

Mixed Transition Systems Revisited

Ou Wei¹, Arie Gurfinkel², and Marsha Chechik¹

¹Department of Computer Science, University of Toronto

²Software Engineering Institute, Carnegie Mellon University

Abstract. Partial models 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 Kripke Modal Transition Systems (KMTSs), with restrictions on necessary and possible behaviors, Mixed Transition Systems (MixTSs), with relaxation on these restrictions, and Generalized Kripke MTSSs (GKMTSs), with hyper-transitions, respectively. In this paper, we compare the three families w.r.t. their expressive power (i.e., what can be modeled, what abstraction can be captured), and the cost and precision of model-checking. We show that these families have the same expressive power (but do differ in succinctness), whereas GKMTSs are more precise (i.e. can establish more properties) for model-checking than the other two families. However, the 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 partial models 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 symbolic 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.

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 abstracting the concrete states, analyzing the resulting abstract system, and 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 is sound for universal properties (e.g., absence of errors), and *under-approximation* – the abstract system contains *less* behaviours than the concrete one and is sound for existential properties (e.g., presence of errors). Abstractions that are sound for arbitrary properties such as full μ -calculus L_μ [14], must combine over- and under-approximation into a *single* model [4, 15]. This can be done by using a model with two types of transitions, *may* and *must*, representing *possible* (or over-approximating), and *necessary* (or under-approximating) behaviours, respectively. We call such models *partial*. Temporal properties over partial models are interpreted using the 3-valued semantics: a property can be either true, false, or *unknown*.

Existing partial modeling formalisms are divided into three separate families. The first is *Kripke Modal Transition Systems* (KMTSs) [13] and their equivalent variants, *Modal TSs* [15], *Partial Kripke Structures* (PKSs) [1], and *3-valued KSs* [2]. KMTSs require that every *must* transition is also a *may* transition. They were introduced as computational models for partial specifications of reactive systems [15] and then adapted

for model-checking [1, 2, 13]. The second is *Mixed Transition Systems* (MixTSs) [4], and equivalently, *Belnap TSs* [11]. MixTSs extend KMTSs by allowing *must only* transitions (i.e., transitions that are *must* but not *may*). MixTSs were introduced in [4] as abstract models for L_μ , and have been used for predicate abstraction and software model-checking in [10]. The third is *Generalized KMTSs* (GKMTSs) [19], and equivalently, *Abstract TSs* [6] and *Disjunctive MTSs* [16]. GKMTSs extend MixTSs by allowing *must hyper-transitions*, (i.e., transitions into sets of states).

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), and (ii) analyzability of the formalisms (i.e., the cost and precision of model-checking).

Expressive Power. 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. 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 all of the above partial models are less expressive than tree automata [5]. We complete the picture by showing the expressive equivalence *between* those formalisms.

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 models as *standard* inductive semantics (SIS). This is the semantics most widely used in practice. A GKMTS G can prove/disprove more properties under SIS than either a corresponding MixTS M or KMTS K obtained from G by semantics-preserving translations. However, while both MixTSs and KMTSs have been used in practical symbolic model-checkers (e.g., [2, 10, 12]), 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 w.r.t. 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 w.r.t. to RIS and apply it to MixTS models constructed using predicate abstraction. We evaluate our implementation empirically against a SIS-based algorithm.

The rest of the paper is organized as follows. Sec. 2 reviews the necessary background on partial models and abstraction. In Sec. 3, 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 Sec. 4, we introduce *reduced* inductive semantics (RIS) for L_μ . In Sec. 5, we present a symbolic model-checking algorithm w.r.t. RIS in the context of predicate abstraction. We report on our experience with this algorithm in Sec. 6. Sec. 7 concludes the paper with a summary and comparison with related work.

2 Preliminaries

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

2.1 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 . For brevity of notation, we denote a statespace using the set S .

Def. 1 (Partial TSs) [4, 8, 13, 19] 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 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 $L : S \rightarrow 2^{Lit(AP)}$ that assigns to each state s a set of literals that are true in s . For a proposition p , 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., $s_1 \preceq_S s_2 \Rightarrow L(s_1) \subseteq L(s_2)$. A pair $\langle M, L \rangle$ of a TS M and a labeling L is called a *model*.

The modal μ -calculus [14] (L_μ) is defined as the set of all formulas satisfying the following grammar: $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \diamond\varphi \mid \mu Z \cdot \varphi(Z)$, 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 do not 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.

A semantics of L_μ is called *inductive* if it is inductive on the syntax of the logic. We refer to the commonly used inductive semantics as *standard* (SIS). We need the following notation. Let $e = \langle U, O \rangle$. We write $U(e)$ and $O(e)$ to denote U and O , respectively; we use operators \sim and \sqcap 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 (SIS) [4, 8, 11, 13, 19]. 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 $\sigma : Var \rightarrow 2^S \times 2^S$. The standard inductive semantics (SIS) of $\varphi \in L_\mu$ is:

$$\begin{aligned} \|p\|_{c,\sigma}^\mathcal{M} &\triangleq \{s \mid p \in L_M(s)\}, \{s \mid \neg p \notin L_M(s)\} \\ \|\neg\varphi\|_{c,\sigma}^\mathcal{M} &\triangleq \sim\|\varphi\|_{c,\sigma}^\mathcal{M} \quad \|\varphi \wedge \psi\|_{c,\sigma}^\mathcal{M} \triangleq \|\varphi\|_{c,\sigma}^\mathcal{M} \sqcap \|\psi\|_{c,\sigma}^\mathcal{M} \quad \|Z\|_{c,\sigma}^\mathcal{M} \triangleq \sigma(Z) \\ \|\diamond\varphi\|_{c,\sigma}^\mathcal{M} &\triangleq \text{pre}_U(\text{U}(\|\varphi\|_{c,\sigma}^\mathcal{M})), \text{pre}_O(\text{O}(\|\varphi\|_{c,\sigma}^\mathcal{M})) \\ \|\mu Z \cdot \varphi\|_{c,\sigma}^\mathcal{M} &\triangleq \text{lfp}^\subseteq(\lambda Q \cdot \text{U}(\|\varphi\|_{c,\sigma[Z \mapsto Q]}^\mathcal{M})), \text{lfp}^\subseteq(\lambda Q \cdot \text{O}(\|\varphi\|_{c,\sigma[Z \mapsto Q]}^\mathcal{M})) \end{aligned}$$

where $Z \in Var$, lfp is the least fixpoint, and the pre-image operators pre_U and pre_O

are defined as follows:

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

2.2 Partial Models and Abstraction

Abstract Statespace. A concrete statespace C is a set of states s.t. for any $c \in C$ and state labeling L , $p \in L(c) \Leftrightarrow \neg p \notin L(c)$. 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 [3], we require that \preceq_a be a partial order (i.e., the order \preceq_S), 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 $\alpha : C \rightarrow S$ to map each concrete element to its best approximation. The image of α is denoted by $\alpha[S] \triangleq \{\alpha(c) \mid c \in C\}$.

Predicate Abstraction. Let n be a natural number, and $P = \{p_1, \dots, p_n\}$ be a set of quantifier-free first-order boolean predicates. 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 soundness relation ρ_P is defined s.t. $(c, s) \in \rho_P$ iff $c \models s$, i.e., c satisfies all predicates in s ; the abstraction $\alpha_P(c) \triangleq (\bigwedge_{c \models p_i} p_i) \wedge (\bigwedge_{c \not\models p_i} \neg p_i)$; $\alpha_P[\text{Mon}(P)] = \text{MT}(P)$; and the approximation ordering is reverse implication, $s \preceq_a t$ iff $s \Leftarrow t$.

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

Def. 3 (Mixed Simulation) [4] Let $M_1 = \langle S_1, R_1^{\text{may}}, R_1^{\text{must}} \rangle$ and $M_2 = \langle S_2, R_2^{\text{may}}, R_2^{\text{must}} \rangle$ 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:

$$\begin{aligned} \forall t_1 \in S_1 \cdot s_1 \xrightarrow{\text{may}} t_1 &\Rightarrow \exists t_2 \in S_2 \cdot s_2 \xrightarrow{\text{may}} t_2 \wedge (t_1, t_2) \in H \\ \forall t_2 \in S_2 \cdot s_2 \xrightarrow{\text{must}} t_2 &\Rightarrow \exists t_1 \in S_1 \cdot s_1 \xrightarrow{\text{must}} t_1 \wedge (t_1, t_2) \in H \end{aligned}$$

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., [19]).

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 $L_M \preceq_\rho L_B$.

Thm. 1 [4] 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, $\gamma(\text{U}(\|\varphi\|_c^{\mathcal{M}})) \subseteq \text{U}(\|\varphi\|_c^{\mathcal{B}})$, and $\text{O}(\|\varphi\|_c^{\mathcal{B}}) \subseteq \gamma(\text{O}(\|\varphi\|_c^{\mathcal{M}}))$.

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 [1] as *thorough semantics*. It is defined as follows: $\|\varphi\|_t^{\mathcal{M}} = \langle U, O \rangle$ iff $a \in U \Leftrightarrow \forall \mathcal{B} \in \mathbb{C}[\mathcal{M}] \cdot \gamma(a) \subseteq U(\|\varphi\|_c^{\mathcal{B}})$, and $a \notin O \Leftrightarrow \forall \mathcal{B} \in \mathbb{C}[\mathcal{M}] \cdot \gamma(a) \subseteq U(\|\neg\varphi\|_c^{\mathcal{B}})$.

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 $\gamma(\overline{O_1}) \subseteq \gamma(\overline{O_2})$.

3 Expressiveness

We show that GKMTSs, MixTSs, and KMTSs are expressively equivalent. Two partial TSs M and M' are *semantically equivalent*, $M \equiv_a M'$, iff they have the same set of concrete refinements. Two modeling formalisms are *expressively equivalent* iff for every TS M from 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 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.

3.1 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 Fig. 1. 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 $(x \leq 0 \wedge \text{even}(x))$, 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_1 in Fig. 1. 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 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. 1 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 below.

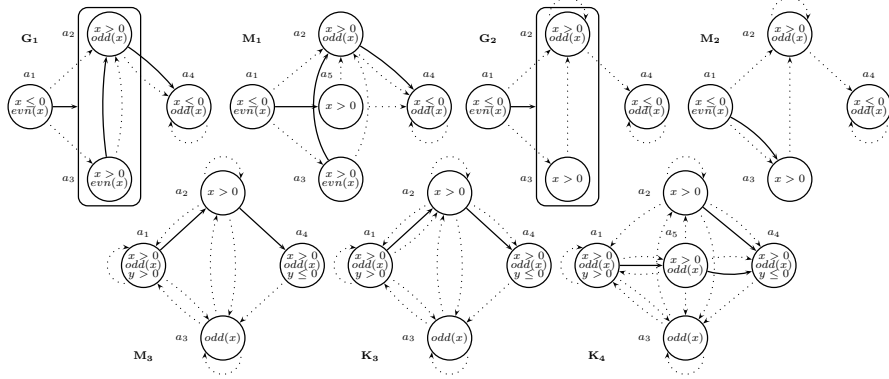


Fig. 1. Two GKMTSs: G_1, G_2 ; three MixTSs: M_1, M_2, M_3 ; two KMTSs: K_3, K_4 . Solid and dashed lines represent must and may transitions, respectively.

Def. 4 (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 $M = \langle S_M, R_M^{\text{may}}, R_M^{\text{must}} \rangle$, such that

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

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

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

The translation GTOM is semantics-preserving.

Thm. 2 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. 2 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. 1 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 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 [3], 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.

3.2 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_3 in Fig. 1. 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. 1. This naive transformation is not semantics-preserving, i.e., $K_3 \not\equiv_a M_3$. For example, the concrete system¹

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

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 $(x > 0 \wedge \text{odd}(x) \wedge y > 0)$, 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 $\gamma(a_2) \cap \gamma(\{a_1, a_3\})$. 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, \dots, b_k , a *must only* transition $b_0 \xrightarrow{\text{must}} b_1$, and a set of *may* transitions $b_0 \xrightarrow{\text{may}} b_2, \dots, b_0 \xrightarrow{\text{may}} b_k$ ensure that every state in $\gamma(b_0)$ has a transition to a state in $\gamma(b_1) \cap \gamma(\{b_2, \dots, b_k\})$.

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 $\gamma(a_4) \cap \gamma(\{a_1, a_2, a_3\}) = \gamma(a_4)$. 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 $\gamma(a_5) = \gamma(a_2) \cap \gamma(\{a_1, a_3\}) = (x > 0 \wedge \text{odd}(x))$, 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_4 in Fig. 1, 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 $a \xrightarrow{\text{must}} \widehat{a \rightarrow b}$ and $a \xrightarrow{\text{may}} \widehat{a \rightarrow b}$, where $\widehat{a \rightarrow b}$ is a (possibly new) abstract state such that $\gamma(\widehat{a \rightarrow b}) = \gamma(b) \cap \gamma(R_M^{\text{may}}(a))$. Second, *may* and *must* transitions are added for all states introduced in the first step. We formalize this below.

Def. 5 (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.

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

where

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

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

In Def. 5, 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 $\text{MTOK}(M_3)$, we have $S^+ = \{a_5\}$, $\text{REPL} = \{(a_1, a_5), (a_2, a_4)\}$, $\text{IMUST} = \emptyset$, $\text{IMO} = \{(a_5, a_4)\}$, and $\text{IMAY} = \{(a_5, a_1), (a_5, a_2), (a_5, a_3)\}$. The result of the translation MTOK is a KMTS: every *must* transition is matched by a *may* transition.

Thm. 3 *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. 3 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. 2 *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 holds because for every state t added by MTOK , there exists a subset $U \subseteq S$ s.t. $\gamma(t) = \gamma(U)$.

4 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 w.r.t. a more tractable inductive semantics SIS. GKMTSs are more precise than MixTSs w.r.t. SIS: for any $\varphi \in L_\mu$, model-checking φ in a GKMTS model \mathcal{G} w.r.t. 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 w.r.t. RIS. In the next section, we provide an efficient symbolic model checking procedure for computing RIS over MixTSs. The outcome is an algorithm that combines the benefits of the efficient symbolic encoding of MixTSs with the model-checking precision of GKMTSs.

In Sec. 4.1, we illustrate the differences between GKMTSs and MixTSs w.r.t. SIS; define RIS in Sec. 4.2; and show how to effectively model-check w.r.t. RIS in Sec. 4.3.

4.1 Example

Let p and q denote predicates ($x > 0$) and $odd(x)$, respectively. Consider the model $\mathcal{G}_1 = \langle G_1, L_{G_1} \rangle$, where G_1 is shown in Fig. 1, and L_{G_1} is a labeling function that labels each abstract state as shown in Fig. 1. 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. 1 and $L_{M_1}(s) \triangleq \mathbf{if} \ s = a_5 \ \mathbf{then} \ \{p\} \ \mathbf{else} \ L_{G_1}(s)$.

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

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

According to \mathcal{G}_1 , in all states corresponding to a_1 , φ is true. 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, although \mathcal{M}_1 and \mathcal{G}_1 are semantically equivalent, \mathcal{M}_1 is less precise than \mathcal{G}_1 for model-checking w.r.t. SIS.

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

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

Since $\gamma(\text{U}(e_1)) = \gamma(\text{U}(e_2))$ and $\gamma(\overline{\text{O}(e_1)}) = \gamma(\overline{\text{O}(e_2)}) = \gamma(\emptyset)$, $e_1 \equiv_a e_2$. However, there is a subtle difference between e_1 and e_2 . In state a_5 of M_1 , $q \vee \neg q$ 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, $\mu Z \cdot (\neg p \wedge q) \vee \diamond Z$, i.e., $EF(\neg p \wedge q)$ in CTL, is true in state a_1 of \mathcal{G}_1 , but is unknown in a_1 of \mathcal{M}_1 .

4.2 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

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 $\text{U}(e)$ is contained in $\text{U}(e')$, and $\text{O}(e)$ contains $\text{O}(e')$. We define the *reduction* operator as follows: $\text{RED}(e) \triangleq \langle \text{RED}_U(U), \text{RED}_O(O) \rangle$, where $\text{RED}_U(U) \triangleq \{s \mid \gamma(s) \subseteq \gamma(U)\}$, and $\text{RED}_O(O) \triangleq \{s \mid \gamma(s) \not\subseteq \gamma(\overline{O})\}$. Intuitively, $\text{RED}(e)$ increases $\text{U}(e)$ and decreases $\text{O}(e)$ as much as possible without affecting the semantic meaning of e . That is, $\text{RED}(e)$ is the largest element w.r.t. information ordering that is semantically equivalent to e . For example, consider $\text{RED}(e_2)$, where e_2 is as defined by (\star) above. Then,

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

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

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

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

$\text{RED}(e)$ 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, $\sim e \equiv_a \sim \text{RED}(e)$, and $e \sqcap e' \equiv_a \text{RED}(e) \sqcap \text{RED}(e')$.

RIS is defined by applying the RED operator before and after \diamond to prevent it from propagating imprecision.

Def. 6 (RIS) Let $\mathcal{M} = \langle M, L_M \rangle$ be a model, s.t. $M = \langle S, R^{\text{may}}, R^{\text{must}} \rangle$ and $\sigma : \text{Var} \rightarrow 2^S \times 2^S$. The reduced inductive semantics of $\varphi \in L_\mu$ is defined as follows:

$$\begin{aligned} \|p\|_{r,\sigma}^{\mathcal{M}} &\triangleq \langle \{s \mid p \in L_M(s)\}, \{s \mid \neg p \notin L_M(s)\} \rangle \\ \|\neg\varphi\|_{r,\sigma}^{\mathcal{M}} &\triangleq \sim \|\varphi\|_{r,\sigma}^{\mathcal{M}} \quad \|\varphi \wedge \psi\|_{r,\sigma}^{\mathcal{M}} \triangleq \|\varphi\|_{r,\sigma}^{\mathcal{M}} \sqcap \|\psi\|_{r,\sigma}^{\mathcal{M}} \quad \|Z\|_{r,\sigma}^{\mathcal{M}} \triangleq \sigma(Z) \\ \|\diamond\varphi\|_{r,\sigma}^{\mathcal{M}} &\triangleq \text{RED}(\langle \text{pre}_U(\text{RED}_U(U(\|\varphi\|_{r,\sigma}^{\mathcal{M}}))), \text{pre}_O(\text{RED}_O(O(\|\varphi\|_{r,\sigma}^{\mathcal{M}}))) \rangle) \\ \|\mu Z \cdot \varphi\|_{r,\sigma}^{\mathcal{M}} &\triangleq \langle \text{lfp}^{\sqsubseteq}(\lambda Q \cdot U(\|\varphi\|_{r,\sigma}^{\mathcal{M}}[Z \mapsto Q])), \text{lfp}^{\sqsubseteq}(\lambda Q \cdot O(\|\varphi\|_{r,\sigma}^{\mathcal{M}}[Z \mapsto Q])) \rangle \end{aligned}$$

The only difference between RIS (Def. 6) and SIS (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, RIS of φ on \mathcal{M}_1 is computed as follows: RIS of q , $\neg q$, and $q \vee \neg q$ is the same as SIS. Thus, $\|q \vee \neg q\|_r^{\mathcal{M}_1} = e_2$. To compute \diamond , recall from $(\star\star)$ that $\text{RED}(e_2) = e_3$; thus, $\|\varphi\|_r^{\mathcal{M}_1} = \langle \{a_1, a_2, a_3, a_5\}, \{a_1, a_2, a_3, a_4, a_5\} \rangle$. Hence, $\|\varphi\|_r^{\mathcal{M}_1}$ is more precise than $\|\varphi\|_c^{\mathcal{M}_1}$.

Thm. 4 *RIS is more precise than SIS:* $\|\varphi\|_c \preceq_a \|\varphi\|_r$.

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

Thm. 5 Let \mathcal{G} be a GKMTS, and $\mathcal{M} = \text{GTOM}(\mathcal{G})$. Then, \mathcal{G} and \mathcal{M} are equivalent w.r.t. RIS: $\forall \varphi \in L_\mu \cdot \|\varphi\|_r^{\mathcal{G}} \equiv_a \|\varphi\|_r^{\mathcal{M}}$

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

4.3 Reduced Inductive Semantics for Monotone Models

We study the reduction operator RED of RIS in the context of monotone models. *Monotone models*, formally defined below, are as expressive as their regular counterparts [11]: for any model there exists an equivalent monotone one. The monotonicity condition simply ensures that all information that can be derived from existing *may* and *must* transitions is made explicit in the model. Furthermore, models built by automated predicate abstraction [10] are monotone by construction. Thus, restricting RED to monotone models is neither a theoretical nor a practical restriction.

Def. 7 A MixTS $M = \langle S, R^{\text{may}}, R^{\text{must}} \rangle$ is monotone iff for any $s_1 \preceq_S s_2, t_2 \preceq_S t_1$,

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

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

```

1: global var Rmay, Rmust : BDD
2: func RIS(Expr  $\varphi$ ) : BDD
3:   match  $\varphi$  with
4:     ATOMIC(p) : return ABSV(BDDVAR("p"),
                             BDDVAR("p"))
5:      $\neg\psi$  : return ABSNOT(RIS( $\psi$ ))
6:      $\psi_1 \wedge \psi_2$  : return ABSAND(RIS( $\psi_1$ ), RIS( $\psi_2$ ))
7:      $\psi_1 \vee \psi_2$  : return ABSOR(RIS( $\psi_1$ ), RIS( $\psi_2$ ))
8:      $\diamond\psi$  : return ABSPRE(Rmay, Rmust, RIS( $\psi$ ))
9:      $\mu\psi$  : return RISifp( $\psi$ )
10:     $\nu\psi$  : return RISgfp( $\psi$ )
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. 2. The RIS algorithm and its supporting functions.

For monotone models, RED can be computed effectively, as we show below.

For a state $s \in S$, the *upset* of s is defined as $\uparrow s \triangleq \{t \in \alpha[S] \mid s \preceq_a t\}$. Thus, $\uparrow s$ 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. 1. 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, $\gamma(s) \not\subseteq \gamma(O)$ iff $\uparrow s \not\subseteq O$. We define a new operator **red** as follows: $\mathbf{red}(e) \triangleq \langle \mathbf{red}_U(U), \mathbf{red}_O(O) \rangle$, where $\mathbf{red}_U(U) \triangleq \{s \mid \uparrow s \subseteq U\}$, and $\mathbf{red}_O(O) \triangleq \{s \mid \uparrow s \not\subseteq O\}$.

Thm. 6 *Let S be an abstract statespace, and e be a monotone element in $2^S \times 2^S$. Then, $\mathbf{red}(e) = \mathbf{RED}(e)$.*

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, shown that it is more precise than SIS, and that GKMTSs and MixTSs are equivalent w.r.t. 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.

5 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 existing software model-checkers [12].

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 $2^{\alpha[S]} \times 2^{\alpha[S]}$.

Thm. 7 *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 theorem allows us to restrict the algorithm to sets over $\alpha[S]$ instead of sets over S . Another consequence of Thm. 7 is that the transition relations can be simplified as well, since we only need the result of the pre-image in the states of $\alpha[S]$.

Thm. 8 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,

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

The algorithm RIS is shown in Fig. 2. 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 $P = \{p_1, \dots, p_n\}$ 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 $L_M(s) = \text{Lit}(s)$, and an L_μ property φ . Without loss of generality, by Thm. 8, 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, $B' = \{b'_i \mid b_i \in B\}$ – 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.

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

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

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 $(\text{se1} \wedge U) \vee (\neg \text{se1} \wedge O)$, where se1 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 Sec. 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 red_U reduction operator of Sec. 4.3 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, 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 Thm. 8.

Thm. 9 *For a monotone MixTS \mathcal{M} and $\varphi \in L_\mu$, algorithm RIS(φ) in Fig. 2 returns the symbolic representation of $\|\varphi\|_r^{\mathcal{M}}$.*

The main difference between the symbolic implementations of SIS and our RIS is the extra ABSREDU operation in function ABSPRE (line 29 in Fig. 2). ABSREDU is similar to existential quantification (BDEXISTS) of BDDs, with one exception: BDEXISTS uses BDDOR in each iteration, but ABSREDU uses one BDDAND and two BDDITE operations. Thus, ABSREDU has the same complexity as BDEXISTS, and symbolic implementations of RIS and SIS also have the same complexity. This means that the extra precision of RIS comes “for free”, without penalty in complexity.

6 Experiments

To empirically evaluate the cost and performance of RIS versus SIS, we have implemented symbolic algorithms for computing both of them using CUDD [21] library, and analyzed reachability and non-termination properties over a realistic model. While our algorithm in Fig. 2 can analyze any μ -calculus formula, our experiments considered just reachability and non-termination properties because of their practical interest.

For the model, we used a template program built out of n blocks, each based on an example from [19] and having one integer variable. The method of [10] was applied to build an abstract MixTS via predicate abstraction. We checked one reachability (least fixed-point) property, Prop₁, and two non-termination (greatest fixed-point) properties, Prop₂, and Prop₃, computing the standard and reduced semantics of the properties. In both cases, we measured 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 considered two sets of initial states, I₁ and I₂, and checked whether conclusive results can be obtained over them.

The results are summarized in Fig. 3. The top part of the table shows that RIS models enjoy significantly smaller encodings than their SIS counterparts, due to restricted transition relations (see Thm. 8). RIS is more precise than SIS: for the two sets of initial states, RIS produces conclusive results for both of them w.r.t. the three properties being checked, whereas SIS cannot decide whether Prop₁ and Prop₂ hold in I₂. As expected, the extra precision of RIS does not cause a complexity penalty: the experiments show that the increases of the analysis time w.r.t. 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.

Note that RIS and SIS may require different numbers of iterations of fixpoint computation depending on the structure of the models and the type of the fixpoint computed. For example, RIS required more iterations than SIS for Prop₁, whereas it converged in just two steps (vs. the number of iterations proportional to the model size for SIS) for Prop₂ over the same model.

These experiments suggest that using the more precise RIS semantics improves the overall performance of model checking, making it a viable alternative to SIS in practice.

	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.	I ₁	I ₂	Analysis (sec.)	ABSREDU (sec.)	Iter.	I ₁	I ₂
Prop ₁	100	2.20	301			3.60	0.74	401		
	200	15.36	601	T	U	27.77	6.45	801	T	T
	250	28.92	751			55.19	13.40	1001		
Prop ₂	100	3.60	203			0.03	$< 10^{-4}$	2		
	200	27.16	403	T	U	0.12	$< 10^{-4}$	2	T	T
	250	54.62	503			0.19	$< 10^{-4}$	2		
Prop ₃	100	33.96	400			21.24	4.5	400		
	200	395.24	800	F	F	258.72	42.44	800	F	F
	250	1108.67	1000			546.88	101.20	1000		

Fig. 3. Experimental results (T, F and U denote *True*, *False* and *Unknown*, respectively).

7 Related Work and Conclusion

Godefroid and Jagadeesan [8], and Gurfinkel and Chechik [9] proved that the models in the KMTS family have the same expressive power and are equally precise for SIS. Dams and Namjoshi [5] showed that the three families considered in this paper are subsumed by tree automata. We complete 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 [5]) with one another.

We did not consider Hyper TSs (HTSs) [20] which allow for both *must* and *may* hyper-transitions. As pointed out in [20], *may* hyper-transitions can be eliminated by increasing the abstract statespace, making HTSs exactly as expressive as GKMTSs.

Both GKMTSs and MixTSs support monotonic abstraction refinement under SIS [4, 11, 19]. The same result holds under RIS, because for any two partial model M and M' over the same abstract statespace, if M' is less precise than M under SIS, i.e., $\forall \varphi \in L_\mu \cdot \|\varphi\|_c^{M'} \preceq_a \|\varphi\|_c^M$, M' is also less precise than M under RIS.

Our reduction operator RED is an instance of normalization from Abstract Interpretation [3] that is sometimes 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 [18].

Since RIS is inductive, its precision lies between SIS and thorough semantics. Thus, existing results comparing SIS and thorough semantics, e.g., [7, 9, 17], apply to RIS as well. We leave open the question of whether the additional precision enjoyed by RIS can be used to improve the above results.

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

We further introduced a new inductive semantics, RIS, for partial models. RIS is more precise than SIS, making MixTSs and GKMTSs equivalent w.r.t. model-checking.

We also described a symbolic implementation of model-checking w.r.t. 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. Our experiments suggest that RIS should be a good alternative to SIS for predicate abstraction-based model-checkers. We leave further investigations along this research direction to future work.

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

References

1. G. Bruns and P. Godefroid. “Generalized Model Checking: Reasoning about Partial State Spaces”. In *CONCUR*, vol. 1877 of *LNCS*, pp. 168–182, 2000.
2. M. Chechik, B. Devereux, S. Easterbrook, and A. Gurfinkel. “Multi-Valued Symbolic Model-Checking”. *ACM TOSEM*, 12(4):1–38, 2003.
3. P. Cousot and R. Cousot. “Abstract Interpretation Frameworks”. *J. of Logic and Computation*, 2(4):511–547, 1992.
4. D. Dams, R. Gerth, and O. Grumberg. “Abstract Interpretation of Reactive Systems”. *ACM TOPLAS*, 2(19):253–291, 1997.
5. D. Dams and K. S. Namjoshi. “Automata as Abstractions”. In *VMCAI*, vol. 3385 of *LNCS*, pp. 216–232, 2005.
6. L. de Alfaro, P. Godefroid, and R. Jagadeesan. “Three-Valued Abstractions of Games: Uncertainty, but with Precision”. In *LICS*, pp. 170–179, 2004.
7. P. Godefroid and M. Huth. “Model Checking v.s. Generalized Model Checking: Semantic Minimizations for Temporal Logics”. In *LICS*, pp. 158–167, 2005.
8. P. Godefroid and R. Jagadeesan. “On the Expressiveness of 3-Valued Models”. In *VMCAI*, vol. 2575 of *LNCS*, pp. 206–222, 2003.
9. A. Gurfinkel and M. Chechik. “How Thorough is Thorough Enough”. In *CHARME*, vol. 3725 of *LNCS*, pp. 65–80, 2005.
10. A. Gurfinkel and M. Chechik. “Why Waste a Perfectly Good Abstraction?”. In *TACAS*, vol. 3920 of *LNCS*, pp. 212–226, 2006.
11. A. Gurfinkel, O. Wei, and M. Chechik. “Systematic Construction of Abstractions for Model-Checking”. In *VMCAI*, vol. 3855 of *LNCS*, pp. 381–397, 2006.
12. 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.
13. 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.
14. D. Kozen. “Results on the Propositional μ -calculus”. *Theoretical Computer Science*, 27:334–354, 1983.
15. K. Larsen and B. Thomsen. “A Modal Process Logic”. In *LICS*, pp. 203–210, 1988.
16. P. Larsen. “The Expressive Power of Implicit Specifications”. In *ICALP*, vol. 510 of *LNCS*, pp. 204–216, 1991.
17. S. Nejati, M. Gheorghiu, and M. Chechik. “Thorough Checking Revisited”. In *FMCAD*, pp. 106–116, 2006.
18. T. W. Reps, A. Loginov, and S. Sagiv. “Semantic Minimization of 3-Valued Propositional Formulae”. In *LICS*, pp. 40–54, 2002.
19. S. Shoham and O. Grumberg. “Monotonic Abstraction-Refinement for CTL”. In *TACAS*, vol. 2988 of *LNCS*, pp. 546–560, 2004.
20. S. Shoham and O. Grumberg. “3-Valued Abstraction: More Precision at Less Cost”. In *LICS*, pp. 399–410, 2006.
21. F. Somenzi. “CUDD: CU Decision Diagram Package Release”, 2001.