On The Expressiveness of 3-Valued Models

Patrice Godefroid[1] and Radha Jagadeesan*[2]

[1] Bell Laboratories, Lucent Technologies, god@bell-labs.com
[2] DePaul University, rjagadeesan@cs.depaul.edu

Abstract. Three-valued models and logics have been recently advocated as being more suitable to reason about automatically-generated abstractions of reactive systems than traditional "2-valued" models such as standard Kripke structures or Labeled Transition Systems. Indeed, abstractions specified in 3-valued models are able to distinguish properties that are true, false and unknown of the concrete system, and hence their analysis can yield correctness proofs and counter-examples that can be both guaranteed to be sound. In this paper, we study several 3-valued modeling formalisms proposed in the literature and show that they have the same expressiveness, in the sense that any model specified in any of these formalisms can be translated into a model specified in any other. We also show that the complexity of the model checking and generalized model checking problems does not change from one formalism to the other.

1 Introduction

Program verification via automatic abstraction and model checking is currently an active area of research (e.g., [BPR01,DDP99,HJMS02]). This approach consists of automatically extracting a model out of a program by statically analyzing its code, and then of analyzing this model using model-checking techniques. If the model-checking results are inconclusive due to too much information being lost in the current abstraction, the model can then be automatically refined into a more detailed one provided the abstraction process can be parameterized and adjusted dynamically guided by the verification needs, as is the case with predicate abstraction [GS97] for instance. Current frameworks and tools that follow the above paradigm (e.g., [BR01,HJMS02]) typically use traditional formalisms (such as Kripke structures or Labeled Transition Systems) for representing models, while the soundness of their analysis is based on using a simulation relation for relating the abstract model to the concrete program being analyzed. Two well-known drawbacks of these design choices are that the scope of verification is then limited to universal properties, and that counter-examples are generally unsound since abstraction usually introduces unrealistic behaviors that may yield spurious errors being reported when analyzing the model.

Recently [GJ02,GHJ01,HJS01,BG00,BG99], it was shown how automatic abstraction can be performed to verify arbitrary formulas of the propositional μ-calculus [Koz83] in such a way that both correctness proofs and counter-examples

* Supported in part by NSF grants CCR 99010171 and CCR 02030716.

are guaranteed to be sound. The key to make this possible is to represent abstract systems using richer models that distinguish properties that are true, false and unknown of the concrete system. Examples of such richer modeling formalisms are partial Kripke structures [BG99] and Modal Transition Systems [LT88]. Reasoning about such systems requires 3-valued temporal logics [BG99], i.e., temporal logics whose formulas may evaluate to *true*, *false* or $\perp$ ("unknown") on a given model. Then, by using an automatic abstraction process that generates by construction an abstract model which is less complete than the concrete system with respect to a completeness preorder logically characterized by 3-valued temporal logic, every temporal property ϕ that evaluates to *true* (resp. *false*) on the abstract model automatically holds (resp. does not hold) of the concrete system, hence guaranteeing soundness of both proofs and counter-examples. In case ϕ evaluates to $\perp$ on the model, *generalized model checking* [BG00,GJ02] can be used to check whether there exist concretizations of the abstract model that satisfies ϕ or violates ϕ; if a negative answer is obtained in either one of these two tests, ϕ does not hold (resp. holds) of the concrete system. Otherwise, the analysis is still inconclusive and a more complete (i.e., less abstract) model is then necessary to provide a definite answer concerning this property of the concrete system. This approach is applicable to check arbitrary formulas of the propositional μ-calculus (thus including negation and arbitrarily nested path quantifiers), not just universal properties as with a traditional "conservative" abstraction that merely simulates the concrete system. It is shown in [GHJ01] that building a 3-valued abstraction can be done using existing abstraction techniques at the same computational cost as building a conservative abstraction. See [GJ02] for examples of programs and properties that cannot be verified using traditional conservation abstraction.

In this paper, we study and compare several 3-valued modeling formalisms that have been proposed in the literature, namely partial Kripke structures [BG99], Modal Transition Systems [LT88], and Kripke Modal Transition Systems [HJS01]. We define procedures for translating models specified in any of these formalisms into models specified in any other. As a corollary of these translations, we show that these modeling formalisms have the same expressiveness. We also study properties of these translations, and show that the model checking and generalized model checking problems can be both reduced from one formalism to any other. We then conclude by discussing applications and consequences of these results, as well as why these formalisms complement each other.

2 Background: 3-Valued Modeling Formalisms

In this section we recall the definitions of several 3-valued models used in the literature, as well as of temporal logics interpreted over them.

2.1 Partial Kripke Structures

The first 3-valued model we review is the *partial Kripke structure* as defined in [BG99,BG00]. A partial Kripke structure is simply a Kripke structure whose atomic propositions can have a third truth value $\perp$, which means "unknown whether true or false". Formally, we have the following.

Definition 1. *A* partial Kripke structure *(PKS) M is a tuple $(S, P, \rightarrow, L)$, where S is a set of* states, *P is a set of* atomic propositions, *$\rightarrow \subseteq S \times S$ is a transition relation on S, and $L : S \times P \rightarrow \{true, \perp, false\}$ is an* interpretation *that associates a truth value in $\{true, \perp, false\}$ with each atomic proposition in P for each state in S.*

A standard Kripke structure is a special case of partial Kripke structure. We sometimes refer to standard Kripke structures as *complete* Kripke structures to emphasize that no propositions within them take value $\perp$. In what follows, we often write $s \rightarrow s'$ as shorthand for $(s, s') \in \rightarrow$.

Propositional operators are interpreted on PKSs using Kleene's strong 3-valued propositional logic [Kle87]. Conjunction $\wedge$ in this logic is defined as the function that returns *true* if both of its arguments are *true*, *false* if either argument is *false*, and $\perp$ otherwise. We define negation $\neg$ using the function 'comp' that maps *true* to *false*, *false* to *true*, and $\perp$ to $\perp$. Disjunction $\vee$ is defined as usual using De Morgan's laws: $p \vee q = \neg(\neg p \wedge \neg q)$. Note that these functions give the usual meaning of the propositional operators when applied to values *true* and *false*.

Propositional modal logic (PML) is propositional logic extended with the modal operator AX (which is read "for all immediate successors"). Formulas of PML have the following abstract syntax: $\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid AX\phi$, where p ranges over P. The following 3-valued semantics generalizes the traditional 2-valued semantics for PML.

Definition 2. *The value of a formula ϕ of 3-valued PML in a state s of a PKS $M = (S, P, \rightarrow, L)$, written $[(M, s) \models \phi]$, is defined inductively as follows:*

$$\begin{aligned}
[(M,s) \models p] &= L(s,p) \\
[(M,s) \models \neg\phi] &= comp([(M,s) \models \phi]) \\
[(M,s) \models \phi_1 \wedge \phi_2] &= [(M,s) \models \phi_1] \wedge [(M,s) \models \phi_2] \\
[(M,s) \models AX\phi] &= \bigwedge_{s \rightarrow s'} [(M,s') \models \phi].
\end{aligned}$$

This 3-valued logic can be used to define a preorder on KMTSs that reflects their degree of completeness. Let $\leq$ be the *information ordering* on truth values, in which $\perp \leq true$, $\perp \leq false$, $x \leq x$ (for all $x \in \{true, \perp, false\}$), and $x \not\leq y$ otherwise.

Definition 3. *Let $M_1 = (S_1, P, \rightarrow_1, L_1)$ and $M_2 = (S_2, P, \rightarrow_2, L_2)$ be partial Kripke structures. The* completeness preorder *$\preceq$ is the greatest relation $\mathcal{B} \subseteq S_1 \times S_2$ such that $(s_1, s_2) \in \mathcal{B}$ implies the following:*

- $\forall p \in P : L_1(s_1, p) \leq L_2(s_2, p)$,
- *if $s_1 \rightarrow s_1'$ then there is some $s_2' \in S_2$ such that $s_2 \rightarrow s_2'$ and $(s_1', s_2') \in \mathcal{B}$,*
- *if $s_2 \rightarrow s_2'$ then there is some $s_1' \in S_1$ such that $s_1 \rightarrow s_1'$ and $(s_1', s_2') \in \mathcal{B}$.*

Intuitively, $s_1 \preceq s_2$ means that s_1 and s_2 are "nearly bisimilar" except that the atomic propositions in state s_1 may be less defined than in state s_2. The following theorem states that 3-valued PML logically characterizes the completeness preorder on PKSs [BG99].

Theorem 1. *Let $M_1 = (S_1, P, \rightarrow_1, L_1)$ and $M_2 = (S_2, P, \rightarrow_2, L_2)$ be partial Kripke structures such that $s_1 \in S_1$ and $s_2 \in S_2$, and let Φ be the set of all formulas of 3-valued PML. Then*

$$s_1 \preceq s_2 \textit{ iff } (\forall \phi \in \Phi : [(M_1, s_1) \models \phi] \leq [(M_2, s_2) \models \phi]).$$

In other words, partial Kripke structures that are "more complete" with respect to $\preceq$ have more definite properties with respect to $\leq$, i.e., have more properties that are either *true* or *false*. Moreover, any formula ϕ of 3-valued PML that evaluates to *true* or *false* on a partial Kripke structure has the same truth value when evaluated on any more complete structure. This result also holds for PML extended with fixpoint operators [BG00], also known as the propositional μ-calculus [Koz83].

2.2 Modal Transition Systems

Modal Transitions Systems are a generalization of Labeled Transition Systems introduced in [LT88,Lar89].

Definition 4. *A* Modal Transition System *(MTS) M is a tuple $(S, \Sigma, \xrightarrow{must}, \xrightarrow{may})$, where S is a set of* states*, Σ is a set of action symbols, $\xrightarrow{must} \subseteq S \times \Sigma \times S$ and $\xrightarrow{may} \subseteq S \times \Sigma \times S$ are transition relations such that $\xrightarrow{must} \subseteq \xrightarrow{may}$.*

An MTS is thus a Labeled Transition System (LTS) with two types of transitions, *must* and *may* transitions, with the additional constraint that every *must*-transition is also a *may*-transition. Reasoning about the existence of transitions of MTSs can be viewed as reasoning with a three-valued logic: transitions that are necessarily true are *true*, transitions that are possibly true but not necessarily true are $\perp$, and transitions that are not possibly true are *false*. Formally, we can define a 3-valued PML on MTSs, let us denote it $\mathrm{PML}^{\mathtt{Act}}$, whose abstract syntax is defined recursively as follows: $\phi ::= \mathtt{tt} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid (\forall a)\phi$, where a ranges over Σ. The semantics of these operators can be defined as follows.

Definition 5. *The value of a formula ϕ of 3-valued $PML^{\mathtt{Act}}$ in a state s of an MTS $M = (S, \Sigma, \xrightarrow{must}, \xrightarrow{may})$, written $[(M, s) \models \phi]$, is defined inductively as follows:*

$$
\begin{aligned}
[(M,s) \models \mathtt{tt}] &= \textit{true} \\
[(M,s) \models \neg\phi] &= \textit{comp}([(M,s) \models \phi]) \\
[(M,s) \models \phi_1 \wedge \phi_2] &= [(M,s) \models \phi_1] \wedge [(M,s) \models \phi_2] \\
[(M,s) \models (\forall a)\phi] &= \begin{cases} \textit{true} \ \textit{if} \ \forall (s,a,s') \in \xrightarrow{may} : [(M,s') \models \phi] = \textit{true} \\ \textit{false} \ \textit{if} \ \exists (s,a,s') \in \xrightarrow{must} : [(M,s') \models \phi] = \textit{false} \\ \perp \quad \textit{otherwise} \end{cases}
\end{aligned}
$$

The above semantics is equivalent to the one that appeared in [HJS01,GHJ01]. We use this alternate form to facilitate the comparison with other models in later sections.

The degree of completeness of MTSs can be measured by the following completeness preorder (whose inverse is called "refinement preorder" [LT88]).

Definition 6. *Let $M_1 = (S_1, \Sigma, \xrightarrow{must}_1, \xrightarrow{may}_1)$ and $M_2 = (S_2, \Sigma, \xrightarrow{must}_2, \xrightarrow{may}_2)$ be MTSs. The* completeness preorder $\preceq$ *is the greatest relation $\mathcal{B} \subseteq S_1 \times S_2$ such that $(s_1, s_2) \in \mathcal{B}$ implies the following:*

- *if $(s_1, a, s_1') \in \xrightarrow{must}_1$, there is some $s_2' \in S_2$ such that $(s_2, a, s_2') \in \xrightarrow{must}_2$ and $(s_1', s_2') \in \mathcal{B}$,*
- *if $(s_2, a, s_2') \in \xrightarrow{may}_2$, there is some $s_1' \in S_1$ such that $(s_1, a, s_1') \in \xrightarrow{may}_1$ and $(s_1', s_2') \in \mathcal{B}$.*

Again, $s_1 \preceq s_2$ means that s_1 is more abstract (i.e., less complete) than s_2. This definition allows to abstract a system M_2 by a more abstract system M_1 by letting *must*-transitions of M_2 become *may*-transitions in M_1, but all *may*-transitions of M_2 must be preserved in M_1. It can be shown that this completeness preorder is logically characterized by 3-valued PML [HJS01,GHJ01].

Theorem 2. *Let $M_1 = (S_1, \Sigma, \xrightarrow{must}_1, \xrightarrow{may}_1)$ and $M_2 = (S_2, \Sigma, \xrightarrow{must}_2, \xrightarrow{may}_2)$ be MTSs such that $s_1 \in S_1$ and $s_2 \in S_2$, and let Φ be the set of all formulas of 3-valued PML$^{\mathtt{Act}}$. Then,*

$$s_1 \preceq s_2 \text{ iff } (\forall \phi \in \Phi : [(M_1, s_1) \models \phi] \leq [(M_2, s_2) \models \phi]).$$

2.3 Kripke Modal Transition Systems

A third model we consider here is (a simplified version of) the Kripke Modal Transition Systems introduced in [HJS01]. This model combines features of both PKSs and MTSs, although it does not increase their expressiveness as will be shown later in this paper. Precisely, we define a Kripke Modal Transition System as follows [GJ02].

Definition 7. *A* Kripke Modal Transition System *(KMTS) M is a tuple $(S, P, \xrightarrow{must}, \xrightarrow{may}, L)$, where S is a nonempty finite set of states, P is a finite set of atomic propositions, $\xrightarrow{may} \subseteq S \times S$ and $\xrightarrow{must} \subseteq S \times S$ are transition relations such that $\xrightarrow{must} \subseteq \xrightarrow{may}$, and $L : S \times P \to \{true, \perp, false\}$ is an* interpretation *that associates a truth value in $\{true, \perp, false\}$ with each atomic proposition in P for each state in S.*

Clearly, KMTSs generalize PKSs since a PKS is a KMTS where $\xrightarrow{must} = \xrightarrow{may}$. Reasoning about KMTSs can be done using the same 3-valued PML defined as for PKSs provided that we modify the semantics of the AX operator to account for the presence of *must* and *may* transitions as follows:

Definition 8. *The value of a formula ϕ of 3-valued PML in a state s of a KMTS $M = (S, P, \xrightarrow{must}, \xrightarrow{may}, L)$, written $[(M,s) \models \phi]$, is defined inductively as follows:*

$$
\begin{aligned}
[(M,s) \models p] &= L(s,p) \\
[(M,s) \models \neg\phi] &= comp([(M,s) \models \phi]) \\
[(M,s) \models \phi_1 \wedge \phi_2] &= [(M,s) \models \phi_1] \wedge [(M,s) \models \phi_2] \\
[(M,s) \models AX\phi] &= \begin{cases} true \ \ if\ \forall s' : s \xrightarrow{may} s' \Rightarrow [(M,s') \models \phi] = true \\ false\ if\ \exists s' : s \xrightarrow{must} s' \wedge [(M,s') \models \phi] = false \\ \perp \quad\ otherwise \end{cases}
\end{aligned}
$$

The completeness preorder on KMTSs is then the following.

Definition 9. *Let $M_1 = (S_1, P, \xrightarrow{must}_1, \xrightarrow{may}_1, L_1)$ and $M_2 = (S_2, P, \xrightarrow{must}_2, \xrightarrow{may}_2, L_2)$ be KMTSs. The* completeness preorder *$\preceq$ is the greatest relation $\mathcal{B} \subseteq S_1 \times S_2$ such that $(s_1, s_2) \in \mathcal{B}$ implies the following:*

- *$\forall p \in P : L_1(s_1,p) \leq L_2(s_2,p)$,*
- *if $s_1 \xrightarrow{must}_1 s_1'$, there is some $s_2' \in S_2$ such that $s_2 \xrightarrow{must}_2 s_2'$ and $(s_1', s_2') \in \mathcal{B}$,*
- *if $s_2 \xrightarrow{may}_2 s_2'$, there is some $s_1' \in S_1$ such that $s_1 \xrightarrow{may}_1 s_1'$ and $(s_1', s_2') \in \mathcal{B}$.*

Thus, the completeness preorder on KMTSs also generalizes the completeness preorder on PKSs. It can be shown that this completeness preorder is logically characterized by 3-valued PML as defined above [GJ02].

Theorem 3. *Let $M_1 = (S_1, P, \xrightarrow{must}_1, \xrightarrow{may}_1), L_1$ and $M_2 = (S_2, P, \xrightarrow{must}_2, \xrightarrow{may}_2, L_2)$ be KMTSs such that $s_1 \in S_1$ and $s_2 \in S_2$, and let Φ be the set of all formulas of 3-valued PML. Then,*

$$s_1 \preceq s_2 \ \textit{iff}\ (\forall \phi \in \Phi : [(M_1,s_1) \models \phi] \leq [(M_2,s_2) \models \phi]).$$

2.4 Model Checking and Generalized Model Checking

Computing the value of $[(M,s) \models \phi]$ is referred to as the *model checking* problem in what follows. As argued in [BG00], the semantics of $[(M,s) \models \phi]$ returns $\perp$ more often that it should. For instance, consider a KMTS M with two states s_0 and s_1 such that $p = q = true$ in s_0 and $p = q = false$ in s_1, and with a *may*-transition from s_0 to s_1. The formula $AXp \wedge \neg AXq$ (which is neither a tautology nor unsatisfiable) is $\perp$ at s_0, yet in all complete structures more complete than (M', s_0) the formula is *false*. This observation is used in [BG00] to define an alternative 3-valued semantics for modal logics called the *thorough* semantics since it does more than the other semantics to discover whether enough information is present in a model to give a definite answer. Let the *completions* $\mathcal{C}(M,s)$ of a state s of a model M be the set of all states s' of complete models M' such that $s \preceq s'$.

Definition 10. *Let ϕ be a formula of any two-valued logic for which a satisfaction relation $\models$ is defined on complete models. The truth value of ϕ in a state s of a model M under the* thorough *interpretation, written $[(M,s) \models \phi]_t$, is defined as follows:*

$$[(M,s) \models \phi]_t = \begin{cases} \mathit{true} \;\; \mathit{if}\ (M',s') \models \phi \ \mathit{for\ all}\ (M',s') \ \mathit{in}\ \mathcal{C}(M,s) \\ \mathit{false}\ \mathit{if}\ (M',s') \not\models \phi \ \mathit{for\ all}\ (M',s') \ \mathit{in}\ \mathcal{C}(M,s) \\ \perp \qquad \mathit{otherwise} \end{cases}$$

By definition, we always have $[(M,s) \models \phi] \leq [(M,s) \models \phi]_t$. In general, interpreting a formula according to the thorough three-valued semantics is equivalent to solving two instances of the generalized model-checking problem [BG00].

Definition 11 (Generalized Model-Checking Problem). *Given a state s of a model M and a formula ϕ of a (two-valued) temporal logic L, does there exist a state s' of a complete model M' such that $s \preceq s'$ and $(M',s') \models \phi$?*

In other words, generalized model checking means checking whether there exists a completion of an abstraction that satisfies a temporal logic formula. This problem is called generalized model checking since it generalizes both model checking and satisfiability checking: at one extreme, when M is completely unknown (i.e., unconstrained), all complete models are more complete than M and the problem reduces to the satisfiability problem; at the other extreme, when M is complete, only a single structure needs to be checked and the problem reduces to model checking. Model checking and generalized model checking are the two main types of analyses of 3-valued models we consider in what follows.

3 Translation from PKS to MTS

We start by showing how to translate PKSs to MTSs. This can simply be done as follows.

Definition 12. *For any PKS $M = (S,P,\rightarrow,L)$, we define an equivalent MTS $M' = (S', \Sigma, \xrightarrow{must}, \xrightarrow{may})$ such that*

- $S' = S \cup \{s_{new}\}$,
- $\Sigma = P \cup \{x\}$,
- $\xrightarrow{must} = \{(s,p,s_{new}) | L(s,p) = \mathit{true}\} \cup \{(s,x,s') | s \rightarrow s'\}$, *and*
- $\xrightarrow{may} = \{(s,p,s_{new}) | L(s,p) \in \{\mathit{true},\perp\}\} \cup \{(s,x,s') | s \rightarrow s'\}$.

Note that the previous definition does indeed generate an MTS since we always have $\xrightarrow{must} \subseteq \xrightarrow{may}$ by construction. Given an input PKS M of size $O(|\rightarrow| + |S| \cdot |P|)$, the size of the output MTS M' produced by Definition 12 is $O(|\xrightarrow{must}| + |\xrightarrow{may}|)$ $= O(|\rightarrow| + |S| \cdot |P|)$, i.e., linear in the size of the input. We can show that the above translation preserves the completeness preorder.

Theorem 4. *Given any two PKSs (M, s_m) and (N, s_n), let M' and N' denote the corresponding MTSs obtained by applying Definition 12. Then, we have*

$$(M, s_m) \preceq (N, s_n) \text{ iff } (M', s_m) \preceq (N', s_n).$$

Proof. (Sketch)[1] Assuming $(M, s_m) \preceq (N, s_n)$, we define a binary relation R on the states of M' and N' such that sRt if $s \preceq t$ or $s = t = s_{new}$; then we show that R satisfies the conditions of Definition 6. Conversely, assuming $(M', s_m) \preceq (N', s_n)$, we define a binary relation R on the states of M and N such that sRt if $s \preceq t$ with $s, t \neq s_{new}$; it can be shown that R satisfies the conditions of Definition 3.

We can also prove the following.

Lemma 1. *Let M be a PKS and M' be the corresponding MTS obtained by applying Definition 12. If there exists an MTS Q such that $(M', s_m) \preceq (Q, s_q)$, then there exists a PKS N such that $(N', s_n) \preceq (Q, s_q)$ and $(Q, s_q) \preceq (N', s_n)$, where N' denotes the translation of N by Definition 12.*

Proof. (Sketch) Let $Q = (S, \Sigma, \xrightarrow{must}, \xrightarrow{may})$, let s_{new} denote the state of M' defined as in Definition 12, and let $S_{new} = \{s \in S | s_{new} \preceq s\}$. Then we define a PKS $N = (S', P, \rightarrow, L)$ from Q as follows: $S' = S \setminus S_{new}$, $P = \Sigma \setminus \{x\}$, $\rightarrow = \{(s, s') \in (S \setminus S_{new}) \times (S \setminus S_{new}) | (s, x, s') \in \xrightarrow{must}\}$, and $L(s, p) = true$ if $\exists s' \in S_{new} : (s, p, s') \in \xrightarrow{must}$, or $L(s, p) = false$ if $\forall s' \in S_{new} : (s, p, s') \not\in \xrightarrow{may}$, or $L(s, p) = \perp$ otherwise. It is then easy to show that $(N', s_n) \preceq (Q, s_q)$ and $(Q, s_q) \preceq (N', s_n)$.

PML temporal-logic formulas defined on PKSs can be translated into $\mathrm{PML}^{\mathtt{Act}}$ formulas defined on MTSs as follows.

Definition 13. *Given any PML formula ϕ defined on PKSs with a set P of atomic propositions, we define an equivalent $\mathrm{PML}^{\mathtt{Act}}$ formula $T(\phi)$ defined on MTSs with action alphabet $\Sigma = P \cup \{x\}$ by applying recursively the following rewrite rules:*

- *for all $p \in P$, $T(p) = (\exists p)\mathtt{tt} = \neg(\forall p)\neg\mathtt{tt}$,*
- *$T(\neg\phi) = \neg T(\phi)$,*
- *$T(\phi_1 \wedge \phi_2) = T(\phi_1) \wedge T(\phi_2)$,*
- *$T(AX\phi) = (\forall x)T(\phi)$.*

The next theorem states that the model and formula translations of Definitions 12 and 13 can be used to reduce (in linear time and logarithmic space) the model checking and generalized model checking problems from PKSs to MTSs.

Theorem 5. *Given any PKS M and formula ϕ defined on PKSs, let M' denote the MTS obtained by applying Definition 12 and let $T(\phi)$ denote the translated formula obtained by applying Definition 13. Then, we have the following:*

1. $[(M, s) \models \phi] = [(M', s) \models T(\phi)]$
2. $[(M, s) \models \phi]_t = [(M', s) \models T(\phi)]_t$

[1] Complete proofs are omitted in this extended abstract due to space limitations.

Proof. (Sketch) (1) is proved by induction on the length of ϕ. (2) is proved as follows: if $[(M, s) \models \phi]_t = \mathit{true}$, i.e., if there exists $s' \in N$ such that $s \preceq s'$ and $[(N, s') \models \phi] = \mathit{true}$, then $(M', s) \preceq (N', s')$ by Theorem 4 and we have $[(N', s') \models T(\phi)] = \mathit{true}$ by (1), hence $[(M', s) \models T(\phi)]_t = \mathit{true}$; conversely, if $[(M', s) \models T(\phi)]_t = \mathit{true}$, i.e., there exists $s' \in Q$ such that $(M', s) \preceq (Q, s')$ and $[(Q, s') \models T(\phi)] = \mathit{true}$, then there exists N such that $N' \preceq Q$ and $Q \preceq N'$ by Lemma 1 and we also have $(M, s) \preceq (N, s')$ by Theorem 4 and $[(N, s') \models \phi] = \mathit{true}$ by (1), hence $[(M, s) \models \phi]_t = \mathit{true}$. The case where $[(M, s) \models \phi]_t = \mathit{false}$ is similar.

4 Translation from MTS to KMTS

The second translation we consider is from MTSs to KMTSs. We present in this section results of the same nature as those of the previous section. A translation from MTSs to KMTSs that preserves the completeness preorder can be defined as follows.

Definition 14. *For any MTS $M = (S, \Sigma, \xrightarrow{must}, \xrightarrow{may})$, we define an equivalent KMTS $M' = (S', P, \xrightarrow{must'}, \xrightarrow{may'}, L)$ such that*

- $S' = S \times \Sigma$,
- $P = \Sigma$,
- $\xrightarrow{must'} = \{((s, a), (s', a')) | (s, a', s') \in \xrightarrow{must}\}$,
- $\xrightarrow{may'} = \{((s, a), (s', a')) | (s, a', s') \in \xrightarrow{may}\}$, *and*
- $\forall (s, a) \in S' : \forall p \in P : L((s, a), p) = \mathit{true}$ *if* $p = a$ *or* $L((s, a), p) = \mathit{false}$ *otherwise.*

The previous definition thus simply replaces transition labels by state labels. Given an MTS with $|S|$ states and an action set Σ, the number of states in the translated KMTS M' defined by Definition 14 is at most $|S| \cdot |\Sigma|$. In other words, each state of M can be copied at most $|\Sigma|$ times in M'. For a fixed action set Σ, the number of states and transitions in M' is nevertheless linear in the number of states and transitions, respectively, in M.

Note that the above translation only uses atomic propositions $p \in P$ whose truth value is always defined as either *true* or *false*. However, it does not seem possible to improve this translation by using the third unused value $\perp$. Indeed, in the particular case of a MTS which is completely defined, i.e., such that $\xrightarrow{must} = \xrightarrow{may}$, the above translation then becomes the traditional translation from LTSs to Kripke structures whose states are pairs of states and actions of the LTS [MSS99], the latter translation being also similar to the classic translation from Mealy automata to Moore automata [HU79].

The translation above preserves the completeness preorder.

Theorem 6. *Given any two MTSs (M, s_m) and (N, s_n), let M' and N' denote the corresponding KMTSs obtained by applying Definition 14. Then, for all $a \in \Sigma$, we have*

$$(M, s_m) \preceq (N, s_n) \text{ iff } (M', (s_m, a)) \preceq (N', (s_n, a)).$$

Proof. (Sketch) Given $(M, s_m) \preceq (N, s_n)$, we define a binary relation R on the states of M' and N' such that $(s,a)R(t,b)$ if $s \preceq t$ and $a = b$; then we show that R satisfies the conditions of Definition 9. Conversely, assuming $(M', (s_m, a)) \preceq (N', (s_n, a))$, we define a binary relation R on the states of M and N such that sRt if $(s,a) \preceq (t,a)$ with $s, t \in S$; R can be shown to satisfy the conditions of Definition 6.

We also have the following.

Lemma 2. *Let M be a MTS and M' be the corresponding KMTS obtained by applying Definition 14. If there exists a KMTS Q such that $(M', s_m) \preceq (Q, s_q)$, then there exists a MTS N such that $N' = Q$, where N' denotes the translation of N by Definition 14.*

Proof. (Sketch) Let $Q = (S, P, \xrightarrow{must}, \xrightarrow{may}, L)$. Since $(M', s_m) \preceq (Q, s_q)$, we know that, in every state $s \in S$, there is exactly one proposition p in P that is *true* in s while all other propositions in P are then *false* in s. We define an MTS $N = (S, P, \xrightarrow{must'}, \xrightarrow{may'})$ from the KMTS Q as follows: $\xrightarrow{must'} = \{(s, p, s') | (s, s') \in \xrightarrow{must}$ and $L(s', p) = true\}$, and $\xrightarrow{may'} = \{(s, a, s') | (s, s') \in \xrightarrow{may}$ and $L(s', p) = true\}$. Clearly, $N' = Q$.

PML$^{\mathsf{Act}}$ formulas can be translated into PML formulas as follows.

Definition 15. *Given any formula ϕ defined on MTSs with an action set Σ, we define an equivalent formula $T(\phi)$ defined on KMTSs with a set $P' = \Sigma$ of atomic propositions by applying recursively the following rewrite rules:*

- $T(\mathtt{tt}) = true$,
- $T(\neg\phi) = \neg T(\phi)$,
- $T(\phi_1 \wedge \phi_2) = T(\phi_1) \wedge T(\phi_2)$,
- $T((\forall a)\phi) = AX((\neg a) \vee T(\phi))$.

The right term of the last rule is thus equivalent to $AX(a \Rightarrow T(\phi))$ where $\Rightarrow$ denotes logical implication. The correctness of this formula translation is again defined by showing a reduction of the model checking and generalized model checking problems on MTSs to KMTSs.

Theorem 7. *Given any MTS M and formula ϕ defined on MTSs, let M' denote the KMTS obtained by applying Definition 14 and let $T(\phi)$ denote the translated formula obtained by applying Definition 15. Then, for any $a \in \Sigma$, we have the following:*

1. $[(M, s) \models \phi] = [(M', (s, a)) \models T(\phi)]$
2. $[(M, s) \models \phi]_t = [(M', (s, a)) \models T(\phi)]_t$

Proof. Similar to the proof of Theorem 5.

5 Translation from KMTS to PKS

The third and last translation we discuss is from KMTSs to PKSs. This translation is more elaborate than the two previous ones. We start by showing how to translate a KMTS to a PKS with the goal of reducing model checking and generalized model checking from KMTSs to PKSs.

Definition 16. *For any KMTS $M = (S, P, \xrightarrow{must}, \xrightarrow{may}, L)$, we define an equivalent PKS $M' = (S', P', \rightarrow', L')$ such that*

- $S' = S \times \{must, may\}$,
- $P' = P \cup \{p_{must}\}$,
- $\rightarrow' = \{((s,x),(s',x')) | (s,s') \in \xrightarrow{may}, x' = must$ *if* $(s,s') \in \xrightarrow{must}$ *or* $x' = may$ *otherwise*$\}$,
- $\forall (s,x) \in S' : \forall p \in P : L'((s,x),p) = L(s,p)$, *and* $L'((s,x),p_{must}) = true$ *if* $x = must$ *or* $L'((s,x),p_{must}) = \perp$ *otherwise.*

Two examples of KMTSs M and N and their PKS translations M' and N' with the previous definition are shown in Figure 1. These KMTSs have a single atomic proposition p whose value is defined in each state as indicated in the figure. The dotted transition in M is a *may*-transition that is not a *must*-transition.

We now define a translation of temporal-logic formulas on KMTSs to formulas on PKSs.

Definition 17. *Given any formula ϕ defined on KMTSs with a set P of atomic propositions, we define an equivalent formula $T(\phi)$ defined on PKSs with a set $P' = P \cup \{p_{must}\}$ of atomic propositions by applying recursively the following rewrite rules:*

- *for all* $p \in P$, $T(p) = p$,
- $T(\neg\phi) = \neg T(\phi)$,
- $T(\phi_1 \wedge \phi_2) = T(\phi_1) \wedge T(\phi_2)$,
- $T(AX\phi) = AX(p_{must} \Rightarrow T(\phi)) = AX((\neg p_{must}) \vee T(\phi))$.

The next theorem states that model checking and generalized model checking can be reduced (in linear time and logarithmic space) from KMTSs to PKSs.

Theorem 8. *Given any KMTS M and formula ϕ defined on KMTSs, let M' denote the PKS obtained by applying Definition 16 and let $T(\phi)$ denote the translated formula obtained by applying Definition 17. Then, we have the following:*

1. $[(M,s) \models \phi] = [(M',(s,must)) \models T(\phi)]$
2. $[(M,s) \models \phi]_t = [(M',(s,must)) \models T(\phi)]_t$

Proof. (Sketch) The proof of (1) is by induction on the length of the formula. To prove (2), we show that, if (N,t) is a complete KMTS such that $(M,s) \preceq (N,t)$, then there exists a PKS M'' such that $(M',(s,must)) \preceq (M'',r)$ and $[(M'',r) \models T(\phi)] = [(N',(t,must)) \models T(\phi)] = [(N,t) \models \phi]$. Next we show that, if (Q,r) is a complete PKS such that $(M',(s,must)) \preceq (Q,r)$, then there exists a complete KMTS (N,t) such that the translation N' of N by Definition 16 is equal to the part of Q containing all the states where $p_{must} = true$; we finally show that $(M,s) \preceq (N,t)$ and that $[(N,t) \models \phi] = [(N',(t',must)) \models T(\phi)] = [(Q,r) \models T(\phi)]$.

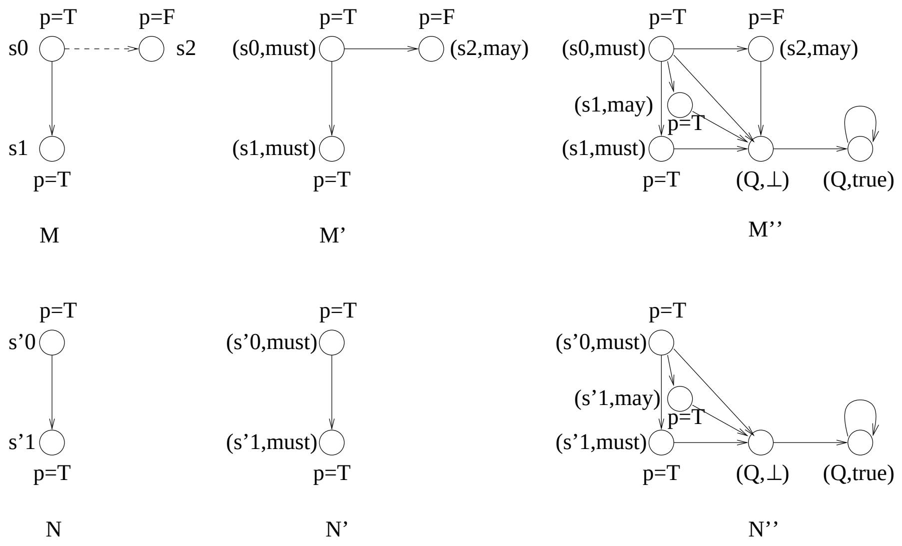

Fig. 1. Examples of KMTSs and of their translations. KMTSs M and N (left) are translated into PKSs M' and N' (middle) using Definition 16, and into PKSs M'' and N'' (right) using Definition 18.

In other words, the model and formula translations of Definitions 16 and 17 can be used to reduce (in linear time and logarithmic space) the model checking and generalized model checking problems from KMTSs to PKSs.

It is worth noticing that the translation from KMTS to PKS of Definition 16 does not preserve the completeness preorder: for any two KMTSs M and N and their respective PKS translations M' and N' obtained by applying the construction of Definition 16, $M \preceq N$ does not necessarily imply that $M' \preceq N'$. For instance, consider the KMTSs M and N of Figure 1. Clearly, $(M, s_0) \preceq (N, s'_0)$ but $(M', (s_0, must)) \not\preceq (N', (s'_0, must))$ since the second condition of Definition 3 is not satisfied for transition $((s_0, must), (s_2, may))$ of M' which cannot be matched in N'.

However, it is possible to design a translation from KMTS to PKS that does preserve the completeness preorder. Such a translation is now presented.

Definition 18. *For any KMTS $M = (S, P, \xrightarrow{must}, \xrightarrow{may}, L)$, we define an equivalent PKS $M' = (S', P', \rightarrow', L')$ such that*

- $S' = S \times \{must, may\} \cup \{(q, \bot), (q, true) | q \in 2^P\}$,
- $P' = P \cup \{p_{must}, p_{dummy}\}$,
- $\rightarrow' = \{((s,x),(s',may)) | (s,s') \in \xrightarrow{may}\} \cup \{((s,x),(s',must)) | (s,s') \in \xrightarrow{must}\} \cup \{((s,x),(q,\bot)) | (s,x) \in S \times \{must, may\}$ *and* $q \in 2^P\} \cup \{((q,\bot),(q',true)), ((q,true),(q',true)) | q, q' \in 2^P\}$,
- $\forall (s,x) \in S \times \{must, may\}, \forall p \in P : L'((s,x),p) = L(s,p)$, $L'((s,x),p_{dummy}) = \bot$, $L'((s,x),p_{must}) = true$ *if* $x = must$ *or* $L'((s,x),p_{must}) = \bot$ *otherwise;* $\forall q \in 2^P, x \in \{\bot, true\}$, $L'((q,x),p_{dummy}) = true$, $L'((q,x),p_{must}) = x$ *and* $\forall p \in P : L'((q,true),p) = true$ *if* $p \in q$ *or* $L'((q,true),p) = false$ *otherwise.*

Considering again the KMTSs M and N of Figure 1, the previous definition generates the PKSs M'' and N'' respectively, where (Q, x) denotes the whole cluster of states $\{(q, x) | q \in 2^P\}$. The reader can check that $(M'', (s_0, must)) \preceq (N'', (s'_0, must))$. Although the number of states of the form (q, x) is exponential in $|P|$, the number of states and transitions in the resulting PKS M' is linear in the number of states and transitions, respectively, of the input KMTS M. The correctness of the previous translation is defined as follows.

Theorem 9. *Given any two KMTSs (M, s_m) and (N, s_n), let M' and N' denote the corresponding PKSs obtained by applying Definition 18. Then, we have*

$$(M, s_m) \preceq (N, s_n) \text{ iff } (M', (s_m, must)) \preceq (N', (s_n, must)).$$

Proof. Omitted here due to space limitations.

An interesting corollary of the previous theorem is that checking whether $M' \preceq N'$ between two PKSs M' and N' is as hard as checking whether $M \preceq N$ between two KMTSs. Since the latter problem is itself in general at least as hard as checking for a simulation relation between M and N (since $\preceq$ reduces to a simulation relation in the case of KMTSs with no *must*-transitions), this implies that checking whether $M' \preceq N'$ between two PKSs can be as expensive as checking the existence of a simulation between them, which may not be obvious when looking at Definition 3.

Finally, note that this second more elaborate translation can also be used to reduce model checking and generalized model checking from KMTSs to PKSs provided another, more complicated, formula translation $T(\phi)$ (not presented here due to space constraints) is used.

6 Applications, Discussion and Other Related Work

The translations and theorems of the previous sections make it possible to derive several new results concerning the expressiveness and conciseness of the 3-valued models considered, as well as the complexity of model checking and generalized model checking for these models.

The first result we obtain is that *Partial Kripke Structures, Modal Transition Systems and Kripke Model Transition Systems are all equally expressive.* In other words, any PKS, MTS or KMTS can be translated into any other of these formalisms using the translations defined in Sections 3 to 5, or any combination of these translations. In what follows, let us denote by "3-valued formalism" either a PKS, MTS or KMTS.

Second, following Theorems 5 and 7 and 8, any model checking or generalized model checking problem defined on any 3-valued formalism can be reduced to a model checking or generalized model checking problem, respectively, on any other of these formalisms. These results hold not only for PML, but also for PML extended with fixpoint operators, i.e., the propositional μ-calculus [Koz83], and hence all of its fragments (such as LTL, CTL and CTL*). This extension to the

μ-calculus follows immediately from the facts that the completeness preorder is logically characterized by both PML and the μ-calculus (exactly as bisimulation is logically characterized by both PML and the μ-calculus in the 2-valued case), and that fixpoint operators are left unchanged when translating μ-calculus formulas from one formalism to another (formula translation essentially affects modal operators only).

Third, since the translations of the previous sections require only linear time and logarithmic space in the size of their inputs and can be applied to translate any 3-valued formalism into any other in both directions, both upper and lower complexity bounds for problems defined on any of these formalisms carry over to the other formalisms. In particular, the algorithms and complexity bounds for the model checking and generalized model checking problems obtained with PKSs in [BG00] and with KMTSs in [GJ02] apply equally to all three formalisms.

Specifically, these translations extend the scope of the results of [BG00] and formally prove that, for any 3-valued formalism, 3-valued model checking for any temporal logic L has the same time and space complexity, both in the size of the model and of the formula, as traditional 2-valued model checking for the logic L. This new result subsumes the direct model-checking procedures of [GHJ01] for MTSs and of [Hut02a] for KMTSs.

Concerning the complexity of generalized model checking in the size of the formula, a similar extension of the results of [BG00] is possible thanks to the translations: for any 3-valued formalism, in the case of a branching-time temporal logic L (such as CTL, CTL*, the mu-calculus, PML and also propositional logic), generalized model checking for L has the same complexity in the size of the formula as satisfiability checking for L, while in the case of linear-time temporal logic (LTL), generalized model checking is EXPTIME-complete in the size of the formula, i.e., harder than both satisfiability and model checking (which are both PSPACE-complete for LTL).

Regarding the complexity of generalized model checking in the size of the model, we note that the formula translations of Sections 3 to 5 preserve the co-Büchi recognizability of formulas: if a property represented by a temporal formula ϕ is recognizable by an automaton (on infinite trees or words, depending if ϕ is a branching or linear property, respectively) with a co-Büchi acceptance condition, then the property represented by the temporal property $T(\phi)$ is also recognizable by an automaton with a co-Büchi acceptance condition. Therefore, the complexity results of [GJ02] can be extended to cover any 3-valued formalism: the worst-case runtime complexity of generalized model checking for the temporal logics LTL and CTL can be quadratic in the size of the model for any 3-valued formalism, but generalized model checking can be solved in time linear in the size of the model in the case of persistence properties, i.e., properties recognizable by co-Büchi automata.

Notice that one could think at first sight that generalized model checking for KMTSs and MTSs is exponential in the size of the KMTS/MTS M since the number of possible completions of a set N of *may*-transitions from any given state is $2^{|N|}$. Instead, our linear translation from KMTSs to PKSs presented in the previous section proves that considering one-by-one the exponentially-many subsets of N is not necessary, and that generalized model checking can be done in polynomial time (linear or quadratic) in $|N|$.

Since PKSs, MTSs and KMTSs are all equally expressive, one can wonder why three different yet equivalent models have been proposed in the literature. Let us first observe that the equivalence of their expressiveness is not as straightforward as one might think (see the results of the previous sections). More importantly, these different formalisms are useful to facilitate abstraction of different aspects of reactive systems. PKSs generalize Kripke structures which model state changes of reactive systems, while MTSs generalize LTSs which model the external behavior of a system (i.e., sequences of actions the system can perform). Although equivalent themselves, Kripke structures and LTSs provide complementary views for reasoning about reactive systems, and PKS versus MTS is just an extension of this duality. Indeed, PKSs conveniently model state abstractions while MTSs are a natural formalism for representing transition abstractions.

The third 3-valued formalism, KMTSs, was created with the purpose of unifying both views by combining features of both PKSs and MTSs [HJS01], although without increasing expressiveness as demonstrated by the results of the present paper. The "syntactic sugaring" provided by KMTSs enables a more nimble representation of abstractions of reactive systems. We illustrate this claim with two examples of applications where the use of KMTSs is convenient: predicate abstraction and shape analysis.

Predicate abstraction (e.g, [GS97]) abstracts a program using a set $\Psi = \{\psi_1, \ldots, \psi_n\}$ of n predicates, each of which is typically represented by a quantifier-free formula of first-order logic (such as $(x == y+1) \vee (x < y-5)$) (e.g., [DDP99]). An abstract state is defined as a vector of n truth-values, which identifies all concrete states that satisfy the same set of predicates in Ψ. Typically, consistency of states is checked using theorem-proving techniques for quantifier-free first-order logic. The resulting abstract transition system can conveniently be represented by a KMTS: propositions at abstract states corresponding to the predicates in Ψ can be 3-valued, with the third value $\perp$ modeling a loss of information due to abstraction or the potential incompleteness of the auxiliary theorem prover; similarly, the transition relation is also 3-valued, where *may*-only transitions can be used to model transitions from a conditional node (such as program statements of the form `if-then-else`) whose guard's evaluation is unknown (e.g., [GHJ01]).

Shape analysis (e.g., [CWZ90]) is a form of pointer analysis where the contents of heap storage is approximated by a graph, whose nodes denote objects and whose arcs denote the values of the objects' fields. Local-variable points-to information can be represented using unary predicates (e.g., $x(u_0)$ is true at an object u_0 if variable x points to object u_0) and binary predicates ($next(u_0, u_1)$ is true if the $next$ field of object u_0 points to object u_1) (e.g., [SRW99]). Again, KMTSs can conveniently encode such abstractions: a proposition x can be associated with each unary predicate $x(u_0)$ at every node of the graph in such a way that proposition x has $\perp$ for value at object (node) u_0 if it is possible but not guaranteed for variable x to point to object u_0 in the current heap; similarly, binary predicates can be modeled as a 3-valued transition relation, where the arc with label *next* from u_0 to u_1 is a *may*-only transition if there is a possibility but no guarantee for the *next* field of object u_0 to point to u_1 (e.g., [HJS01]).

We conclude by mentioning a few other modeling formalisms closely related to those considered in this paper. Variants of KMTSs with labeled transitions and two interpretation functions L^{may} and L^{must} are defined in [HJS01]; it is straightforward to show by using the translation techniques of our paper that these simple extensions do not increase expressiveness over the definition of KMTS considered in Section 2. Extended transition systems [Mil81] can be viewed as a particular class of MTSs [HJS01,BG99]. Mixed transition systems [Dam96] are MTSs where the constraint $\xrightarrow{must} \subseteq \xrightarrow{may}$ is removed; eliminating this constraint makes it possible to specify inconsistent models, i.e., models that cannot be refined by any complete systems [HJS02], and hence increases expressiveness compared to the modeling formalisms considered in this paper. Recent work on 3-valued models for probabilistic systems can be found in [Hut02b].

References

[BG99] G. Bruns and P. Godefroid. Model Checking Partial State Spaces with 3-Valued Temporal Logics. In *Proceedings of the 11th Conference on Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 274–287, Trento, July 1999. Springer-Verlag.

[BG00] G. Bruns and P. Godefroid. Generalized Model Checking: Reasoning about Partial State Spaces. In *Proceedings of CONCUR'2000 (11th International Conference on Concurrency Theory)*, volume 1877 of *Lecture Notes in Computer Science*, pages 168–182, University Park, August 2000. Springer-Verlag.

[BPR01] T. Ball, A. Podelski, and S. K. Rajamani. Boolean and Cartesian Abstraction for Model Checking C Programs. In *Proceedings of TACAS'2001 (Tools and Algorithms for the Construction and Analysis of Systems)*, volume 2031 of *Lecture Notes in Computer Science*. Springer-Verlag, April 2001.

[BR01] T. Ball and S. Rajamani. The SLAM Toolkit. In *Proceedings of CAV'2001 (13th Conference on Computer Aided Verification)*, volume 2102 of *Lecture Notes in Computer Science*, pages 260–264, Paris, July 2001. Springer-Verlag.

[CWZ90] D.R. Chase, M. Wegman, and F.K. Zadeck. Analysis of pointers and structures. In *Proccedings of Conference on Programming Language Design and Implementation*, pages 296–310, June 1990.

[Dam96] D. Dams. *Abstract interpretation and partition refinement for model checking*. PhD thesis, Technische Universiteit Eindhoven, The Netherlands, 1996.

[DDP99] S. Das, D. L. Dill, and S. Park. Experience with Predicate Astraction. In *Proc. of the 11th International Conference on Computer-Aided Verification*, Lecture Notes in Computer Science, pages 160–172, Trento, July 1999. Springer Verlag.

[GHJ01] P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based Model Checking using Modal Transition Systems. In *Proceedings of CONCUR'2001 (12th International Conference on Concurrency Theory)*, volume 2154 of *Lecture*

Notes in Computer Science, pages 426–440, Aalborg, August 2001. Springer-Verlag.

[GJ02] P. Godefroid and R. Jagadeesan. Automatic Abstraction Using Generalized Model Checking. In *Proceedings of CAV'2002 (14th Conference on Computer Aided Verification)*, volume 2404 of *Lecture Notes in Computer Science*, pages 137–150, Copenhagen, July 2002. Springer-Verlag.

[GS97] S. Graf and H. Saidi. Construction of Abstract State Graphs with PVS. In *Proceedings of the 9th International Conference on Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83, Haifa, June 1997. Springer-Verlag.

[HJMS02] T. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Lazy Abstraction. In *Proceedings of the 29th ACM Symposium on Principles of Programming Languages*, pages 58–70, Portland, January 2002.

[HJS01] M. Huth, R. Jagadeesan, and D. Schmidt. Modal Transition Systems: a Foundation for Three-Valued Program Analysis. In *Proceedings of the European Symposium on Programming (ESOP'2001)*, volume 2028 of *Lecture Notes in Computer Science*. Springer-Verlag, April 2001.

[HJS02] M. Huth, R. Jagadeesan, and D. Schmidt. A Domain Equation for Refinement of Partial Systems. Submitted to Mathematical Structures in Computer Science, 2002.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[Hut02a] M. Huth. Model Checking Modal Transition Systems using Kripke Structures. In *Proceedings of the Third International Workshop on Verification, Model Checking and Abstract Interpretation (VMCAI'2002)*, volume 2294 of *Lecture Notes in Computer Science*, pages 302–316, Venice, January 2002. Springer-Verlag.

[Hut02b] M. Huth. Possibilistic and Probabilistic Abstraction-based Model Checking. In *Proceedings of the PAPM-Probmiv 2002 Workshop*, volume 2329 of *Lecture Notes in Computer Science*, pages 115–134, Copenhagen, July 2002. Springer-Verlag.

[Kle87] S. C. Kleene. *Introduction to Metamathematics.* North Holland, 1987.

[Koz83] D. Kozen. Results on the Propositional Mu-Calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[Lar89] K. G. Larsen. Modal Specifications. In J. Sifakis, editor, *Workshop on Automatic Verification Methods for Finite-State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 232–246. Springer-Verlag, June 1989. International Workshop, Grenoble, France.

[LT88] K. G. Larsen and B. Thomsen. A Modal Process Logic. In *Proceedings of Third Annual Symposium on Logic in Computer Science*, pages 203–210. IEEE Computer Society Press, 1988.

[Mil81] R. Milner. A Modal Characterization of Observable Machine Behavior. In *Proc. CAAP'81*, volume 112 of *Lecture Notes in Computer Science*, pages 25–34. Springer-Verlag, 1981.

[MSS99] M. Muller-Olm, D. Schmidt, and B. Steffen. Model Checking: A Tutorial Introduction. In *Proceedings of the 6th International Static Analysis Symposium (SAS'99)*, volume 1694 of *Lecture Notes in Computer Science*, pages 331–354, Berlin, September 1999. Springer-Verlag.

[SRW99] M. Sagiv, T. Reps, and R. Wilhelm. Parametric Shape Analysis Via 3-Valued Logic. In *Proceedings of the 26th ACM Symposium on Principles of Programming Languages*, January 1999.