Modal transition systems: a foundation for three-valued program analysis

Michael Huth\textsuperscript{1}, Radha Jagadeesan\textsuperscript{*2}, and David Schmidt\textsuperscript{**1}

\textsuperscript{1} Computing and Information Sciences, Kansas State University, 
\{huth, schmidt\}@cis.ksu.edu. WWW home page: 
http://www.cis.ksu.edu/\{huth, schmidt\}

\textsuperscript{2} Department of Mathematics and Computer Science, Loyola University of Chicago, 
radha@cs.luc.edu. WWW home page: http://www.cs.luc.edu/radha

**Abstract.** We present Kripke modal transition systems (Kripke MTSs), a generalization of modal transition systems \cite{27,26}, as a foundation for three-valued program analysis. The semantics of Kripke MTSs are presented by means of a mixed power domain of states; soundness and consistency are proved. Two major applications, model checking partial state spaces and three-valued program shape analysis, are presented as evidence of the suitability of Kripke MTSs as a foundation for three-valued analyses.

1 Introduction

A modal transition system (MTS) \cite{27,26} labels each of its state transitions with a modality — may or must — expressing transition behaviors that (i) necessarily occur (must modality), (ii) possibly occur (may modality), and (iii) not possibly occur (absence of a transition). Figure 1 shows an example MTS — a specification of a slot machine, where some behaviors of the final implementation are fixed (the must-transitions) and some are uncertain (the may-transitions).

Conventional state-transition modellings are over-approximations made by adding more computation paths \cite{8,7}, thereby limiting validation to safety properties (“nothing bad will happen”). MTSs, however, perform both over- and under-approximation, admitting both safety and liveness properties (“something good will happen”) to be deduced. As a bonus, the outcomes of analyses of MTSs are three-valued, meaning that validation, refutation, and conditional reasoning can be undertaken in the framework. Abstractions of both control and data can be modelled with MTSs.

The paper proceeds as follows: Section 2 reviews doubly labeled transition systems \cite{12} and characterizes their behaviors by logics of “liveness” and “safety.” Section 3 introduces Kripke MTSs, their refinement relation, and their semantics for the modal mu-calculus \cite{25}, Sections 4 and 5 show how to apply Kripke MTSs to two analyses that rely on three-valued logic: (i) Bruns and Godefroid’s partial

\textsuperscript{*} Supported by NSF CCR-9901071.

\textsuperscript{**} Supported by NSF CCR-9970679 and INT-9981558.
Kripke structures [3] and extended transition systems [29,39] and (ii) the pointer shape-graph analysis of Sagiv, Reps, and Wilhelm [35]. Section 6 concludes.

2 Doubly labeled transition systems

We begin with the definition of a doubly labeled transition system [12]:

Definition 1 (Doubly labeled transition systems). A doubly labeled transition system (DLTS), $\mathcal{K}$, is a tuple $(\Sigma_K, \text{act}, AP, \rightarrow, L)$, where $\Sigma_K$ is a set of states, $\text{act}$ is a (countable) set of action symbols, $AP$ is a (countable) set of atomic propositions, $\rightarrow$ is a transition relation that is a subset of $\Sigma_K \times \text{act} \times \Sigma_K$, and $L$ is a labeling function $L: \Sigma_K \rightarrow \mathcal{P}(AP)$. We call $\mathcal{K}$ finitely-branching if for each $s \in \Sigma_K$ and $a \in \text{act}$, the sets $L(s)$ and $\{s' \in \Sigma_K | s \rightarrow^a s'\}$ are finite.

As Figure 2 shows, each state is annotated with the set of primitive properties that hold for it. Behaviors are compared by means of simulations:

Definition 2 (Simulation). Let $\mathcal{C}$ and $\mathcal{A}$ be doubly labeled transition systems, where for simplicity $\text{act}_C = \text{act}_A$. A relation, $Q \subseteq \Sigma_C \times \Sigma_A$, is a simulation if, for all $s \in \Sigma_C$, $t \in \Sigma_A$, $Q(s,t)$ holds iff: for all $a \in \text{act}_C$ and $s' \in \Sigma_C$ with $s \rightarrow^a s'$, there is some $t' \in \Sigma_A$ with $t \rightarrow^a t'$ and $Q(s',t')$.

Given $\mathcal{C}$ and $\mathcal{A}$, we can compute the greatest simulation, $\preceq$, on $\Sigma_C \times \Sigma_A$, a preorder, by a standard fixed-point argument. The intuition behind a simulation is that a transition made by $\mathcal{C}$ can be "mimicked" by one in $\mathcal{A}$. In practice, one of $\mathcal{C}$ or $\mathcal{A}$ is an "implementation" and the other is its "abstraction" or "specification" or "model," which must be analyzed for correctness properties. As noted in [36], there are natural connections between simulations and Galois connections [8] on such transition systems.

Live Simulations. Simulations should respect atomic properties. A simulation, $Q \subseteq \Sigma_C \times \Sigma_A$, is a live simulation if $Q(s,t)$ implies $L(s) \subseteq L(t)$, for all $s \in \Sigma_C$ and $t \in \Sigma_A$. It is easy to prove that there is a greatest live simulation, $\preceq_{live}$, on $\Sigma_C \times \Sigma_A$. There is a crucial connection between live simulations and modal logic:

1 Making $L(s)$ finite as well prevents inconsistencies if we convert state propositions into action labels.
Consider the following modal logic, $L_{\text{pos}}$, which expresses liveness or "possibility" properties\cite{33}, where $p \in \text{AP}$ and $a \in \text{Act}$;

$$\phi ::= \top \mid p \mid \phi_1 \land \phi_2 \mid \phi_1 \lor \phi_2 \mid \langle a \rangle \phi$$

The diamond modality denotes the possibility of an $a$-transition. For a DLTS, $K$, we define $\models \phi \subseteq \Sigma_K$ by induction on the grammar for $\phi$:

$$\models \top \equiv \Sigma_K,$$

$$\models p \equiv \{ s \in \Sigma_K \mid p \in L(s) \},$$

$$\models \phi_1 \land \phi_2 \equiv \{ \phi_1 \cap \phi_2 \},$$

$$\models \phi_1 \lor \phi_2 \equiv \{ \phi_1 \cup \phi_2 \},$$

$$\models \langle a \rangle \phi \equiv \{ s \in \Sigma_K \mid \text{for some } s', s \rightarrow^a s' \text{ and } s' \in \models \phi \}.$$

**Proposition 1 (Logical characterization).** [18] Let $C$ and $A$ be finitely-branching DLTSs and $s \in \Sigma_C$, $t \in \Sigma_A$. Then $s \sim_{\text{live}} t$ iff for all $\phi \in L_{\text{pos}}$, $[s \models \phi] \Rightarrow t \in \models \phi$.

Thus, to calculate liveness properties of an implementation, $A$, we construct a model, $C$, and calculate the greatest live simulation. Then, liveness properties that are deduced to hold for $C$'s states will hold for the corresponding states in $A$. Dually, we might model an implementation, $C$, by an abstract model, $A$, and use the latter to refute liveness properties of $C$.

**Safe simulations.** The dual of a live simulation is a safe one: a simulation, $Q \subseteq \Sigma_C \times \Sigma_A$, is safe if $Q(s,t)$ implies $L(s) \supseteq L(t)$, for all $s \in \Sigma_C$ and $t \in \Sigma_A$. There is a greatest safe simulation, $\sim_{\text{safe}}$, on $\Sigma_C \times \Sigma_A$. The logic $L_{\text{nec}}$ expresses safety or "necessarily" properties\cite{33}, where $p \in \text{AP}$ and $a \in \text{Act}$:

$$\phi ::= \top \mid p \mid \phi_1 \land \phi_2 \mid \phi_1 \lor \phi_2 \mid [a] \phi$$

We define $[s \models \phi] \subseteq \Sigma_K$ for the first four clauses in the same manner as for the logic $L_{\text{pos}}$, and we define $[a] \phi \equiv \{ s \in \Sigma_K \mid \text{for all } s', s \rightarrow^a s' \text{ implies } s' \in \models \phi \}$ as the meaning of the box modality.

**Proposition 2 (Logical characterization).** Let $C$ and $A$ be finitely-branching DLTSs and $s \in \Sigma_C$, $t \in \Sigma_A$. Then $s \sim_{\text{safe}} t$ iff for all $\phi \in L_{\text{nec}}$, $[t \models \phi] \Rightarrow s \in \models \phi$.  

---

**Fig. 2.** Slot machine implementation as a doubly labeled transition system

---
Thus, to calculate safety properties of an implementation, \( C \), we construct \( \mathcal{A} \) and calculate a safe simulation. Then, safety properties that hold for \( \mathcal{A} \)'s states will hold for the corresponding states in \( C \). (This is the standard approach in abstract interpretation studies [8,36].) Dually, we might model an implementation, \( \mathcal{A} \), by an abstract model, \( \mathcal{C} \), and use the latter to refute safety properties of \( \mathcal{A} \).

## 3 Modal transition systems

Kripke MTSs allow us to freely combine safety and liveness properties in property validation and refutation. An MTS’s “loose” transitions — that is, transitions that may or may not be present in the final implementation — are labeled as may-transitions, and “tight” transitions, which must be preserved in the final implementation, are labeled as must-transitions [27]. Review Figure 1. With this intuition, every must-transition is by definition a may-transition. These ideas also apply to the atomic properties that label an MTS’s states, giving us the modal version of DLTSs, which we call a Kripke MTS:

**Definition 3 (Kripke MTS).** A Kripke MTS is a tuple \( \mathcal{K} = (\Sigma, \text{act}, \mathcal{A}, \mathcal{P}, \mathcal{L}^\text{must}, \mathcal{L}^\text{may}) \), where both \( (\Sigma, \text{act}, \mathcal{A}, \mathcal{P}, \mathcal{L}^\text{must}) \) and \( (\Sigma, \text{act}, \mathcal{A}, \mathcal{P}, \mathcal{L}^\text{may}) \) are DLTSs with \( \mathcal{L}^\text{must} \subseteq \mathcal{L}^\text{may} \) and \( \mathcal{L}^\text{must}(s) \subseteq \mathcal{L}^\text{may}(s) \), for all \( s \in \Sigma \).

Note the pairings: \( \mathcal{L}^\text{must} \) and \( \mathcal{L}^\text{may} \) are paired, because they define a system of transitions and properties that must be preserved in any implementation of the Kripke MTS; \( \mathcal{L}^\text{must} \) and \( \mathcal{L}^\text{may} \) are paired, because they define a system of transitions and properties that may be preserved in any implementation. For a Kripke MTS \( \mathcal{C} \) to be a refinement of a Kripke MTS \( \mathcal{A} \), it must preserve all must-aspects of \( \mathcal{A} \) and it may selectively discard \( \mathcal{A} \)'s may-aspects:

**Definition 4 (Refinement).** A refinement between Kripke MTSs \( \mathcal{C} \) and \( \mathcal{A} \) is a relation \( Q \subseteq \Sigma \times \Sigma \) such that, for all \( s \in \Sigma \) and \( t \in \Sigma \), if \( Q(s,t) \), then

1. if \( s \rightarrow^\text{must} t \), then for some \( s' \in \Sigma \), \( s \rightarrow^\text{must} s' \) and \( Q(s',t) \);
2. if \( s \rightarrow^\text{may} s' \), then for some \( t' \in \Sigma \), \( t \rightarrow^\text{may} t' \) and \( Q(s',t') \);
3. \( \mathcal{L}^\text{must}(s) \subseteq \mathcal{L}^\text{must}(t) \) and \( \mathcal{L}^\text{may}(s) \subseteq \mathcal{L}^\text{may}(t) \).

A Kripke MTS such that \( \mathcal{L}^\text{must} = \mathcal{L}^\text{may} \) and \( \mathcal{L}^\text{must} = \mathcal{L}^\text{may} \) is concrete, that is, it is a doubly labeled transition system [12], a “final implementation.” As usual, for Kripke MTSs \( \mathcal{C} \) and \( \mathcal{A} \), there is a greatest refinement relation \( \prec_r \).

Next, consider the logic \( \mathcal{L} \):

\[
\phi ::= \top \mid p \mid \phi_1 \land \phi_2 \mid \langle a \rangle \phi \mid [a]\phi
\]  

where \( p \in \mathcal{P} \) and \( a \in \text{act} \).

**Definition 5 (Semantics of modal logic).** For a Kripke MTS \( \mathcal{K} \) and any \( \phi \in \mathcal{L} \), we define a semantics \( \models \phi \in \mathcal{P}(\Sigma_K) \times \mathcal{P}(\Sigma_K) \), where \( \mathcal{P}(\Sigma_K) \) is the powerset of \( \Sigma_K \), ordered by set inclusion, and \( \models [\phi]^{\text{conc}} \) and \( \models [\phi]^{\text{pos}} \) are the projection of \( \models [\phi] \) to its first and second component, respectively:
1. \[ \top \models \defeq (\Sigma_K, \Sigma_K); \]
2. \( p \models \defeq \{ s \in \Sigma_K | p \in L^{\text{must}}(s) \} \cup \{ s \in \Sigma_K | p \in L^{\text{may}}(s) \}; \]
3. \( \phi_1 \land \phi_2 \models \defeq (\phi_1)_{\text{sec}} \cap (\phi_2)_{\text{sec}}, \models (\phi_1)_{\text{pos}} \cap (\phi_2)_{\text{pos}}; \]
4. \[ \langle a \rangle \phi \models \defeq \{ s \in \Sigma_K | \text{for some } s', s \rightarrow^{a}_{\text{must}} s' \text{ and } s' \in \{ \phi \}_{\text{sec}}, \}
\{ s \in \Sigma_K | \text{for some } s', s \rightarrow^{a}_{\text{may}} s' \text{ and } s' \in \{ \phi \}_{\text{pos}} \}; \]
5. \[ [a] \phi \models \defeq \{ s \in S | \text{for all } s', s \rightarrow^{a}_{\text{may}} s' \text{ implies } s' \in \{ \phi \}_{\text{sec}}, \}
\{ s \in S | \text{for all } s', s \rightarrow^{a}_{\text{must}} s' \text{ implies } s' \in \{ \phi \}_{\text{may}} \}; \]

The “necessarily” interpretation, \( \models \phi \)_{\text{sec}}, is an under-approximation of those states for which a proposition necessarily holds true (that is, the states for which the proposition holds for all future refinements/implementations). Dually, the “possibly” interpretation, \( \models \phi \)_{\text{pos}}, is an over-approximation of those states for which there is some refinement for which the proposition holds. The semantics \( \models \phi \)_{\text{sec}} is the one given by Larsen [26]; it produces this result:

**Proposition 3 (Logical characterization).** Let \( C \) and \( A \) be finitely-branching\(^3\) Kripke MTs and \( s \in \Sigma_C, t \in \Sigma_A \). Then \( s \prec_{C,t} \) iff for all \( \phi \in \mathcal{L} \), \[ t \models \phi \models \text{sec} \Rightarrow s \in \models \phi \models \text{sec}. \]

This result tells us to build an MTs, \( A \), that abstracts an implementation, \( C \). Both safety and liveness properties can be validated on \( A \), and they carry over to \( C \). Using the “possibly” interpretation, a new logical characterization follows, allowing us to refute safety and liveness properties of an implementation, \( C \), by refuting them on \( A \):

**Proposition 4 (Logical characterization).** Let \( C \) and \( A \) be finitely-branching Kripke MTs and \( s \in \Sigma_C, t \in \Sigma_A \). Then \( [s] \prec_{C,t} [t] \) iff for all \( \phi \in \mathcal{L} \), \[ [s] \models \phi \models \text{pos} \Rightarrow [t] \models \phi \models \text{pos}. \]

**Negation and invariants.** We can retain both validation and refutation on Kripke MTs if we add negation and recursive definition to our logic, giving the modal-mu calculus [25,2], \( \text{Act}_{\mu} \):

\[ \phi ::= \top \cup p \cup Z \cup \neg \phi \cup \phi_1 \land \phi_2 \cup \langle a \rangle \phi \cup [a] \phi \cup \mu Z. \phi \]

(4)

where \( p \) ranges over \( \text{Act} \), \( Z \) over a (countable) set of variables, \( a \in \text{Act} \), and the bodies \( \phi \) in \( \mu Z. \phi \) are formally monotone. Disjunction (\( \lor \)) and implication (\( \to \)) are derived as \( \neg (\neg \phi \land \neg \psi) \) and \( \neg (\phi \land \neg \psi) \), respectively. This logic is very expressive, and important specification logics like CTL\(^*\) can be embedded into it [10]. We require environments, \( \rho \), mapping variables \( Z \) to elements of \( P(\Sigma_K) \times P(\Sigma_K) \). The semantics, \( \models \phi \models \rho = P(\Sigma_K) \times P(\Sigma_K) \), is defined as for \( \mathcal{L} \), but parametric in an environment \( \rho \); this semantics is essentially the one in [19]. Given the must- and may-aspects of an MTs, the semantics of negation is delicate, and we follow Kest [24] and Levi [28]. The semantics of the remaining clauses, \( Z \) and \( \mu Z. \phi \), are dealt with in a standard manner:

\(^2\) Larsen's results were proved for MTs.

\(^3\) For all \( s \in \Sigma_K \) and \( a \in \text{Act} \), the sets \( \{ s' \in \Sigma_K | s \to^{a}_{\text{may}} s' \} \) and \( L^{\text{may}}(s) \) are finite.
1. $|Z|_\rho \mathrel{=} \rho(Z)$;
2. $\neg \phi |_\rho \mathrel{=} (\Sigma_K \setminus |\phi |_\rho^{\text{pos}}, \Sigma_K \setminus |\phi |_\rho^{\text{neg}})$;
3. $|\mu Z. \phi |_\rho$ is the least fixed point of the monotone functional
   \[ d \mapsto |\phi |_{\rho|_{Z \to d}} : \mathcal{P}(\Sigma_K) \times \mathcal{P}(\Sigma_K) \to \mathcal{P}(\Sigma_K) \times \mathcal{P}(\Sigma_K). \]

Note the semantics of negation: "necessarily $\neg \phi$" is "not possibly $\phi$," and "possibly $\neg \phi$" is "not necessarily $\phi$".

**Theorem 1 (Soundness and consistency of semantics).** For any Kripke MTS $K$, $\phi, \psi \in \text{Act}\mathcal{M}_K$, and environment $\rho$:

1. $|\phi |_\rho^{\text{neg}} \subseteq |\phi |_\rho^{\text{pos}}$.
2. $|\phi \wedge \neg \phi |_\rho^{\text{neg}} = \emptyset$; and $|\phi \lor \neg \phi |_\rho^{\text{pos}} = \Sigma_K$. That is, the semantics is consistent for $|\phi |_\rho^{\text{neg}}$ and "complete" for $|\phi |_\rho^{\text{pos}}$;
3. if $s \leq t$, then $t \in |\phi |_\rho^{\text{neg}}$ implies $s \in |\phi |_\rho^{\text{neg}}$; and $s \in |\phi |_\rho^{\text{pos}}$ implies $t \in |\phi |_\rho^{\text{pos}}$. That is, the semantics is sound;
4. if $K$ is concrete, then $|\phi |_\rho^{\text{neg}} = |\phi |_\rho^{\text{pos}}$ and corresponds to the standard semantics for doubly labeled transition systems (as given for CTL* in [12]).

The semantics of negation behaves classically, that is, $|\neg \neg \phi |_\rho = |\phi |_\rho$ and $|\neg (a) \phi |_\rho = |[a] \neg \neg \phi |_\rho$ hold. The underlying interpretations $|\phi |_\rho^{\text{neg}}$ and $|\phi |_\rho^{\text{pos}}$, however, are not classical, but three-valued, in the sense that a state, $s$, can possess a property, $\phi$, in only three possible ways:

1. $s \in |\phi |_\rho^{\text{neg}}$ (and hence, $s \in |\phi |_\rho^{\text{pos}}$): "$\phi$ necessarily holds for $s$;"
2. $s \in |\phi |_\rho^{\text{pos}}$ and $s \not\in |\phi |_\rho^{\text{neg}}$ ($s \in |\phi \wedge \neg \phi |_\rho^{\text{pos}}$): "$\phi$ possibly holds for $s$;"
3. $s \not\in |\phi |_\rho^{\text{pos}}$ (hence, $s \not\in |\phi |_\rho^{\text{neg}}$): "$\phi$ does not possibly hold for $s$;"

Note the loss of precision in the second case above: $s \in |\phi \wedge \neg \phi |_\rho^{\text{pos}}$; there cannot be a final implementation which satisfies $\phi \wedge \neg \phi$. For the partial Kripke structures of Section 4, a more precise analysis is possible [4].

To finish the technical development, we note that the Kripke MTS semantics and its properties adapt to the scenario where $\Sigma_K$ is no longer flat. For finite-state models, meanings $|\phi |_\rho$ are elements of the mixed power domain $M[\Sigma_K]$ [15, 17, 20].\footnote{Elements of $M[\Sigma_K]$ are pairs $(H, S)$, where $H$ is a Scott-closed lower subset and $S$ a Scott-compact upper subset of $\Sigma_K$ such that $H$ equals $\{s \in \Sigma_K \mid \exists s' \in H \cap S: s \leq s'\}$. This consistency condition replaces the inclusion requirement $(H \subseteq S)$ of the discrete case.} This lets us adapt standard abstract interpretation studies and even lets us define and manipulate a fully abstract domain of MTSs by the isomorphism, $D \cong \prod_{\alpha \in \text{Act}} M[D]$ [20].

**Abstract Kripke MTSs.** Let $\mathcal{C} = (\Sigma_K, \text{Act}, \text{AP}, \to, L)$ be a possibly infinite-state DLTS. Given a finite set of boolean predicates $\{p_1, p_2, \ldots, p_n\}$, we can derive a finite-state abstract Kripke MTS $\mathcal{A}$; abstract states, $\alpha$, are equivalence classes of states that satisfy exactly the same predicates $p_i$ in $\mathcal{C}$; abstract transitions are defined, as in [11], by (i) $[s] \to^{\text{must}} [s']$ iff for all $s* \in [s]$,...
there exists $s'_* \in [s']$ such that $s_* \rightarrow^a s'_*$ and (ii) $[s] \rightarrow^m_{m_{\text{any}}} [s']$ iff there exist $s_* \in [s]$ and $s'_* \in [s']$ such that $s_* \rightarrow^a s'_*$; finally, propositions are defined as $L^\text{must}([s]) \overset{\text{def}}{=} \bigcap_{s' \in [s]} L(s')$ and $L^\text{may}([s]) \overset{\text{def}}{=} \bigcup_{s' \in [s]} L(s')$.

This refinement relationship is well behaved and induces two Galois connections [8], one between the concrete model $\mathcal{C}$ and the may-component of $\mathcal{A}$, and one between $\mathcal{C}$ and $\mathcal{A}$'s must-component. This phenomenon is foreshadowed by the universal ($\bigwedge^\forall$) and existential ($\bigwedge^\exists$) abstraction maps of Cousot and Cousot [9], which extract may- and must-behaviors from linear-time models.

It is immediate that $\mathcal{C}$ is a refinement of $\mathcal{A}$, giving us a sound tool for verifying $\textbf{and}$ refuting $\textbf{all}$ properties expressed in the modal mu-calculus. With some effort, this approach applies to refinements to non-concrete Kripke MTs as well.

4 Abstracting control and data

**Partial Kripke structures.** For model checking partial state spaces, Bruns and Godefroid devised partial Kripke structures [3]:

**Definition 6.** A partial Kripke structure\(^5\) is a 4-tuple, $\mathcal{K} = (\Sigma, \mathcal{A}, \rightarrow, L)$, where $\Sigma$ is a set of states, $\mathcal{A}$ is a set of atomic propositions, $\rightarrow \subseteq \Sigma \times \Sigma$ is a set of transitions, and $L: \Sigma \times \mathcal{A} \rightarrow \mathbf{3}$ is a labeling function, where $\mathbf{3}$ is the set $\{\bot, \top\}$ endowed with the information ordering $\bot \leq \top$ and $\bot \leq T$.

Figure 3 depicts three partial Kripke structures with initial states $s_1, s_2, s_3$ respectively [3]. We write $p = v$ at a state $s$ to denote $L(s, p) = v$. These systems have different information regarding the truth of $p$ at their initial states and their rightmost successor states. At their leftmost successor states, all three systems leave the truth status of $p$ unresolved. Below, we show that these systems, their abstraction notion and temporal logic semantics are special instances of the corresponding notions for Kripke MTs. Partial Kripke structures are related in the following fashion:

**Definition 7 (Completeness order).** A completeness order [3] on two partial Kripke structures is a binary relation $Q \subseteq \Sigma \times \Sigma$ such that $Q(s, t)$ implies

\(^5\) We assume that these structures are finitely branching.
1. for all $p \in \mathcal{A}$, $L(s, p) \leq L(t, p)$ in the information ordering,
2. if $s \rightarrow s'$, then there is some $t' \in \Sigma_A$ with $t \rightarrow t'$ and $Q(s', t')$, and
3. if $t \rightarrow t'$, then there is some $s' \in \Sigma_C$ with $s \rightarrow s'$ and $Q(s', t')$.

In usual fashion, we write $s \ll t$ if there is a completeness order $Q$ in which $Q(s, t)$ holds. Properties of states of partial Kripke structures are expressed in the logic PML: $\phi := p \mid \neg \phi \mid \phi \land \phi \mid \diamond \phi$. Bruns and Godefroid require a second, truth ordering on the set \{\bot, \top\} for defining a three-valued semantics for this logic: $\mathcal{F} < \bot < \top$. Denotations $[s \models \phi]$ are elements of \{T \subseteq \mathcal{F}\}:

$$
[s \models p] \overset{\text{def}}{=} L(s, p)
$$

$$
[s \models \neg \phi] \overset{\text{def}}{=} \neg s \models \phi
$$

$$
[s \models \phi_1 \land \phi_2] \overset{\text{def}}{=} \min\{[s \models \phi_1], [s \models \phi_2]\}
$$

$$
[s \models \diamond \phi] \overset{\text{def}}{=} \max\{[s' \models \phi] \mid s \rightarrow s'\},
$$

where $\neg$ is strict logical complement and min and max are meet and join in the truth ordering, respectively. Bruns and Godefroid logically characterize the completeness preorder: For partial Kripke structures $\mathcal{C}$ and $\mathcal{A}$, $s \in \Sigma_C, t \in \Sigma_A$, $s \ll t$ iff for all $\phi$ in PML, $[s \models \phi] \leq [t \models \phi]$ in the truth ordering.

The embedding of partial Kripke structures into Kripke MTS rests on the order-isomorphism $\Psi : (E \rightarrow 3) \rightarrow \mathcal{M}[E]$ between the mixed power domain $\mathcal{M}[E]$ and the set of all functions $E \rightarrow 3$, ordered pointwise in the information ordering — if $E$ is discrete; identify a function $f : E \rightarrow 3$ with the pair $\Psi f \overset{\text{def}}{=} (f^{-1}\{\bot\}, f^{-1}\{\top\})$. We then translate a partial Kripke structure, $\mathcal{C}$, into the Kripke MTS, $\mathcal{K}' = (\Sigma_K, \{\ast\}, \mathcal{A}, \rightarrow, \rightarrow, L^{mut}, L^{maj})$, where the propositional component $(L^{mut}(s), L^{maj}(s))$ is defined as $\Psi L$.

**Proposition 5 (Correspondence of semantics).** For partial Kripke structure, $\mathcal{K}$, and its Kripke MTS translation, $\mathcal{K}'$, for all $s \in \Sigma_K$ and $\phi$ in PML, $\models \phi = \Psi (\lambda s. [s \models \phi])$. The inverse $\overset{\text{def}}{=}^{-1}$ of the greatest completeness order $\ll$ on $\mathcal{K}$ is the greatest refinement of its Kripke MTS translation $\mathcal{K}'$.

The full MTS formulation, unlike partial Kripke structures, allows for modalities on transitions and remains well defined when the domain of states is nonflat, making it applicable to conventional abstraction frameworks [8, 7]. To illustrate model checks on such systems, consider $\phi \overset{\text{def}}{=} \mu Z.p \lor ([\ast]Z \land (\ast) \top)$, saying “$p$ holds eventually on all paths” [2], on the systems of Figure 3.7. Our semantics computes $s_1 \in \models \phi \overset{\text{def}}{=} \mu \overset{\text{def}}{=} \text{rec}$, since $p$ is true at $s_1$; $s_2 \in \models \phi \overset{\text{def}}{=} \text{pos}$, since $p$ may be true at $s_2$; and $s_3 \not\in \models \phi \overset{\text{def}}{=} \text{pos}$, since there is a path on which $p$ is never true.

**Partial bisimulations.** Partial Kripke structures abstract propositional information only. Bruns and Godefroid also studied systems that abstract state transitions, the so-called extended transition systems [3], and their partial bisimulation [29, 39]:

---

6 We identify $\Sigma_K \times \Sigma_K$ with $\Sigma_K \times \{\ast\}$.

7 We use $\ast$ in $\phi$ to denote the sole action type.
**Definition 8.** An extended transition system (ETS)\(^8\) \([3]\) is a 4-tuple \(\mathcal{E} = (\Sigma_E, \text{Act}, \rightarrow, \uparrow)\), where \(\Sigma_E\) is a set of states, \(\rightarrow \subseteq \Sigma_E \times \text{Act} \times \Sigma_E\) is a set of transitions, and \(\uparrow \subseteq \Sigma_E \times \text{Act}\) is a divergence relation.

Read \(s \uparrow a\) as “some of the \(a\)-transitions from \(s\) in the full model may be missing at \(s\) in the ETS” \([3]\). We write \(s \downarrow a\) when \(s \uparrow a\) fails to hold, meaning that all \(a\)-transitions from \(s\) in the full state space are present at \(s\) in the ETS.

**Definition 9.** Given ETSs \(\mathcal{E}\) and \(\mathcal{F}\), a partial bisimulation \([29, 39]\) (divergence preorder \([3]\)) is a subset, \(Q\), of \(\Sigma_E \times \Sigma_F\) such that \(Q(s, t)\) implies

1. whenever \(s \rightarrow_a s'\), there exists some \(t' \in \Sigma_F\) such that \(t \rightarrow_a t'\) and \(Q(s', t')\);
2. if \(s \downarrow a\), then (i) \(t \downarrow a\), and (ii) whenever \(t \rightarrow_a t'\), there exists some \(s' \in \Sigma_E\) such that \(s \rightarrow_a s'\) and \(Q(s', t')\).

Every ETS has a greatest partial bisimulation, \(\subseteq\). We embed \(\mathcal{E}\) into a Kripke MTS \(T[\mathcal{E}] \cong (\Sigma_E, \text{Act}, 0, \rightarrow_{\text{must}}, \rightarrow_{\text{may}}, 0, 0)\) by (i) \(s \rightarrow_{\text{must}} s'\) iff \(s \rightarrow_a s'\) and (ii) \(s \rightarrow_{\text{may}} s'\) iff \((s \rightarrow_a s'\) or \(s \uparrow a\)). Note how \(\uparrow\) makes \(\mathcal{E}\) three-valued in may-transitions.

**Theorem 2.** Given an ETS \(\mathcal{E}\), \(T[\mathcal{E}]\) is an MTS satisfying, for all \(s \in \Sigma_E\),

\[(\exists s' \in \Sigma_E: s \rightarrow_{\text{may}} s' \land s \rightarrow_{\text{must}} s' \rightarrow_{\text{may}} s'') \rightarrow (\forall s'' \in \Sigma_E: s \rightarrow_{\text{may}} s'')\].

The inverse \(\subseteq^{-1}\) of the greatest partial bisimulation \(\subseteq\) on \(\mathcal{E}\) is \(\prec\), the greatest refinement on \(T[\mathcal{E}]\). The intuitionistic semantics for Hennessy-Milner logic in \([29]\) corresponds to the semantics \([\cdot]_{\text{int}}\) of that fragment of \(\text{Act} \mu\) on \(T[\mathcal{E}]\).

5 Abstracting data: Shape-based pointer analysis

An important form of pointer analysis is shape analysis \([6, 14, 22, 35, 40]\), 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 (“stack”) variables that point into the heap are drawn as arcs pointing to the nodes.

Figure 4 displays the syntax of such shape graphs. The example in the Figure depicts an approximation to a singly linked list of length at least two: Objects are circles; a double-circled object is a “summary node,” meaning that it possibly

---

\(^8\) We assume that all such structures are finitely branching.
represents more than one concrete object. Since the objects were constructed from a class/struct that owns a next field, objects have next-labeled arcs. For discussion, the objects are named $u_0$ and $u_1$, and local variables $x$ and $y$ point to the objects. A solid arc denotes that a field definitely points to an object; a dotted arc means the field possibly points to it. Thus, the self-arc on $u_1$ must be dotted because $u_1$ possibly denotes multiple nodes, meaning that a next dereference possibly points to one of the concrete objects denoted by the node.

Shape graphs can be encoded in various ways; in Figure 4, we display a coding due to Sagiv, Reps, and Wilhelm [35], who define local-variable points-to information with unary predicates and field points-to information with binary ones. The predicates produce the answers “necessarily points to” (1), “possibly points to” (1/2), and “not points to” (0), where the values are ordered $0 \leq 1/2 \leq 1$. The predicate, sm, notes which nodes are summary nodes.

Shape graphs can be used as data values for a data-flow analysis, where a program’s transfer functions transform an input shape graph to an output one. The transfer functions for assignment and object construction appear in Figure 5, where $p'$ denotes predicate $p$ updated by the transfer function, $T[C]$, for command C. The transfer functions are written as predicate-logic formulas, where conjunction is interpreted as meet; disjunction as join; and negation as strict complement. A data-flow analysis is assembled in the usual way $[8, 16, 23, 31, 32]$:

1. a control-flow graph is extracted from the source program, where transfer functions annotate the arcs of the graph;
2. a finite-height sup-semilattice of data-flow values (here, based on shape graphs) is defined, where values from the semilattice will be collected at the nodes (program points) of the control-flow graph.
3. a least fixed-point calculation is undertaken on the flow equations induced by the control-flow graph.
Step 2 is the most interesting, in that a single shape graph might be collected at each node ("independent attribute" analysis, e.g., [40]) or a set of shape graphs might be collected ("relational" analysis, e.g., [35]). In the former case, the join operation weakens solid arcs into dotted ones when dissimilar graphs are joined; in the latter case, a bounded set union (umdening [8]) operation is employed so that no infinite set of graphs is constructed.

**Modal shape graphs.** The dotted and solid arcs of shape graphs strongly suggest that modal transition systems lurk in the foundations, and so they do:

**Definition 10 (Modal shape graph).** A modal shape graph (MSG) is a Kripke MTS \( \mathcal{M} \equiv (\Sigma_M, \text{Act}, \text{AP}, \mu_{\text{must}}, \mu_{\text{may}}, L_{\text{must}}, L_{\text{may}}) \), where \( \text{AP} \) is a set of local variables along with the distinguished predicate, \( sm \), \( \text{Act} \) is a set of field names, and \( \Sigma_M \) is a set of heap objects.

Our modelling sets \( x \in L_{\text{must}}(s) \) when a solid arc shows that \( x \) points to object \( s \). Similarly, \( L_{\text{may}}(s) \) collects the names of local variables that possibly point to it. When \( s \) is a summary node, then \( sm \in L_{\text{must}}(s) \). Of course, \( \mu_{\text{must}} \) and \( \mu_{\text{may}} \) model the solid and dotted field-labeled arcs, respectively. It is easy to translate shape graphs, like that in Figure 4, into MTS format:

\[
\begin{align*}
\Sigma_M &= \{ u_0, u_1 \} & \text{Act} &= \{ \text{next} \} & \text{AP} &= \{ x, y, sm \} \\
\mu_{\text{must}} &= \{ (u_0, \text{next}, u_1) \} & \mu_{\text{may}} &= \{ (u_1, \text{next}, u_1) \} \\
L_{\text{must}}(u_0) &= \{ y \} & L_{\text{must}}(u_1) &= \{ x, sm \} \\
L_{\text{may}}(u_0) &= \emptyset & L_{\text{may}}(u_1) &= \emptyset
\end{align*}
\]

A concrete, run-time execution state is coded as a concrete modal shape graph.\(^9\)

**Transfer functions.** There is little challenge in writing the transfer functions for modal shape graphs: Given a modal shape graph \( \mathcal{M} \), let \( \mathcal{M}_{[C]} \) be the graph obtained from \( \mathcal{M} \) by executing command \( C \). We only specify those aspects of \( \mathcal{M}_{[C]} \) that are different from \( \mathcal{M} \):

1. \( x = y \): For all \( s \in \Sigma_M \), \( x \in L_{[C]}^{\text{must}}(s) \) iff \( y \in L_{\text{must}}(s) \); and \( x \in L_{[C]}^{\text{may}}(s) \) iff \( y \in L_{\text{may}}(s) \).
2. \( x = y.n \): For all \( s \in \Sigma_M \), \( x \in L_{[C]}^{\text{must}}(s) \) iff (\( \exists s' \in S \) \( y \in L_{\text{must}}(s') \)) and \( s' \xrightarrow{\mu_{\text{must}}} n s' \); and \( x \in L_{[C]}^{\text{may}}(s) \) iff (\( \exists s' \in S \) \( y \in L_{\text{may}}(s') \)) and \( s' \xrightarrow{\mu_{\text{may}}} n s' \).
3. \( x.n = y \): For all \( s \in \Sigma_M \), (i) if \( x \in L_{\text{must}}(s) \) and there is some \( s'' \) with \( s \xrightarrow{\mu_{\text{must}}} n s'' \), then, for all \( s' \in S \), \( y \in L_{[C]}^{\text{must}}(s') \) implies \( s \xrightarrow{\mu_{[C]}} n s' \); and (ii) if \( x \in L_{\text{may}}(s) \) and there is some \( s'' \) with \( s \xrightarrow{\mu_{\text{may}}} n s'' \), then, for all \( s' \in S \), \( y \in L_{[C]}^{\text{may}}(s) \) implies \( s \xrightarrow{\mu_{[C]}} n s' \). If \( sm \in L_{\text{must}}(y) \), then all transitions from \( y \) are preserved.
4. \( x = \text{new Node}() \): Creates a fresh object that is labeled with \( x \). Set \( \Sigma_{M_{[C]}} \equiv \Sigma_M \cup \{ s_{\text{new}} \} (s_{\text{new}} \not\in \Sigma_M) \); and for all \( s \in \Sigma_M \), \( x \not\in L_{[C]}^{\text{must}}(s) \), and \( x \in L_{[C]}^{\text{may}}(s_{\text{new}}) \).

\(^9\) Where every local variable, \( x \), belongs to at most one \( L_{\text{must}}^\text{(s)} \).
Properties as temporal formulas. Once a data-flow analysis builds a shape graph, we check the graph for correctness properties that are expressible in the CTL-subset \([5, 10]\) of the modal \(\mu\)-calculus. In \([35]\), such properties are encoded in predicate logic augmented with a transitive closure operator.

Here are examples: The direction relationship \([14]\), stating that an access path exists from the object named by \(x\) to an object named by \(y\), is written \(D(x, y) \triangleq T(x \land T(x, y))\) — an object, \(s\), has atomic property \(x\) iff \(x\) points to \(s\). Recall that \(T_{x} \phi\) states, “there exists a path of \(a\)-labeled transitions such that, at some state in the future, \(\phi\) holds true.” To validate that there is necessarily (possibly) a path from \(s\), we check if \(s \in D(x, y) \triangleright}\text{sec} (s \in D(x, y) \triangleright}\text{pos}); to refute existence of a path, we check \(s \not\in \sim D(x, y) \triangleright\text{sec}.

The interference relationship \([14]\), saying that pointers \(x\) and \(y\) have access paths to a common heap node, is written with inverse transition relationships of \(\mu\)-calculus: \(I(x, y) \triangleq (\exists T_{x}) \land (\exists T_{y})\). We check \(s \in I(x, y) \triangleright\text{pos} to see if aliasing of object \(s\) by \(x\) and \(y\) is possible.

Aliasing of pointers can be expressed: For aliasing \(\triangleq (\exists T_{x}) \land (\exists T_{y})\), the formulas \(a) A_{\text{next}} \rightarrow \neg \text{aliasing}\), \(b) A_{\text{next}} \rightarrow \neg (x \land \neg \text{aliasing})\), and \(c) A_{\text{next}} \rightarrow (x \land \text{aliasing})\) can then be used to check: (a) the absence of any kind of aliasing; (b) that \(x\) has no alias; and that (c) \(x\) and \(y\) never point to the same heap node. (Recall that \(A_{\text{next}} \phi\) states, “for all \(a\)-paths, it is globally true for all states along the path, that \(\phi\) holds.”)

We can check for possibly cyclic data structures. The predicate cyclic \(\triangleq \forall X_{\text{heap}} X \land EX_{\text{next}} \forall X_{\text{heap}}\) states that a heap node is pointed to by some \(X\) that has an access path to, presumably the same, heap node pointed to by \(X\). (Recall that \(EX_{\text{next}} \phi\) says, “there exists an \(a\)-transition to a next state where \(\phi\) holds.”)

Improving precision: The embedding theorem and focus operation. For improving the analysis's precision (e.g., “strong updates”), Sagiv, Wilhelm, and Reps employ a focus operation \([35]\), which refines a shape graph into a set of shape-graph variants, such that a predicate that formerly evaluated to \(1/2\) (“possibly holds”) in the original graph now evaluates to 0 or to 1 in every variant. The set of variants must be consistent and complete with regards to the original graph, and an Embedding Theorem is used to make this argument. A more precise, relational, data-flow analysis is the result.

Within the representation of modal shape graphs, the hypotheses of the Embedding Theorem ensure a refinement relation, and the consequences of the Theorem follow from the soundness of refinement (Theorem 1).

The focus operation itself defines a cover: A set of MTSs, \(\mathcal{S}_{A}\), covers an MTS, \(\mathcal{A}\), iff (i) for all \(K \in \mathcal{S}_{A}\), \(K \nsucceq_{\text{r}} A\); (ii) for every concrete MTS, \(C\), such that \(C \nsucceq_{\text{r}} A\), there exists some \(K \in \mathcal{S}_{A}\) such that \(C \nsucceq_{\text{r}} K\). Any property that necessarily holds true for all the MTSs in \(\mathcal{S}_{A}\) holds true for all concrete refinements of \(A\); dually, any property that possibly holds for any MTS in \(\mathcal{S}_{A}\) holds true for \(A\) thus, \(\mathcal{S}_{A}\) is consistent and complete regarding the concrete MTSs represented by \(A\).

The focus operation in \([35]\) generates one particular shape-graph cover by examining a may-transition reachable from a program variable and refining it to
a must-transition in one graph variant, removing it in another graph variant, and splitting the transition’s source or destination summary node in a third variant. Of course, there can exist other forms of focus operations; in all cases, a cover must be defined.

6 Conclusions

The case studies demonstrate how modal transition systems provide a foundation for development of analyses whose outcomes take the form, “necessarily” (yes), “possibly” (maybe), and “not possibly” (no). These applications go beyond the traditional use for MTSs (proving properties of loose specifications). In addition to the two examples in this paper, there are other program analyses that are neatly expressed via Kripke MTSs; two noteworthy ones are

- Whaley and Rinard’s points-to escape analysis [40], where multi-threaded programs are analyzed for object sharing. Shape graphs are generated, where solid arcs represent assignments made by the thread being analyzed, and dotted arcs represent assignments made by other threads executing in parallel. Jackson’s Z-like Alloy logic [21] is used to model check the graphs.
- Interprocedural data-flow analysis [34,36], where graphs are used to denote control flow. Those program transitions that must occur (e.g., intraprocedural transitions) are denoted by solid arcs; transitions that might occur (e.g., procedure call- and return-arcs, where the exact procedure invoked or the exact invocation point is uncertain) are denoted by may-arcs.

Other applications await discovery (e.g., cartesian abstraction in the SLAM project [1]), and the relationship of our semantic framework to earlier studies of 3-valued modal logic [13,30,37] and intuitionistic modal logic [38] deserves examination as well.

References