1) \subseteq \gamma(U_2)$ and $\gamma(\overline{O_1}) \subseteq \gamma(\overline{O_2})$.

III. Consistency

A. Two definitions of consistency

A consistency of a partial model can be defined in two ways: either based on satisfaction of temporal logic formulas (logical consistency), or based on possible concrete refinements (semantic consistency). Here, we formally define the two notions.

A model $\langle M, L\rangle$ is *logically consistent* if for every $\varphi \in L_\mu$, $\mathsf{U}(\|\varphi\|_c) \subseteq \mathsf{O}(\|\varphi\|_c)$. That is, the value of φ in a state $s \in S$, denoted $\|\varphi\|_c(s)$, can be defined using 3-valued logic:

$$\|\varphi\|_c(s) \quad = \quad \begin{cases} \mathsf{t} & \text{if } s \in \mathsf{U}(\|\varphi\|_c) \\ \mathsf{f} & \text{if } s \notin \mathsf{O}(\|\varphi\|_c) \\ \mathsf{m} & \text{otherwise} \end{cases}$$

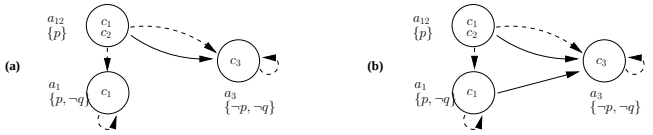

Fig. 1. (a) A semantically inconsistent KMTS $\mathcal{A}_1$ where $\gamma(a_1) = \{c_1\}$, $\gamma(a_{12}) = \{c_1, c_2\}$ and $\gamma(a_3) = \{c_3\}$. (b) A monotone KMTS $\mathcal{A}_2$ equivalent to $\mathcal{A}_1$.

Def. 4 *A transition system M is* logically consistent *if $\langle M, L\rangle$ is logically consistent for every (consistent) labeling function L.*

Let $\mathbb{C}[M]$ denote the set of all concrete refinements of M. A model M is *semantically consistent* if $\mathbb{C}[M]$ is not empty.

Def. 5 *A transition system M is* semantically consistent *if $\langle M, L\rangle$ is semantically consistent for every (consistent) labeling function L.*

There is an obvious relationship between the two notions of consistency.

Theorem 2 *If an abstract transition system M is semantically consistent, then it is logically consistent.*

The converse of Theorem 2 is not true in general. Filling this gap is the subject of the rest of this section.

B. Checking for consistency

It is of practical interest of ensure semantic consistency for partial models, e.g., to ensure a specification given by a partial model is implementable in model-based software development. Surprisingly, we found that such conditions have not been precisely defined before. For example, one may think that the condition $R^{\text{must}} \subseteq R^{\text{may}}$ is sufficient to ensure that KMTSs are semantically consistent. However, although this does guarantee that every KMTS is logically consistent, it does not ensure semantic consistency. For example, for the KMTS $\mathcal{A}_1$ shown in Figure 1(a), every *must* transition is also a *may* transition, but there does not exist a concrete refinement of $\mathcal{A}$ over the states $\{c_1, c_2, c_3\}$. To see this, suppose such a concrete model $\mathcal{C}$ exists. Because of the transition $a_{12} \xrightarrow{must} a_3$, there must be a transition $c_1 \rightarrow c_3$ in $\mathcal{C}$. This in turn requires a *may* transition from a_1 to a_3 which does not exists in $\mathcal{A}_1$. Similarly, for a GKMTS $\mathcal{A} = \langle A, R^{\text{must}}, R^{\text{may}}\rangle$, the sufficient condition for logical consistency [7], which requires that every destination of a *must* hyper-transition intersects with the destination of a *may* transition from the same state, i.e, $\forall a \in A \cdot \forall U \subseteq A \cdot U \in R^{\text{must}}(a) \Rightarrow U \cap R^{\text{may}}(a) \neq \emptyset$, can be viewed as an analogue of the logical consistency condition for KMTSs, and in general does not ensure semantic consistency either.

The reason that logical and semantic consistency are not equivalent is that they describe consistency for different states-

Fig. 2. (a) A consistent MixTS that **does not** satisfy $R^{\text{must}} \subseteq R^{\text{may}}$. (b) A BTS in that refines a MixTS in (a).

paces — the former ensures consistency of temporal properties for abstract states, and the latter — for concrete states. In general, even if $\mathsf{U}(\|\varphi\|_c) \subseteq \mathsf{O}(\|\varphi\|_c)$ holds for each formula φ, it may still be possible that there exist concrete states approximated by both of the states in $\mathsf{U}(\|\varphi\|_c)$ and that in $\overline{\mathsf{O}(\|\varphi\|_c)}$, which results in inconsistency on those concrete states. In the previous example, for the formula $\psi \triangleq \Diamond\, \neg p$, we have that $\mathsf{U}(\|\psi\|_c) = \{a_{12}\} \subseteq \overline{\mathsf{O}(\|\psi\|_c)} = \{a_{12}, a_3\}$. However, because of $\gamma(\mathsf{U}(\|\psi\|_c)) \cap \gamma(\overline{\mathsf{O}(\|\psi\|_c)}) = \{c_1\}$, it implies that c_1 satisfies both ψ and $\neg\psi$! To fix the problem, we add the *monotonicity* requirement on transition relations (see Def. 9) that guarantees the equivalence between logical and semantic consistency.

Theorem 3 *Let $M = \langle S_M, R_M^{\text{must}}, R_M^{\text{may}}\rangle$ be a monotone (Def. 9) MixTS transition system, then the following are equivalent:*

1) *M is semantically consistent, i.e., $\mathbb{C}[M]$ is not empty.*
2) *M is logically consistent, i.e., $\langle M, L\rangle$ is logically consistent for every (consistent) labeling function L.*
3) *for any $a, b_1 \in S_M$, $a \xrightarrow{\text{must}} b_1 \Rightarrow \exists b_2 \in S_M \cdot b_1 \preceq b_2 \wedge a \xrightarrow{\text{may}} b_2$.*

Intuitively, by adding the monotonicity requirement to a MixTS $\mathcal{A}$, if an abstract state a_1 is less precise than a_2, then the truth value of any formula φ at a_1 is less precise than that at a_2, i.e., $\mathcal{A}, a_1 \models \varphi \Rightarrow \mathcal{A}, a_2 \models \varphi$. Therefore, for each concrete state c, if the truth value of φ is consistent at $\alpha(c)$ – the most precise abstract state approximating c, then it is also consistent for c.

The next corollary follows immediately from the fact that KMTSs require that every *must* transition is also a *may* transition.

Cor. 1 *Every monotone KMTS is logically/semantically consistent.*

Every MixTS can be translated to an equivalent monotone one without affecting the concrete models it approximates [12]. Thus, Theorem 3 can also be used to check semantic consistency of non-monotone MixTSs. For example, the MixTS $\mathcal{A}_1$ in the previous example is equivalent to a monotone MixTS $\mathcal{A}_2$ shown in Figure 1(b). It is then easy to check that $\mathcal{A}_2$ is not consistent because of the *must* transition $a_1 \xrightarrow{\text{must}} a_3$ is not matched by any *may* transition.

The only difference between KMTSs and MixTSs is that KMTSs require that $R^{\text{must}} \subseteq R^{\text{may}}$. We believe that originally this requirement was introduced to ensure implementability of specifications represented by KMTSs [16]. However, as we have shown, even with this condition a KMTS may still be semantically inconsistent. In the abstraction refinement framework of software model-checking [11], requiring $R^{\text{must}} \subseteq R^{\text{may}}$ makes KMTSs incapable of supporting precise monotone refinement [20] — extra *may* transitions imposed by this requirement lead to imprecise model checking results. This is not a problem form MixTSs. Note that GKMTS, proposed by Shoham and Grumberg [20] for solving this problem, achieve the same goal by using hyper-transitions. This also ensure that no extra *may* transitions are added to imprecise states.

IV. Expressiveness

We show that GKMTSs, MixTSs, and KMTSs are expressively equivalent. Two partial TSs M and M' are *semantically equivalent*, denoted $M \equiv_a M'$, iff they have the same set of concrete refinements. Two modeling formalisms are *expressively equivalent* iff for every TS M from the one formalism, there exists a TS M' from the other, s.t. $M \equiv_a M'$. The equivalence of the three formalisms is proved by defining semantics-preserving translations between them.

A. GTOM: Translation from GKMTSs to MixTSs

Here, we present the translation GTOM that converts a GKMTS into a semantically equivalent MixTS. We begin by illustrating the translation on a GKMTS G_1 in Fig. 3. G_1 is not a MixTS because of a *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 a (regular) *must* transition $a_1 \xrightarrow{\text{must}} a_5$. The result is a MixTS M_1 in Fig. 3. Moreover, since a_5 replaces a "hyper-state" $\{a_2, a_3\}$, a_5 needs to preserve its *may* behaviours. We do so 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 and M_1 are semantically equivalent: any BTS that refines G_1 also refines M_1, 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_2 in Fig. 3 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_2 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 s.t. $\gamma(b) = \gamma(U)$; (ii) *may* transitions are added for every state introduced in the first step, if any. We formalize this translation below.

Def. 6 (GTOM) *Let $G = \langle S_G, R_G^{\text{may}}, R_G^{\text{must}}\rangle$ be a GKMTS. The translation* $\text{GTOM}(G)$ *is a MixTS*

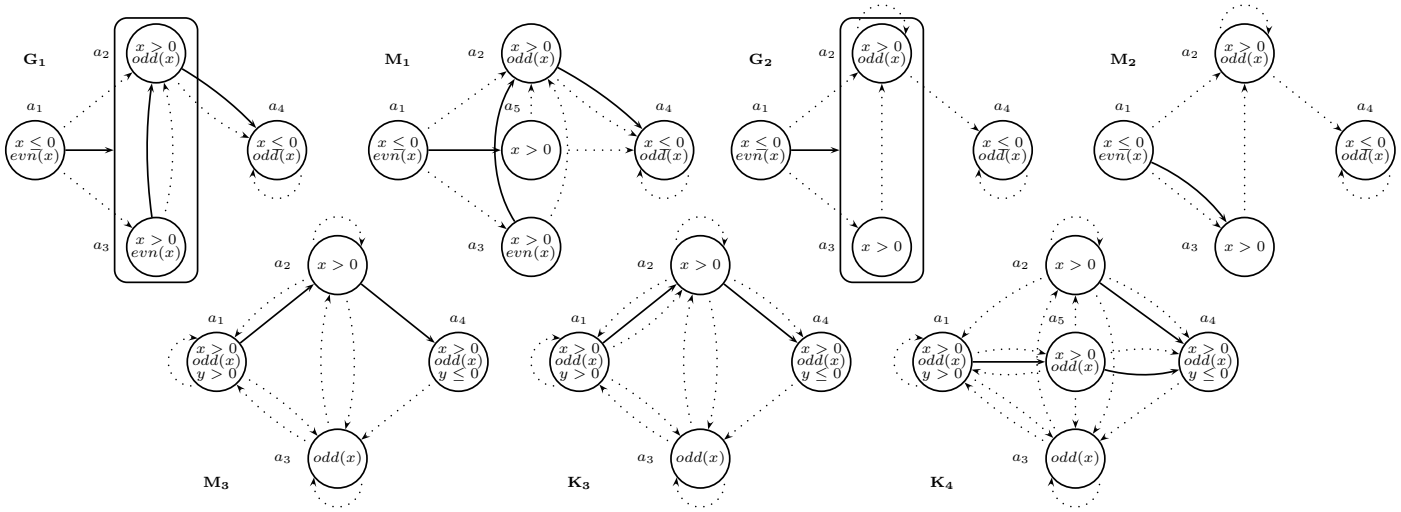

Fig. 3. Two GKMTSs: G_1, G_2; three MixTSs: M_1, M_2, M_3; two KMTSs: K_3, K_4. Solid and dashed lines represent must and may transitions, respectively.

$M = \langle S_M, R_M^{\text{must}}, R_M^{\text{may}} \rangle$, *such that*

$$S_M \triangleq S_G \cup S^+$$

$$S^+ \triangleq \{a \mid \exists (s,U) \in R_G^{\text{must}} \cdot \gamma(a) = \gamma(U) \wedge (\forall t \in S_G \cdot \gamma(t) \neq \gamma(U))\}$$

$$R_M^{\text{may}} \triangleq R_G^{\text{may}} \cup \{(a,b) \mid a \in S^+ \wedge b \in S_G \wedge \exists s \in S_G \cdot (s,b) \in R_G^{\text{may}} \wedge \gamma(s) \subseteq \gamma(a)\}$$

$$R_M^{\text{must}} \triangleq \{(a,b) \mid a \in S_G \wedge b \in S_M \wedge \exists U \subseteq S_G \cdot (a,U) \in R_G^{\text{must}} \wedge \gamma(b) = \gamma(U)\}$$

The translation GTOM is semantics-preserving.

Theorem 4 *Let G be a GKMTS, and $M = \text{GTOM}(G)$. Then, M is a MixTS, and G and M are semantically equivalent.*

A corollary of Thm. 4 is that GKMTSs and MixTSs are equivalent w.r.t. 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 $\text{GTOM}(\langle G, L_G \rangle) \triangleq \langle M, L_M \rangle$, where $M = \text{GTOM}(G)$, and L_M is a labeling function for S_M defined as follows:

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

Then L_M is well-defined and approximates the same labellings as L_G. This ensures that $\langle G, L_G \rangle$ and $\langle M, L_M \rangle$ satisfy the same properties under thorough semantics.

Cor. 2 *Let $\langle G, L_G \rangle$ be a GKMTS model and $\langle M, L_M \rangle = \text{GTOM}(\langle G, L_G \rangle)$. 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 GKTMS 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 [4], 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.

B. MTOK*: Translation from MixTSs to KMTSs*

Below we present the translation MTOK that converts a MixTS into a semantically equivalent KMTS. We begin by illustrating the translation using a MixTS M_3 in Fig. 3. M_3 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_3 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_3 in Fig. 3. However, this transformation is not semantics-preserving, i.e., $K_3 \not\equiv_a M_3$. For example, the concrete system[1]

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

refines K_3, but not M_3: the transition $\langle x = 1, y = 1 \rangle \rightarrow \langle x = 2, y = 1 \rangle$ cannot be simulated by any *may* transition of M_3.

The *must only* transition $a_1 \xrightarrow{\text{must}} a_2$ of M_3 ensures that in any concrete BTS refining M_3, all states in $\gamma(a_1)$, i.e., those satisfying $(x > 0 \wedge \mathit{odd}(x) \wedge y > 0)$, 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_3, every state in $\gamma(a_1)$ must (and may) have a transition to a state in $\gamma(a_2) \cap \gamma(\{a_1, a_3\})$.

Intuitively, the *must only* transition $a_2 \xrightarrow{\text{must}} a_4$ in M_3 is equivalent to a pair of *may* and *must* transitions from a_2 to a_4, since $\gamma(a_4) \cap \gamma(\{a_1, a_2, a_3\}) = \gamma(a_4)$. On the other hand, 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 $\gamma(a_5) = \gamma(a_2) \cap \gamma(\{a_1, a_3\}) = (x > 0 \wedge \mathit{odd}(x))$, 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

[1]Unprimed and primed variables represent current- and next-state valuations, respectively.

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_4 in Fig. 3, which is semantically equivalent to M_3.

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 $a \xrightarrow{\text{must}} \widehat{a \to b}$ 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 translation below.

Def. 7 (MTOK) *Let* $M = \langle S_M, R_M^{\text{may}}, R_M^{\text{must}} \rangle$ *be a MixTS. The translation* $\text{MTOK}(M)$ *is a KMTS* $K = \langle S_K, R_K^{\text{may}}, R_K^{\text{must}} \rangle$*, s.t.*

$$S_K \triangleq S_M \cup S^+$$

$$S^+ \triangleq \{\widehat{a \to b} \mid \exists (a,b) \in (R_M^{\text{must}} \setminus R_M^{\text{may}}) \cdot \forall s \in S_M \cdot \gamma(s) \neq \gamma(\widehat{a \to b})\}$$

$$R_K^{\text{may}} \triangleq R_M^{\text{may}} \cup \text{REPL} \cup \text{IMAY} \cup \text{IMO}$$

$$R_K^{\text{must}} \triangleq (R_M^{\text{must}} \cap R_M^{\text{may}}) \cup \text{REPL} \cup \text{IMUST} \cup \text{IMO}$$

where

$$\text{REPL} \triangleq \{(a, \widehat{a \to b}) \mid \exists (a,b) \in (R_M^{\text{must}} \setminus R_M^{\text{may}})\}$$

$$\text{IMAY} \triangleq \{(\widehat{a \to b}, b') \mid \exists a,b,b' \in S_M \cdot (a,b) \in (R_M^{\text{must}} \setminus R_M^{\text{may}}) \wedge (b,b') \in R_M^{\text{may}} \wedge \widehat{a \to b} \in S^+\}$$

$$\text{IMUST} \triangleq \{(\widehat{a \to b}, b') \mid \exists a,b,b' \in S_M \cdot (a,b) \in (R_M^{\text{must}} \setminus R_M^{\text{may}}) \wedge (b,b') \in (R_M^{\text{must}} \cap R_M^{\text{may}}) \wedge \widehat{a \to b} \in S^+\}$$

$$\text{IMO} \triangleq \{(\widehat{a \to b}, \widehat{b \to b'} \mid \exists a,b,b' \in S_M \cdot (a,b),(b,b') \in (R_M^{\text{must}} \setminus R_M^{\text{may}}) \wedge \widehat{a \to b} \in S^+\}$$

In Def. 7, 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. For our example of $\text{MTOK}(M_3)$, we have

$$S^+ = \{a_5\} \quad \text{REPL} = \{(a_1, a_5), (a_2, a_4)\}$$
$$\text{IMUST} = \emptyset \qquad \text{IMO} = \{(a_5, a_4)\}$$
$$\text{IMAY} = \{(a_5, a_1), (a_5, a_2), (a_5, a_3)\}$$

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

Theorem 5 *Let* M *be a MixTS, and* $K = \text{MTOK}(M)$*. Then* K *is a KMTS, and* M *and* K *are semantically equivalent.*

A corollary of Thm. 5 is that MixTSs and KMTSs are equivalent w.r.t. 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:

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

Then, L_K is well-defined and approximates the same labellings as L_M. This is sufficient to ensure that $\langle M, L_M \rangle$ and $\langle K, L_K \rangle$ satisfy the same properties under thorough semantics.

Cor. 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 $R_M^{\text{must}} \setminus R_M^{\text{may}}$. Therefore, 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: for a fixed statespace S, the set of MixTSs over S is strictly more expressive than the set of KMTSs over S. This is true since for every state t added by MTOK, there exists a subset $U \subseteq S$ s.t. $\gamma(t) = \gamma(U)$.

V. REDUCED COMPOSITIONAL SEMANTICS

GKMTSs and MixTSs are equally expressive: a GKMTS model and its equivalent MixTS model satisfy the same properties under thorough semantics. However, thorough check has exponential complexity. In practice, partial models are evaluated using a more tractable compositional semantics SCS. Unfortunately, GKMTSs are more precise than MixTSs w.r.t. SCS: for any $\varphi \in L_\mu$, the value of φ in a GKMTS model $\mathcal{G}$ under SCS is more precise than its value in the MixTS model $\mathcal{M} = \text{GTOM}(\mathcal{G})$. We propose an alternative semantics, called *reduced compositional semantics* (RCS). While RCS is defined (and evaluated) inductively on the structure of the formula, it is strictly more precise than SCS. We show that GKMTSs and MixTSs are equivalent w.r.t. RCS.

In Sec. V-A, we illustrate the differences between GKMTSs and MixTSs w.r.t. SCS. We define RCS in Sec. V-B, and show how to compute it effectively in Sec. V-C.

A. Example

Let p and q denote predicates $x > 0$ and $odd(x)$, respectively. Consider the model $\mathcal{G}_1 = \langle G_1, L_{G_1} \rangle$, where G_1 is shown in Fig. 3, and L_{G_1} is a labeling function that labels each abstract state as shown in Fig. 3. Let $\mathcal{M}_1 = \langle M_1, L_{M_1} \rangle$ be the model obtained from $\mathcal{G}_1$ by GTOM, where M_1 is shown in Fig. 3 and $L_{M_1}(s) \triangleq$ **if** $s = a_5$ **then** $\{p\}$ **else** $L_{G_1}(s)$.

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

$$\|\varphi\|_c^{\mathcal{G}_1} = \langle \{a_1, a_2, a_3\}, \{a_1, a_2, a_3, a_4\} \rangle$$
$$\|\varphi\|_c^{\mathcal{M}_1} = \langle \{a_2, a_3\}, \{a_1, a_2, a_3, a_4, a_5\} \rangle$$

According to $\mathcal{G}_1$, φ is true in all states corresponding to a_1. According to $\mathcal{M}_1$, the value of φ is unknown in exactly the same states. Since $\mathcal{M}_1 = \text{GTOM}(\mathcal{G}_1)$, $\mathcal{G}_1 \equiv_a \mathcal{M}_1$. Thus, $\mathcal{M}_1$ is less precise than $\mathcal{G}_1$ under SCS.

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

$$e_1 = \|q \vee \neg q\|_c^{\mathcal{G}_1} = \langle \{a_1, a_2, a_3, a_4\}, \{a_1, a_2, a_3, a_4\} \rangle \quad (\star)$$
$$e_2 = \|q \vee \neg q\|_c^{\mathcal{M}_1} = \langle \{a_1, a_2, a_3, a_4\}, \{a_1, a_2, a_3, a_4, a_5\} \rangle$$

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 them. $q \vee \neg q$ is unknown in state a_5 of M_1, even though φ

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.

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

B. Reduced Compositional Semantics for Partial Models

In this section, we define the reduced compositional semantics (RCS). The new semantics is compositional and is *strictly more precise* than SCS. The key idea is to use a *reduction* operator to eliminate any local imprecision.

Let S be an abstract statespace, and $e, e' \in 2^S \times 2^S$ be two abstract elements. 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: $\texttt{RED}(e) \triangleq \langle \texttt{RED}_\mathsf{U}(U), \texttt{RED}_\mathsf{O}(O)\rangle$, where $\texttt{RED}_\mathsf{U}(U) \triangleq \{s \mid \gamma(s) \subseteq \gamma(U)\}$, and $\texttt{RED}_\mathsf{O}(O) \triangleq \{s \mid \gamma(s) \not\subseteq \gamma(\overline{O})\}$. Intuitively, $\texttt{RED}(e)$ increases $\mathsf{U}(e)$ and decreases $\mathsf{O}(e)$ as much as possible without affecting the semantic meaning of e. That is, $\texttt{RED}(e)$ is the largest element w.r.t. information ordering and semantically equivalent to e. For example, consider $\texttt{RED}(e_2)$, where e_2 is as defined by ($\star$). Then,

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

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_2 is less informative since $\mathsf{U}(e_2) \subset \mathsf{U}(e_3)$.

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

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

$\texttt{RED}(e)$ is monotone for any e, and commutes with propositional operations on monotone elements. That is, let e and e' be monotone elements of $2^S \times 2^S$. Then, $\sim e \equiv_a \sim\texttt{RED}(e)$, and $e \sqcap e' \equiv_a \texttt{RED}(e) \sqcap \texttt{RED}(e')$.

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

Def. 8 (RCS) *Let $\mathcal{M} = \langle M, L_M\rangle$ be a model, s.t. $M = \langle S, R^{\text{may}}, R^{\text{must}}\rangle$ and $\sigma : Var \to 2^S \times 2^S$. The* reduced compositional semantics *of $\varphi \in L_\mu$ is defined as follows:*

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

The only difference between RCS (Def. 8) and SCS (Def. 2) is the semantics of $\Diamond$. Since we assume that state-labellings are monotone, applying RED to other operators as well does not improve precision.

Returning to our running example, RCS of φ on $\mathcal{M}_1$ is computed as follows: RCS of q, $\neg q$, and $q \vee \neg q$ is the same as SCS. Thus, $\|q \vee \neg q\|^{\mathcal{M}_1}_r = e_2$. To compute $\Diamond$, recall from ($\star\star$) that $\texttt{RED}(e_2) = e_3$; thus,

$$\|\varphi\|^{\mathcal{M}_1}_r = \langle \{a_1, a_2, a_3, a_5\}, \{a_1, a_2, a_3, a_4, a_5\}\rangle$$

Hence, $\|\varphi\|^{\mathcal{M}_1}_r$ is more precise than $\|\varphi\|^{\mathcal{M}_1}_c$.

Theorem 6 *RCS is more precise than SCS: $\|\varphi\|_c \preceq_a \|\varphi\|_r$.*

The previous example illustrates another important point: GKMTSs and MixTSs are equivalent w.r.t. RCS. For example, $\|\varphi\|^{\mathcal{M}_1}_r$ is equivalent to $\|\varphi\|^{\mathcal{G}_1}_r$. The following theorem formalizes this relationship.

Theorem 7 *Let $\mathcal{G}$ be a GKMTS, and $\mathcal{M} = \text{GTOM}(\mathcal{G})$. Then, $\mathcal{G}$ and $\mathcal{M}$ are equivalent w.r.t. RCS: $\forall\varphi \in L_\mu \cdot \|\varphi\|^{\mathcal{G}}_r \equiv_a \|\varphi\|^{\mathcal{M}}_r$.*

Our new semantics RCS is both compositional and precise enough to make GKMTSs and MixTSs equivalent. However, the RED operator requires comparing concretizations of abstract elements. In practice, this may be impossible or inefficient. We address this limitation next.

C. Reduced Compositional Semantics for Monotone Models

We specialize the reduction operator RED of RCS to monotone models.

Def. 9 *A MixTS $M = \langle S, R^{\text{may}}, R^{\text{must}}\rangle$ is* monotone *iff*

$$\forall s_1 \preceq_S s_2, t_2 \preceq_S t_1 \cdot ((s_2, t_2) \in R^{\text{may}} \Rightarrow (s_1, t_1) \in R^{\text{may}}) \wedge \big((s_1, t_1) \in R^{\text{must}} \Rightarrow (s_2, t_2) \in R^{\text{must}}\big)$$

A model $\mathcal{M} = \langle M, L_M\rangle$ is monotone *iff M is monotone.*

Monotone models are as expressive as their regular counterparts [12]. The monotonicity condition simply ensures that all information that can be derived from the existing *may* and *must* transitions is made explicit in the model.

For a state $s \in S$, let the *upset* of s be defined as $\uparrow s \triangleq \{t \in \alpha[S] \mid s \preceq_a t\}$. Then, $\uparrow s$ is the set of all those states in $\alpha[S]$ that are more precise than s. For example, let S_1 be the statespace of M_1 in Fig. 3. Then, $\alpha[S_1] = \{a_1, a_2, a_3, a_4\}$, and $\uparrow a_5 = \{a_2, a_3\}$. Note that the state s and the set $\uparrow s$ approximate the same set of concrete states, i.e., $\gamma(s) = \gamma(\uparrow s)$. For example, $\gamma(\uparrow a_5) = \gamma(a_5) = (x > 0)$.

Let $e = \langle U, O\rangle$ be a monotone element of $2^S \times 2^S$, and $s \in S$. By monotonicity, $\gamma(s) \subseteq \gamma(U)$ iff $\uparrow s \subseteq U$. Dually, $\gamma(s) \not\subseteq \gamma(\overline{O})$ iff $\uparrow s \not\subseteq \overline{O}$. Thus, we define a new operator red as follows: $\texttt{red}(e) \triangleq \langle \texttt{red}_\mathsf{U}(U), \texttt{red}_\mathsf{O}(O)\rangle$, where $\texttt{red}_\mathsf{U}(U) \triangleq \{s \mid \uparrow s \subseteq U\}$, and $\texttt{red}_\mathsf{O}(O) \triangleq \{s \mid \uparrow s \not\subseteq \overline{O}\}$.

Theorem 8 *Let S be an abstract statespace, and e be a monotone element in $2^S \times 2^S$. Then, $\texttt{red}(e) = \texttt{RED}(e)$.*

red can be computed effectively since it does not need to concretize abstract elements.

In this section, we have introduced a new compositional semantics RCS, and shown that it is more precise than SCS, and that GKMTSs and MixTSs are equivalent w.r.t. RCS. RCS can be computed effectively on monotone models, which is not restrictive since monotone models are as expressive as their non-monotone counterparts.

VI. Symbolic Computation of RCS using BDDs

In this section, we describe a symbolic algorithm RCS that implements the RCS semantics for monotone models

```
1: global var Rmay, Rmust : BDD
2: func RCS(Expr φ) : BDD
3:   match φ with
4:     ATOMIC(p) : return ABSV(BDDVAR("p"),
                              BDDVAR("p"))
5:     ¬ψ : return ABSNOT(RCS(ψ))
6:     ψ1∧ψ2 : return ABSAND(RCS(ψ1), RCS(ψ2))
7:     ψ1 ∨ ψ2 : return ABSOR(RCS(ψ1), RCS(ψ2))
8:     ◇ψ : return ABSPRE(Rmay, Rmust, RCS(ψ))
9:     μψ : return RCS_lfp(RCS(ψ))
10:    νψ : return RCS_gfp(RCS(ψ))
11:
12: func ABSV(BDD u, BDD o) : BDD
13:   sel := BDDVAR("sel")
14:   return BDDITE(sel, u, o)
15:
16: func ABSO(BDD v) = v[0/sel]
17: func ABSU(BDD v) = v[1/sel]
18: func ABSAND(BDD v1, BDD v2) = BDDAND(v1, v2)
19: func ABSOR(BDD v1, BDD v2) = BDDOR(v1, v2)
20: func ABSEQ(BDD v1, BDD v2) = BDDEQ(v1, v2)
21:
22: func ABSNOT(BDD v) : BDD
23:   o := ABSO(v), u := ABSU(v)
24:   return ABSV(BDDNOT(o), BDDNOT(u))
25:
26: func ABSREDU(BDD v) : BDD
27:   if (BDDISCONST(v)) return v
28:   b := BDDROOTVAR(v), h := UVAR(b)
29:   T := ABSREDU(v[1/b]), F := ABSREDU(v[0/b])
30:   tmp := BDDITE(b, T, F)
31:   return BDDITE(h, BDDAND(T, F), tmp)
32:
33: func ABSPRE(BDD Rmay, BDD Rmust, BDD v) : BDD
34:   o := ABSO(v), u := ABSREDU(ABSU(v))
35:   return ABSV(BDDPRE(Rmust, u), BDDPRE(Rmay, o))
```

Fig. 4. The RCS algorithm and its supporting functions.

constructed using predicate abstraction. These are the models used by an existing software model-checker [13].

Our implementation is based on the following observation. Let S be an abstract statespace. Then, for any monotone element of $2^S \times 2^S$ there exists a semantically equivalent element in $2^{\alpha[S]} \times 2^{\alpha[S]}$.

Theorem 9 *Let $e_1 = \langle U_1, O_1\rangle$ be a monotone element of $2^S \times 2^S$, and $e_2 = \langle U_2, O_2\rangle$ be in $2^{\alpha[S]} \times 2^{\alpha[S]}$. If $U_1 \cap \alpha[S] = U_2$ and $O_1 \cap \alpha[S] = O_2$, then $e_1 \equiv_a e_2$.*

This allows us to restrict the algorithm to sets over $\alpha[S]$ instead of sets over S. Another consequence of Thm. 9 is that the transition relations can be simplified as well, since we only need the result of the pre-image in the states in $\alpha[S]$.

Theorem 10 *Let $R^{\text{may}} \subseteq S \times S$ and $R^{\text{must}} \subseteq S \times S$ be the* may *and* must *transition relations of a monotone MixTS, respectively, and $e = \langle U, O\rangle$ be a monotone element of $2^S \times 2^S$. Define $\hat{U} \triangleq U \cap \alpha[S]$, $\hat{O} \triangleq O \cap \alpha[S]$, $\hat{R}^{\text{must}} \triangleq R^{\text{must}} \cap (\alpha[S] \times S)$, and $\hat{R}^{\text{may}} \triangleq R^{\text{may}} \cap (\alpha[S] \times \alpha[S])$. Then,*

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

The algorithm RCS is shown in Fig. 4. 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. The algorithm works recursively on the structure of the input formula φ. The fixpoints are computed in the usual way, by iterating until convergence. We describe the details of the implementation below.

Let $P = \{p_1, \ldots, p_n\}$ be a set of n predicates. Recall that $\text{Mon}(P)$ denotes the set of monomials over P, and $\text{MT}(P)$ — the set of minterms over P. Furthermore, $\alpha[\text{Mon}(P)] = \text{MT}(P)$. 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. By Thm. 10, we assume that the transition relations are restricted s.t. $R^{\text{may}} \subseteq \text{MT}(P) \times \text{MT}(P)$, 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, $B' = \{b'_i \mid p_i \in P\}$ – 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.

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 $B \cup H$ as follows:

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

Intuitively, given 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. For example, $(\neg h_1 \wedge b_1) \wedge (\neg h_2 \wedge b_2) \wedge h_3$ 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 $(\texttt{sel} \wedge U) \vee (\neg \texttt{sel} \wedge O)$, where sel is a designated BDD variable. This encoding is implemented by ABSV. U and O elements of the pair 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 Sec. II.

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 $(b_1 \wedge b_2 \wedge b_3) \wedge ((\neg u'_1 \wedge b'_1) \wedge (\neg u'_2 \wedge \neg b'_2) \wedge u'_3)$.

ABSREDU implements the $\texttt{red}_{\texttt{U}}$ reduction operator of Sec. V-C, using the following observation: let $Q \subseteq \text{Mon}(P)$ be a monotone subset, and $a \in \text{Mon}(P)$. If $a \in \text{MT}(P)$, then $\uparrow a \subseteq Q \Leftrightarrow a \in Q$; otherwise, $\uparrow a \subseteq Q$ iff $\uparrow(a \wedge p) \subseteq Q$ and $\uparrow(a \wedge \neg p) \subseteq Q$, where p is a term not occurring in a. ABSREDU applies this reasoning recursively on the input diagram. It uses a function UVAR to find a variable $h_i \in H$

for each variable $b_i \in B$. The function ABSPRE implements the pre-image computation based on Thm. 10.

Theorem 11 *For a monotone MixTS $\mathcal{M}$ and $\varphi \in L_\mu$, the function* RCS(φ) *returns the symbolic representation of* $\|\varphi\|_r^{\mathcal{M}}$.

VII. EXPERIMENTS

We have implemented symbolic algorithms for computing both SCS and RCS using the CUDD [22] library. Our goal was to evaluate the cost and performance of RCS versus SCS on a realistic model. Understanding and analyzing RCS in the context of abstraction refinement and software model-checking is left for future work.

For the model, we used a template program built out of n blocks, each based on an example from [20] and having one integer variable. The method of [11] was applied to build an abstract MixTS via predicate abstraction. We checked one reachability (least fixed-point) property, $\texttt{Prop}_1$, and two non-termination (greatest fixed-point) properties, $\texttt{Prop}_2$, and $\texttt{Prop}_3$. The code for the experiments is available from http://www.cs.toronto.edu/~owei/MixTS/FMCAD08.html.

The results are summarized in Fig. 5. The top part of the table shows that, as expected, the model for RCS is significantly smaller, in the # of DD nodes, than the model for SCS. RCS is always more precise than SCS, and the extra precision changes the number of iterations of the fixpoint computation. For $\texttt{Prop}_1$, RCS requires more iterations, and takes more time than SCS. For $\texttt{Prop}_2$, RCS is a lot faster than SCS— the fixpoint computation in the former converges in just two steps, whereas the number of iterations in latter is proportional to the size of the model. For $\texttt{Prop}_3$, where RCS and SCS take the same number of iterations, RCS performs significantly better. In all the cases, the time spent in ABSREDU, which represents the main difference between the two semantics, is only about 20% - 25% of the total time.

These experiments suggest that the additional precision of RCS improves the overall performance of model checking, making it a viable alternative to SCS in practice.

	n	SCS		RCS		
Model Size	100	370,070		216,689		
	150	825,112		482,531		
	200	1,460,270		853,389		
	250	2,275,196		1,329,215		
Prop.	n	Analysis (sec.)	Num. of Iterations	Analysis (sec.)	ABSREDU (sec.)	Num. of Iterations
$\texttt{Prop}_1$	100	2.20	301	3.60	0.74	401
	150	6.66	451	12.12	2.57	601
	200	15.36	601	27.77	6.45	801
	250	28.92	751	55.19	13.40	1001
$\texttt{Prop}_2$	100	3.60	203	0.03	$< 10^{-4}$	2
	150	11.91	303	0.07	$< 10^{-4}$	2
	200	27.16	403	0.12	$< 10^{-4}$	2
	250	54.62	503	0.19	$< 10^{-4}$	2
$\texttt{Prop}_3$	100	33.96	400	21.24	4.5	400
	150	137.38	600	76.38	15.77	600
	200	395.24	800	258.72	42.44	800
	250	1108.67	1000	546.88	101.20	1000

Fig. 5. Experimental results for SCS and RCS.

VIII. RELATED WORK AND CONCLUSION

Related Work. Godefroid and Jagadeesan [9], and Gurfinkel and Chechik [10] proved that the models in the KMTS family have the same expressive power and are equally precise for SCS. Dams and Namjoshi [6] showed that the three families considered in this paper are subsumed by tree automata. Our paper completes 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 [6]) with one another.

In this paper, we did not consider Hyper TSs (HTSs) [21] which allow for both *must* and *may* hyper-transitions. As pointed out in [21], *may* hyper-transitions can be eliminated by increasing the abstract statespace, making HTSs exactly as expressive as GKMTSs.

Our reduction operator RED is an instance of normalization from Abstract Interpretation [4], typically used to provide a canonical representation of equivalent abstract properties. Our symbolic implementation ABSREDU is similar to the semantic minimization of 3-valued propositional formulas [19].

Since RCS is compositional, its precision is between SCS and thorough semantics. Thus, existing results comparing SCS and thorough semantics, i.e., [8,10,18], apply to RCS as well. It is interesting to investigate whether the additional precision enjoyed by RCS can be used to improve the above results.

Summary. In this paper, we compared three families of partial modeling formalisms: KMTSs, MixTSs and GKMTSs. We showed that they are equally expressive – a model of one formalism can be transformed into an equivalent model of another. Thus, neither hyper-transitions nor restrictions on *may* and *must* transitions affect expressiveness; they only affect the succinctness of the formalism.

We also proposed a reduced compositional semantics for partial models. This semantics is more precise than the standard one, MixTSs and GKMTSs are equivalent w.r.t. it. We also provided a symbolic implementation of the new semantics, and our experiments suggested that the new semantics is a good alternative to the standard one for predicate abstraction-based model-checkers.

Acknowledgments. We thank Sagar Chaki for comments on earlier drafts of this paper.

REFERENCES

[1] G. Bruns and P. Godefroid. "Model Checking Partial State Spaces with 3-Valued Temporal Logics". In *CAV*, vol. 1633 of *LNCS*, pp. 274–287, 1999.

[2] G. Bruns and P. Godefroid. "Generalized Model Checking: Reasoning about Partial State Spaces". In *CONCUR*, vol. 1877 of *LNCS*, pp. 168–182, 2000.

[3] M. Chechik, B. Devereux, S. Easterbrook, and A. Gurfinkel. "Multi-Valued Symbolic Model-Checking". *ACM TOSEM*, 12(4):1–38, 2003.

[4] P. Cousot and R. Cousot. "Abstract Interpretation Frameworks". *J. of Logic and Computation*, 2(4):511–547, 1992.

[5] D. Dams, R. Gerth, and O. Grumberg. "Abstract Interpretation of Reactive Systems". *ACM TOPLAS*, 2(19):253–291, 1997.

[6] D. Dams and K. S. Namjoshi. "Automata as Abstractions". In *VMCAI*, vol. 3385 of *LNCS*, pp. 216–232, 2005.

[7] L. de Alfaro, P. Godefroid, and R. Jagadeesan. "Three-Valued Abstractions of Games: Uncertainty, but with Precision". In *LICS*, pp. 170–179, 2004.
[8] P. Godefroid and M. Huth. "Model Checking v.s. Generalized Model Checking: Semantic Minimizations for Temporal Logics". In *LICS*, pp. 158–167, 2005.
[9] P. Godefroid and R. Jagadeesan. "On the Expressiveness of 3-Valued Models". In *VMCAI*, vol. 2575 of *LNCS*, pp. 206–222, 2003.
[10] A. Gurfinkel and M. Chechik. "How Thorough is Thorough Enough". In *CHARME*, vol. 3725 of *LNCS*, pp. 65–80, 2005.
[11] A. Gurfinkel and M. Chechik. "Why Waste a Perfectly Good Abstraction?". In *TACAS*, vol. 3920 of *LNCS*, pp. 212–226, 2006.
[12] A. Gurfinkel, O. Wei, and M. Chechik. "Systematic Construction of Abstractions for Model-Checking". In *VMCAI*, vol. 3855 of *LNCS*, pp. 381–397, 2006.
[13] A. Gurfinkel, O. Wei, and M. Chechik. "YASM: A Software Model-Checker for Verification and Refutation". In *CAV*, vol. 4144 of *LNCS*, pp. 170–174, 2006.
[14] M. Huth, R. Jagadeesan, and D. A. Schmidt. "Modal Transition Systems: A Foundation for Three-Valued Program Analysis". In *ESOP*, vol. 2028 of *LNCS*, pp. 155–169, 2001.
[15] D. Kozen. "Results on the Propositional μ-calculus". *Theoretical Computer Science*, 27:334–354, 1983.
[16] K. Larsen and B. Thomsen. "A Modal Process Logic". In *LICS*, pp. 203–210, 1988.
[17] P. Larsen. "The Expressive Power of Implicit Specifications". In *ICALP*, vol. 510 of *LNCS*, pp. 204–216, 1991.
[18] S. Nejati, M. Gheorghiu, and M. Chechik. "Thorough Checking Revisited". In *FMCAD*, pp. 106–116, 2006.
[19] T. W. Reps, A. Loginov, and S. Sagiv. "Semantic Minimization of 3-Valued Propositional Formulae". In *LICS*, pp. 40–54, 2002.
[20] S. Shoham and O. Grumberg. "Monotonic Abstraction-Refinement for CTL". In *TACAS*, vol. 2988 of *LNCS*, pp. 546–560, 2004.
[21] S. Shoham and O. Grumberg. "3-Valued Abstraction: More Precision at Less Cost". In *LICS*, pp. 399–410, 2006.
[22] F. Somenzi. "CUDD: CU Decision Diagram Package Release", 2001.

APPENDIX

The appendix contains proof sketches of theorems for the convenience of the reviewers.

Theorem 2 *Let G be a GKMTS, and $M = \text{GTOM}(G)$. Then, M is a MixTS, and G and M are semantically equivalent.*

Proof: (1) According to the construction in Def. 6, every must hyper-transition is replaced by a regular one. It is easy to show that 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 $\rho_M : C \times S_M$ is a mixed simulation between B and M. This is proved based on the construction of transition relations given in Def. 6. ■

Theorem 3 *Let $M = \langle S_M, R_M^{\text{must}}, R_M^{\text{may}}\rangle$ be a monotone (Def. 9) MixTS transition system, then the following are equivalent:*

1) *M is semantically consistent, i.e., $\mathbb{C}[M]$ is not empty.*
2) *M is logically consistent, i.e., $\langle M, L\rangle$ is logically consistent for every (consistent) labeling function L.*
3) *for any $a, b_1 \in S_M$, $a \xrightarrow{\text{must}} b_1 \Rightarrow \exists b_2 \in S_M \cdot b_1 \preceq b_2 \wedge a \xrightarrow{\text{may}} b_2$.*

Proof:

- $(1) \Rightarrow (2)$. Trivial (by Theorem 2)
- $(2) \Rightarrow (3)$. Given $a \xrightarrow{\text{must}} b_1$, we prove the result by constructing a consistent labeling function L such that for some atomic proposition p, to ensure the value $\|\Diamond p\|(a)$ is consistent, i.e., belonging to $\{\mathsf{t}, \mathsf{f}, \mathsf{m}\}$, a must have a *may* transition to some state b_2 such that $b_1 \preceq b_2$.
  We construct the labeling function L as follows.
  We define three sets S_1, S_2, and S_3.

$$S_1 = \{s \in S_M \mid b_1 \preceq s\}$$

That is, S_1 is the set of states which are more precise than b_1. Let

$$S_2 = \{s \in S_M \mid \exists t \in S_1 \cdot s \preceq t\} \setminus S_1$$

That is, S_2 is the set of states which are less precise than some state in S_1, but does not include the states in S_1 themselves. Let

$$S_3 = S_M \setminus (S_1 \cup S_2)$$

That is, the set of states which are outside of S_1 and S_2. Note that S_1, S_2 and S_2 are disjoint.
Let $AP = \{p\}$ and the labeling function L is defined as follows:

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

That is, p is true in S_1, unknown in S_2, and false in S_3. We show that L is consistent:

- at most one of p or $\neg p$ belongs to $L(s)$ for any $s \in S_M$
- L is monotone, which follows from the following fact: for any $s, t \in S_M$
  1) $s, t \in S_1$:
     In this case $L(s) = L(t)$, and the monotonicity condition is always satisfied. Same result holds for the cases of $s, t \in S_2$, and $s, t \in S_3$.
  2) $s \in S_1$ and $t \in S_2$: In this case, suppose $s \preceq t$, then $b_1 \preceq t$, and $t \in S_1$; contradiction.
     Therefore, the only possible ordering relation between s and t is $t \preceq s$. Since $L(t) \subseteq L(s)$, the monotonicity condition is satisfied.
  3) $s \in S_2$ and $t \in S_3$:
     In this case, suppose $t \preceq s$, then there exists $s' \in S_1$, and $t \preceq s'$, i.e., $t \in S_2$; contradiction.
     Therefore, the only possible ordering relation between s and t is $s \preceq t$. Since $L(s) \subseteq L(t)$, monotone condition holds.
  4) $s \in S_3$ and $t \in S_1$:
     In this case, suppose $s \preceq t$, then $s \in S_2$; a contradiction. Moreover, suppose $t \preceq s$, then $s \in S_1$; a contradiction. Therefore, neither $s \preceq t$ nor $t \preceq s$ holds. Monotone condition is satisfied.

We now show that

$$\exists b_2 \in S_M \cdot b_1 \preceq b_2 \wedge a \xrightarrow{\text{may}} b_2 \qquad (1)$$

$a \xrightarrow{\text{must}} b_1$
$\Rightarrow$ (by definition of L, $\|p\| = \langle S_1, S_1 \cup S_2 \rangle$)
$a \xrightarrow{\text{must}} b_1 \wedge b_1 \in \mathsf{U}(\|p\|)$
$\Rightarrow$ (by definition of $\|\Diamond p\|$)
$a \in \mathsf{U}(\|\Diamond p\|)$
$\Rightarrow$ (since $\|\Diamond p\|(a)$ is consistent)
$a \in \mathsf{O}(\|\Diamond p\|)$
$\Rightarrow$ (by definition of $\|\Diamond p\|$)
$\exists b \in S_1 \cup S_2 \cdot a \xrightarrow{\text{may}} b$

Consider different cases of b,

- $b \in S_1$: by definition of S_1, $b_1 \preceq b$; let $b_2 = b$, then (1) holds.
- $b \in S_2$: in this case, we have that

$\exists b \in S_2 \cdot a \xrightarrow{\text{may}} b$
$\Rightarrow$ (by definition of S_2)
$\exists b \in S_2 \cdot a \xrightarrow{\text{may}} b \wedge \exists b' \in S_1 \cdot b \preceq b'$
$\Rightarrow$ (by monotonicity of M)
$\exists b' \in S_1 \cdot a \xrightarrow{\text{may}} b'$
$\Rightarrow$ (by definition of S_1)
$\exists b' \in S_1 \cdot a \xrightarrow{\text{may}} b' \wedge b_1 \preceq b'$

let $b_2 = b'$, then (1) holds.

- (3) $\Rightarrow$ (1): we prove the result by constructing a BTS $B = \langle C, R \rangle$ such that $M \preceq_\rho B$, where C is the concrete statespace approximated by S_M and $\rho \subseteq C \times S_M$ is the soundness relation.
The BTS B be defined as follows: for any $c, d \in C$,

$$(c, d) \in R \Leftrightarrow \exists b \in S_M \cdot \alpha(c) \xrightarrow{\text{may}} b \wedge (d, b) \in \rho$$

We now show that $M \preceq_\rho B$, i.e., ρ is a mixed simulation between M and B.
For every $(c, a) \in \rho$, we have the following results

- for any $b \in A$

$a \xrightarrow{\text{must}} b$
$\Rightarrow$ (since $a \preceq \alpha(c)$, by monotonicity of M)
$\alpha(c) \xrightarrow{\text{must}} b$
$\Rightarrow$ (by assumption)
$\alpha(c) \xrightarrow{\text{must}} b \wedge \exists b' \cdot b \preceq b' \wedge \alpha(c) \xrightarrow{\text{may}} b'$
$\Rightarrow$ (let d be a state in $\gamma(b')$; by the construction of B)
$\exists b' \cdot b \preceq b' \wedge c \rightarrow d \wedge (d, b') \in \rho$
$\Rightarrow$ (since $b \preceq b'$, $\gamma(b') \subseteq \gamma(b)$)
$c \rightarrow d \wedge (d, b) \in \rho$

- for any $d \in C$, suppose $c \rightarrow d$, i.e., $c \xrightarrow{\text{may}} d$, then

$c \xrightarrow{\text{may}} d$
$\Rightarrow$ (by the construction of B)
$\exists b \in S_M \cdot \alpha(c) \xrightarrow{\text{may}} b \wedge (d, b) \in \rho$
$\Rightarrow$ (since $a \preceq \alpha(c)$, by monotonicity of M)
$\exists b \in S_M \cdot a \xrightarrow{\text{may}} b \wedge (d, b) \in \rho$

■

Theorem 4 *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 Def. 7 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 $\rho_K : C \times S_K$ is a mixed simulation between B and K. The proof follows from the construction of transition relations in Def. 7. ■

Theorem 5 *RCS is more precise than SCS: $\|\varphi\|_c \preceq_a \|\varphi\|_r$.*

Proof: The proof is by structural induction on φ. For the base case, it is obvious that for any atomic proposition p, $\|p\|_c \equiv_a \|p\|_r$. In the following, we show the inductive case for $\Diamond\varphi$; the proofs of other cases are trivial.

We show that if $||\varphi||_c \preceq_a ||\varphi||_r$, then $||\Diamond\varphi||_c \preceq_a ||\Diamond\varphi||_r$, i.e.,

$$||\varphi||_c \preceq_a ||\varphi||_r \Rightarrow \gamma(\mathsf{U}(||\Diamond\varphi||_c)) \subseteq \gamma(\mathsf{U}(||\Diamond\varphi||_r)) \quad (2)$$

and

$$||\varphi||_c \preceq_a ||\varphi||_r \Rightarrow \gamma(\overline{\mathsf{O}(||\Diamond\varphi||_c)}) \subseteq \gamma(\overline{\mathsf{O}(||\Diamond\varphi||_r)}) \quad (3)$$

Proof of (2). For any two sets Q_1, Q_2, we have that

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

This follows from the following derivation: suppose $Q_1 \nsubseteq \texttt{RED}_\mathsf{U}(Q_2)$, then there exists a state s s.t. $s \in Q_1$ and $s \notin \texttt{RED}_\mathsf{U}(Q_2)$. By the definition of $\texttt{RED}_\mathsf{U}$, $\gamma(s) \nsubseteq \gamma(Q_2)$; on the other hand, since $\gamma(Q_1) \subseteq \gamma(\texttt{RED}_\mathsf{U}(Q_2)) = \gamma(Q_2)$, $\gamma(s) \subseteq \gamma(Q_2)$. A contradiction.

The proof of (2) is shown below.

$||\varphi||_c \preceq_a ||\varphi||_r$
$\Rightarrow$ (by the definition of $\preceq_a$)
$\gamma(\mathsf{U}(||\varphi||_c) \subseteq \gamma(\mathsf{U}(||\varphi||_r))$
$\Rightarrow$ (since $\gamma(Q) = \gamma(\texttt{RED}_\mathsf{U}(Q))$)
$\gamma(\mathsf{U}(||\varphi||_c) \subseteq \gamma(\texttt{RED}_\mathsf{U}(\mathsf{U}(||\varphi||_r)))$
$\Rightarrow$ (by **(P1)**)
$\mathsf{U}(||\varphi||_c)) \subseteq \texttt{RED}_\mathsf{U}(\mathsf{U}(||\varphi||_r))$
$\Rightarrow$ (since $Q_1 \subseteq Q_2 \Rightarrow pre_\mathsf{U}(Q_1) \subseteq pre_\mathsf{U}(Q_2)$)
$pre_\mathsf{U}(\mathsf{U}(||\varphi||_c)) \subseteq pre_\mathsf{U}(\texttt{RED}_\mathsf{U}(\mathsf{U}(||\varphi||_r)))$
$\Rightarrow$ (by the definition of γ)
$\gamma(pre_\mathsf{U}(\mathsf{U}(||\varphi||_c))) \subseteq \gamma(pre_\mathsf{U}(\texttt{RED}_\mathsf{U}(\mathsf{U}(||\varphi||_r))))$
$\Rightarrow$ (since $\gamma(Q) = \gamma(\texttt{RED}_\mathsf{U}(Q))$)
$\gamma(pre_\mathsf{U}(\mathsf{U}(||\varphi||_c))) \subseteq \gamma(\texttt{RED}_\mathsf{U}(pre_\mathsf{U}(\texttt{RED}_\mathsf{U}(\mathsf{U}(||\varphi||_r)))))$
$\Rightarrow$ (by the definition of SCS and RCS)
$\gamma(\mathsf{U}(||\Diamond\varphi||_c)) \subseteq \gamma(\mathsf{U}(||\Diamond\varphi||_r))$

Proof of (3). Dual of the one above. ■

Theorem 6 *Let $\mathcal{G}$ be a GKMTS, and $\mathcal{M} = \text{GTOM}(\mathcal{G})$. Then, $\mathcal{G}$ and $\mathcal{M}$ are equivalent w.r.t. RCS: $\forall\varphi \in L_\mu \cdot \|\varphi\|_r^\mathcal{G} \equiv_a \|\varphi\|_r^\mathcal{M}$.*

Proof: The proof is by structural induction on φ. For the base case, according to the definition of L_M, we have that for any atomic proposition p, $\|p\|_r^\mathcal{G} \equiv_a \|p\|_r^\mathcal{M}$. In the following, we show the inductive case for $\Diamond\varphi$; the proofs of the other cases are trivial.

We show that if $\|\varphi\|_r^{\mathcal{G}} \equiv_a \|\varphi\|_r^{\mathcal{M}} \Rightarrow \|\Diamond\varphi\|_r^{\mathcal{G}} \equiv_a \|\Diamond\|_r\varphi^{\mathcal{M}}$, i.e.,

$$\|\varphi\|_r^{\mathcal{G}} \equiv_a \|\varphi\|_r^{\mathcal{M}} \Rightarrow \gamma(\mathsf{U}(\|\Diamond\varphi\|_r^{\mathcal{G}})) \subseteq \gamma(\mathsf{U}(\|\Diamond\varphi\|_r^{\mathcal{M}})) \quad (4)$$

and

$$\|\varphi\|_r^{\mathcal{G}} \equiv_a \|\varphi\|_r^{\mathcal{M}} \Rightarrow \gamma(\overline{\mathsf{O}(\|\Diamond\varphi\|_r^{\mathcal{G}})}) \subseteq \gamma(\overline{\mathsf{O}(\|\Diamond\varphi\|_r^{\mathcal{M}})}) \quad (5)$$

Proof of (4). For any concrete state c and a set of abstract states Q,

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

The proof of (4) is as follows: for any concrete state c,

$c \in \gamma(\mathsf{U}(\|\Diamond\varphi\|_r^{\mathcal{G}}))$
$\Leftrightarrow$ (by the definition of RCS)
$c \in \gamma(\mathtt{RED}_{\mathsf{U}}(pre_{\mathsf{U}}^{G}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}})))))$
$\Leftrightarrow$ (($\Rightarrow$) let a be the abstract state in **(P2)**,
($\Leftarrow$) since $\gamma(Q) = \gamma(\mathtt{RED}_{\mathsf{U}}(Q))$)
$c \in \gamma(a) \wedge a \in pre_{\mathsf{U}}^{G}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}})))$
$\Leftrightarrow$ (by the definition of pre_{U})
$c \in \gamma(a) \wedge \exists Q \subseteq \mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}})) \cdot R_G^{\text{must}}(a, Q)$
$\Leftrightarrow$ (by the construction in Def. 6 and **(P2)**)
$c \in \gamma(a) \wedge \exists b \cdot \gamma(b) \subseteq \gamma(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}}))) \wedge R_M^{\text{must}}(a, b)$
$\Leftrightarrow$ (since $\|\varphi\|_r^{\mathcal{G}} \equiv_a \|\varphi\|_r^{\mathcal{M}}$, $\gamma(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}})) = \gamma(\mathsf{U}(\|\varphi\|_r^{\mathcal{M}}))$)
$c \in \gamma(a) \wedge \exists b \cdot \gamma(b) \subseteq \gamma(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{M}}))) \wedge R_M^{\text{must}}(a, b)$
$\Leftrightarrow$ (since $\gamma(Q) = \gamma(\mathtt{RED}_{\mathsf{U}}(Q))$, by the definition of $\mathtt{RED}_{\mathsf{U}}$)
$c \in \gamma(a) \wedge \exists b \in \mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{M}})) \cdot R_M^{\text{must}}(a, b)$
$\Leftrightarrow$ (by the definition of pre_{U})
$c \in \gamma(a) \wedge a \in pre_{\mathsf{U}}^{M}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{M}})))$
$\Leftrightarrow$ (($\Rightarrow$) since $\gamma(Q) = \gamma(\mathtt{RED}_{\mathsf{U}}(Q))$,
($\Leftarrow$) let a be the abstract state in **(P2)**)
$c \in \gamma(\mathtt{RED}_{\mathsf{U}}(pre_{\mathsf{U}}^{M}(\mathtt{RED}_{\mathsf{U}}(\mathsf{U}(\|\varphi\|_r^{\mathcal{G}})))))$
$\Leftrightarrow$ (by the definition of RCS)
$c \in \gamma(\mathsf{U}(\|\Diamond\varphi\|_r^{\mathcal{M}}))$

Proof of (5). The proof is similar to the one above. It is based on the observation that for any concrete state c and a set of abstract states Q, $c \in \gamma(\overline{\mathtt{RED}_{\mathsf{O}}(Q)}) \Leftrightarrow \exists a \in \overline{Q} \cdot c \in \gamma(a)$. ■

Theorem 7 *Let S be an abstract statespace, and e be a monotone element in $2^S \times 2^S$. Then,* $\mathtt{red}(e) = \mathtt{RED}(e)$.

Proof: According to the definition of RED, the proof of the theorem is equivalent to showing that for any $e = \langle U, O\rangle \in 2^S \times 2^S$ and $s \in S$,

$$\gamma(s) \subseteq \gamma(U) \Leftrightarrow \uparrow s \subseteq U \quad (6)$$

and

$$\gamma(s) \nsubseteq \gamma(\overline{O}) \Leftrightarrow \uparrow s \nsubseteq \overline{O} \quad (7)$$

Proof of (6). The ($\Rightarrow$) direction follows from the following derivation: supposing that $\uparrow s \nsubseteq U$, we have that

$\uparrow s \nsubseteq U$
$\Rightarrow$ $\exists a \in S \cdot a \in \uparrow s \wedge a \notin U$
$\Rightarrow$ (since $a \in \uparrow s$, $a \in \alpha[S]$)
$\exists a \in S \cdot a \in \uparrow s \wedge a \notin U \wedge \exists c \in C \cdot a = \alpha(c)$
$\Rightarrow$ (by the def. of $\uparrow s$, $\gamma(a) \subseteq \gamma(s)$; since $\gamma(s) \subseteq \gamma(U)$)
$\exists a \in S \cdot a \in \uparrow s \wedge a \notin U \wedge \exists c \in C \cdot a = \alpha(c) \wedge$
$c \in \gamma(U)$
$\Rightarrow$ (by the definition of γ)
$\exists a \in S \cdot a \in \uparrow s \wedge a \notin U \wedge \exists c \in C \cdot a = \alpha(c) \wedge$
$c \in \gamma(U) \wedge \exists b \in U \cdot c \in \gamma(b)$
$\Rightarrow$ (by the definition of α)
$\exists a \in S \cdot a \in \uparrow s \wedge a \notin U \wedge \exists c \in C \cdot a = \alpha(c) \wedge$
$c \in \gamma(U) \wedge \exists b \in U \cdot c \in \gamma(b) \wedge b \preceq_a a$
$\Rightarrow$ (by monotonicity of e, $a \in U$)
$\exists a \in S \cdot a \in \uparrow s \wedge a \notin U \wedge a \in U$

A contradiction. The ($\Leftarrow$) direction is trivial.

Proof of (7). Dual of the one above. ■

Theorem 8 *Let $e_1 = \langle U_1, O_1\rangle$ be a monotone element of $2^S \times 2^S$, and $e_2 = \langle U_2, O_2\rangle$ be in $2^{\alpha[S]} \times 2^{\alpha[S]}$. If $U_1 \cap \alpha[S] = U_2$, and $O_1 \cap \alpha[S] = O_2$, then $e_1 \equiv_a e_2$.*

Proof: This is proved by showing that $\mathtt{RED}(e_1) = \mathtt{RED}(e_2)$; since RED is semantics preserving, the result holds. ■

Theorem 9 *Let $R^{\text{may}} \subseteq S \times S$ and $R^{\text{must}} \subseteq S \times S$ be the* may *and* must *transition relations of a monotone MixTS, respectively, and $e = \langle U, O\rangle$ be a monotone element of $2^S \times 2^S$. Define $\hat{U} \triangleq U \cap \alpha[S]$, $\hat{O} \triangleq O \cap \alpha[S]$, $\hat{R}^{\text{must}} \triangleq R^{\text{must}} \cap (\alpha[S] \times S)$, and $\hat{R}^{\text{may}} \triangleq R^{\text{may}} \cap (\alpha[S] \times \alpha[S])$. Then,*

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

Proof: According to the definition of $\equiv_a$, we show that

$$\gamma(pre[R^{\text{must}}](\mathtt{RED}_{\mathsf{U}}(U))) = \gamma(pre[\hat{R}_1](\mathtt{RED}_{\mathsf{U}}(\hat{U}))) \quad (8)$$

and

$$\gamma(pre[R^{\text{may}}](\mathtt{RED}_{\mathsf{O}}(O))) = \gamma(pre[\hat{R}_2](\hat{O})) \quad (9)$$

Proof of (8). The fact that $\gamma(pre[R^{\text{must}}](\mathtt{RED}_{\mathsf{U}}(U))) \subseteq \gamma(pre[\hat{R}_1](\mathtt{RED}_{\mathsf{U}}(\hat{U})))$ is shown as follows. For any concrete state c,

$c \in \gamma(pre[R^{\text{must}}](\mathtt{RED}_{\mathsf{U}}(U)))$
$\Rightarrow$ $\exists a \in S \cdot c \in \gamma(a) \wedge a \in pre[R^{\text{must}}](\mathtt{RED}_{\mathsf{U}}(U))$
$\Rightarrow$ (by the definition of pre)
$\exists a \in S \cdot c \in \gamma(a) \wedge \exists b \in \mathtt{RED}_{\mathsf{U}}(U) \cdot R^{\text{must}}(a, b)$
$\Rightarrow$ (let $a' = \alpha(c)$, by the definition of α)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \exists a \in S \cdot a \preceq_a a' \wedge$
$\exists b \in \mathtt{RED}_{\mathsf{U}}(U) \cdot R^{\text{must}}(a, b)$
$\Rightarrow$ (by the definition of monotone MixTSs)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \exists b \in \mathtt{RED}_{\mathsf{U}}(U) \cdot R^{\text{must}}(a', b)$
$\Rightarrow$ (by the definition of $\hat{R}^{\text{must}}$)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \exists b \in \mathtt{RED}_{\mathsf{U}}(U) \cdot \hat{R}^{\text{must}}(a', b)$
$\Rightarrow$ (since e is a monotone element, by Theorem 8)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \exists b \in \mathtt{red}_{\mathsf{U}}(U) \cdot \hat{R}^{\text{must}}(a', b)$
$\Rightarrow$ (by the definition of $\hat{U}$, $\mathtt{red}_{\mathsf{U}}(U) = \mathtt{red}_{\mathsf{U}}(\hat{U})$)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \exists b \in \mathtt{red}_{\mathsf{U}}(\hat{U}) \cdot \hat{R}^{\text{must}}(a', b)$
$\Rightarrow$ (by the definition of pre)
$c \in \gamma(a') \wedge a' \in pre[\hat{R}_1](\mathtt{RED}_{\mathsf{U}}(\hat{U}))$
$\Rightarrow$ $c \in \gamma(pre[\hat{R}^{\text{must}}](\mathtt{RED}_{\mathsf{U}}(U)))$

The proof of $\gamma(\mathit{pre}[R^{\text{must}}](\mathtt{RED_U}(U))) \supseteq \gamma(\mathit{pre}[\hat{R}_1](\mathtt{RED_U}(\hat{U})))$ is trivial.

Proof of (9). The fact that $\gamma(\overline{\mathit{pre}[R^{\text{may}}](\mathtt{RED_O}(O))}) \subseteq \gamma(\overline{\mathit{pre}[\hat{R}_2](\hat{O})})$ is shown as follows. For any concrete state c,

$c \in \gamma(\overline{\mathit{pre}[R^{\text{may}}](\mathtt{RED_O}(O))})$
$\Rightarrow$ $\exists a \in S \cdot c \in \gamma(a) \wedge a \in \overline{\mathit{pre}[R^{\text{may}}](\mathtt{RED_O}(O))}$
$\Rightarrow$ (by the definition of pre)
$\exists a \in S \cdot c \in \gamma(a) \wedge R^{\text{may}}(a) \subseteq \overline{\mathtt{RED_O}(O)} \wedge c \in \gamma(a)$
$\Rightarrow$ (let $a' = \alpha(c)$, by the definition of α)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge$
$\exists a \in S \cdot a \preceq_a a' \wedge R^{\text{may}}(a) \subseteq \overline{\mathtt{RED_O}(O)}$
$\Rightarrow$ (by the definition of monotone MixTSs)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge$
$\exists a \in S \cdot R^{\text{may}}(a') \subseteq R^{\text{may}}(a) \subseteq \overline{\mathtt{RED_O}(O)}$
$\Rightarrow$ (by the definition of $\hat{R}^{\text{may}}$, $R^{\text{may}}(a') \cap \alpha[S] = \hat{R}^{\text{may}}(a')$)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \hat{R}^{\text{may}}(a') \subseteq (\overline{\mathtt{RED_O}(O)} \cap \alpha[S])$
$\Rightarrow$ (since e is a monotone element, by Theorem 8)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \hat{R}^{\text{may}}(a') \subseteq (\overline{\mathtt{red_O}(O)} \cap \alpha[S])$
$\Rightarrow$ (by the definition of $\mathtt{red_O}$)
$\forall s \in \alpha[S] \cdot s \in \mathtt{red_O}(O) \Leftrightarrow s \in O$)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \hat{R}^{\text{may}}(a') \subseteq (\overline{O} \cap \alpha[S])$
$\Rightarrow$ (by the definition of $\hat{O}$)
$c \in \gamma(a') \wedge a' \in \alpha[S] \wedge \hat{R}^{\text{may}}(a') \subseteq \alpha[S] \setminus \hat{O}$
$\Rightarrow$ (by the definition of pre)
$c \in \gamma(a') \wedge a' \in \gamma(\overline{\mathit{pre}[\hat{R}_2](\hat{O})})$
$\Rightarrow$ $c \in \gamma(\overline{\mathit{pre}[\hat{R}_2](\hat{O})})$

The proof of $\gamma(\overline{\mathit{pre}[R^{\text{may}}](\mathtt{RED_O}(O))}) \supseteq \gamma(\overline{\mathit{pre}[\hat{R}_2](\hat{O})})$ is similar. ■

Theorem 10 *For a monotone MixTS $\mathcal{M}$ and $\varphi \in L_\mu$, the function* $\mathrm{RCS}(\varphi)$ *returns the symbolic representation of* $\|\varphi\|_r^{\mathcal{M}}$.

Proof: The proof follows from Thm. 8, Thm. 9, and Thm. 10. In particular, Thm. 9 is used to show that in the interpretation of $\Diamond\varphi$ in Def. 8, removing the application of RED after pre_{U} and pre_{O} does not affect precision. ■