Abstraction for Falsification

Thomas Ball[1], Orna Kupferman[2], and Greta Yorsh[3]

[1] Microsoft Research, Redmond, WA, USA.
Email: tball@microsoft.com, URL: research.microsoft.com/∼tball
[2] Hebrew University, School of Eng. and Comp. Sci., Jerusalem 91904, Israel.
Email: orna@cs.huji.ac.il, URL: www.cs.huji.ac.il/∼orna
[3] Tel-Aviv University, School of Comp. Sci., Tel-Aviv 69978, Israel.
Email:gretay@post.tau.ac.il, URL: www.math.tau.ac.il/∼gretay

Microsoft Research Technical Report
MSR-TR-2005-50

Abstract. Abstraction is traditionally used in the process of *verification*. There, an abstraction of a concrete system is sound if properties of the abstract system also hold in the concrete system. Specifically, if an abstract state a satisfies a property ψ then *all* the concrete states that correspond to a satisfy ψ too. Since the ideal goal of proving a system correct involves many obstacles, the primary use of formal methods nowadays is *falsification*. There, as in *testing*, the goal is to detect errors, rather than to prove correctness. In the falsification setting, we can say that an abstraction is sound if errors of the abstract system exist also in the concrete system. Specifically, if an abstract state a violates a property ψ, then *there exists* a concrete state that corresponds to a and violates ψ too.
An abstraction that is sound for falsification need not be sound for verification. This suggests that existing frameworks for abstraction for verification may be too restrictive when used for falsification, and that a new framework is needed in order to take advantage of the weaker definition of soundness in the falsification setting.
We present such a framework, show that it is indeed stronger (than other abstraction frameworks designed for verification), demonstrate that it can be made even stronger by parameterizing its transitions by predicates, and describe how it can be used for falsification of branching-time and linear-time temporal properties, as well as for generating testing goals for a concrete system by reasoning about its abstraction.

1 Introduction

Automated abstraction is a powerful technique for reasoning about systems. An abstraction framework [CC77] consists of a concrete system with (large, possibly infinite) state space C, an abstract system with (smaller, often finite) state space A, and an abstraction function $\rho\colon C \to A$ that relates concrete and abstract states. An abstraction framework is sound with respect to a logic L if all properties specified in L that hold in an abstract state a also hold in all the concrete states that correspond to a. Formally, for all $a \in A$ and $\varphi \in L$, if a satisfies φ then for all $c \in C$ with $\rho(c, a)$, we have that c satisfies φ. The soundness of the abstraction framework enables the user to verify properties of the abstract system using techniques such as model checking [CE81,QS81] and conclude their validity in the concrete system.

While the ultimate goal of formal verification is to prove that a system satisfies some specification, there are many obstacles to achieving this ideal in practice. Thus, the primary use of formal methods nowadays is *falsification*, where the goal is to detect errors rather than to provide a proof of correctness. This is reflected in the extensive research done on bounded model checking (c.f., [FKZ$^+$00]), runtime verification (c.f., [Sip99]), property testing (c.f., [CK02]), etc. In the falsification setting, we can say that an abstraction is sound with respect to a logic L if all errors specified in L that hold in an abstract state a also hold in some concrete state that corresponds to a. Formally, for all $a \in A$ and $\varphi \in L$, if a satisfies φ then there is $c \in C$ such that $\rho(c, a)$ and c satisfies φ. [4]

Since every abstract state corresponds to at least one concrete state, the soundness condition in the falsification setting is weaker than the soundness condition in the verification setting. To see that this weaker definition is sufficiently strong for falsification, note that the concrete state c that satisfies φ witnesses that the concrete system is erroneous (we note that in the falsification setting φ is a "bad" property that we don't wish the system to have, while in the verifications setting φ is a "good" property that we wish the system to have).

We develop a new abstraction framework to take advantage of the weaker definition of soundness in the falsification setting. Our framework is based on *modal transition systems* (MTS) [LT88]. Traditional MTS have two types of transitions: may (over-approximating transitions) and $must$ (under-approximating transitions). The use of $must$ transitions in the falsification setting was explored in [PDV01,GLST05], with different motivations. Our framework contains, in addition, a new type of transition, which can be viewed as the reverse version of $must$ transitions [Bal04]. Accordingly, we refer to transitions of this type as $must^-$ transitions and refer to the traditional $must$ transitions as $must^+$ transitions. While a $must^+$ transition from an abstract state a to an abstract state a' implies that for all concrete states c with $\rho(c, a)$ there is a successor concrete state c' with $\rho(c', a')$, a $must^-$ transition from a to a' implies that for all concrete states c' with $\rho(c', a')$ there is a concrete predecessor state c with $\rho(c, a)$. The $must^-$ transitions correspond to the weaker soundness requirement in the falsification setting and are incomparable to $must^+$ transitions.

Consider, for example, a simple concrete system consisting of the assignment statement `x:=x-3`. Suppose that the abstract system is formed via predicate abstraction using the predicate $x > 6$. Consider the abstract transition $\{x > 6\}$ `x:=x-3` $\{x > 6\}$. This transition is not a $must$ transition, as there are pre-states satisfying $x > 6$ (namely $x = 7$, $x = 8$, and $x = 9$) for which the assignment statement results in a post-state that does not satisfy $x > 6$. In a traditional MTS, this transition is a may transition. However, in an MTS with $must^-$ transitions, the above transition is a $must^-$ transition, as for every post-state c' satisfying $x > 6$ there is a pre-state c satisfying $x > 6$ such that the execution of `x:=x-3` from c yields c'. It is impossible to make this inference in a traditional MTS, even those augmented with hyper-must transitions [LX90,SG04]. As we shall see below, the observation that the abstract transition is a $must^-$ transition rather than a may transition enables better reasoning about the concrete system.

We study MTS with these three types of transitions, which we refer to as *ternary modal transition systems* (TMTS)[5]. We first show that the TMTS model is indeed

[4] Note that the falsification setting is different than the problem of *generalized model checking* [GJ02]. There, the existential quantifier ranges over all possible concrete systems and the problem is one of satisfiability (does there exist a concrete system with the same property as the abstract system?). Here, the concrete system is given and we only replace the universal quantification on concrete states that correspond to a by an existential quantification on them.

[5] Not to be confused with the three-valued logic sometimes used in these systems.

stronger than the MTS model: while MTS with only may and $must^+$ transitions are logically characterized by a 3-valued modal logic with the AX and EX (for all successors/exists a successor) operators, TMTS are logically characterized by a strictly more expressive modal logic which has, in addition, the AY and EY (for all predecessors/exists a predecessor) past operators. We then show that by replacing $must^+$ transitions by $must^-$ transitions, existing work on abstraction/refinement for verification [GHJ01,SG03,BG04,SG04,DN05] can be lifted to abstraction/refinement for falsification.

In particular, this immediately provides a framework for falsification of CTL and μ-calculus specifications. Going back to our example, by letting existential quantification range over $must^-$ transitions, we can conclude from the fact that the abstract system satisfies the property $EXx > 6$ (there is a successor in which $x > 6$ is valid) that some concrete state also satisfies $EXx > 6$. Note that such reasoning cannot be done in a traditional MTS, as there the $must^-$ transition is overapproximated by a may transition, which is not helpful for reasoning about existential properties. Thus, there are cases where evaluation of a formula on a traditional MTS returns $\perp$ (nothing can be concluded for the concrete system, and refinement is needed) and its evaluation on a TMTS returns an *existential* **true** or *existential* **false**.

Formally, we describe a *6-valued semantics* for TMTS. In addition to the **T** (all corresponding concrete states satisfy the formula), **F** (all corresponding concrete states violate the formula), and $\perp$ truth values that the 3-valued semantics for MTS has, the 6-valued semantics also has the $\mathbf{T}_\exists$ (there is a corresponding concrete state that satisfies the formula), $\mathbf{F}_\exists$ (there is a corresponding concrete state that violates the formula), and M (mixed – both $\mathbf{T}_\exists$ and $\mathbf{F}_\exists$ hold) truth values.

The combination of $must^+$ and $must^-$ transitions turn out to be especially powerful when reasoning about *weak reachability*, which is useful for abstraction-guided test generation [Bal04] and falsification of linear-time properties. As discussed in [Bal04], if there is a sequence of $must^-$ transitions from a_0 to a_j followed by a sequence of $must^+$ transitions from a_j to a_k, then there are guaranteed to be concrete states c_0 and c_k (corresponding to a_0 and a_k) such that c_k is reachable from c_0 in the concrete system (in which case we say that a_k is weakly reachable from a_0). In this case, we can conclude that it is possible to cover the abstract state a_k via testing. When the abstraction is the product of an abstract system with a nondeterministic Büchi automaton accepting all the faults of the system, weak reachability can be used in order to detect faults in the concrete system. We focus on abstractions obtained from programs by predicate abstraction, and study the problem of composing transitions in an TMTS in a way that guarantees weak reachability. We suggest a method where $must^+$ and $must^-$ transitions are parameterized with predicates, automatically induced by the weakest preconditions and the strongest postconditions of the statements in the program[6].

We distinguish between the case of composing the last $must^-$ transition in the sequence with the first $must^+$ transition (there, we show that the most effective predicates to work with are the strongest postcondition and the weakest precondition of the $must^-$ and the $must^+$ transitions, respectively, and these can be generated automatically), and

[6] We note (see the remark at the end of Section 3 for a detailed discussion) that our approach is different than refining the TMTS as the predicates we use are local to the transitions.

the case of composing two $must^-$ or $must^+$ transitions, where each transition is parameterized by two predicates, and reasoning is done in an assume-guarantee fashion. We show that our method is sound and complete for reasoning about weak reachability. We show how the new framework can be used for falsification of LTL properties, as well as for generating testing goals for the concrete system by reasoning about its abstraction.

The paper is organized as follows. Section 2 formally presents ternary modal transition systems (TMTS), how they abstract concrete systems (as well as each other) and characterizes their abstraction pre-order via the full propositional modal logic (full-PML). Section 2.3 presents the 6-valued semantics for TMTS and demonstrates that TMTS are more precise for falsification than traditional MTS. We also show that falsification can be lifted to the μ-calculus as well as linear-time logics. Section 3 shows that weak reachability can be made more precise by parameterizing both $must^+$ and $must^-$ transitions via predicates. Section 4 describes how to use TMTS to falsify LTL properties in model checking and to generate testing goals. Section 5 concludes the paper.

2 The Abstraction Framework

In this section we describe our abstraction framework. We define TMTS — ternary modal transition systems, which extend modal transition systems by a third type of transition, and study their theoretical aspects.

2.1 Ternary modal transition systems

A *concrete transition system* is a tuple $C = \langle AP, S_C, I_C, \longrightarrow_C, L_C\rangle$, where AP is a finite set of atomic propositions, S_C is a (possibly infinite) set of states, $I_C \subseteq S_C$ is a set of initial states, $\longrightarrow_C \subseteq S_C \times S_C$ is a transition relation and $L_C\colon S_C \times AP \mapsto \{\mathbf{T}, \mathbf{F}\}$ is a labeling function that maps each state and atomic proposition to the truth value of the proposition in the state.[7]

An abstraction of C is a partially defined system. Incompleteness involves both the value of the atomic propositions, which can now take the value $\bot$ (unknown), and the transition relation, which is approximated by over- and/or under-approximating transitions. Several frameworks are defined in the literature (c.f. [LT88,BG99,HJS01]). We define here a new framework, which consists of *ternary transition systems* (TMTS, for short). Unlike the traditional MTS, our TMTS has two types of under-approximating transitions. Formally, we have the following.

A TMTS is a tuple $A = \langle AP, S_A, I_A, \overset{may}{\longrightarrow}_A, \overset{must^+}{\longrightarrow}_A, \overset{must^-}{\longrightarrow}_A, L_A\rangle$, where AP is a finite set of atomic propositions, S_A is a finite set of abstract states, $I_A \subseteq S_A$ is a set of initial states, the transition relations $\overset{may}{\longrightarrow}_A$, $\overset{must^+}{\longrightarrow}_A$, and $\overset{must^-}{\longrightarrow}_A$ are subsets of $S_A \times S_A$ satisfying $\overset{must^+}{\longrightarrow}_A \subseteq \overset{may}{\longrightarrow}_A$ and $\overset{must^-}{\longrightarrow}_A \subseteq \overset{may}{\longrightarrow}_A$, and $L_A\colon S_A \times AP \rightarrow \{\mathbf{T}, \mathbf{F}, \bot\}$ is a labeling function that maps each state and atomic proposition to the truth value (possibly unknown) of the proposition in the state. When A is clear from the context we sometimes

[7] We use **T** and **F** to denote the truth values **true** and **false** of the standard (verification) semantics, and introduce additional truth values in Section 2.3.

use $may(a,a')$, $must^+(a,a')$, and $must^-(a,a')$ instead of $a \xrightarrow{may}_A a$, $a \xrightarrow{must^+}_A a'$, and $a \xrightarrow{must^-}_A a'$, respectively.

The elements of $\{\mathbf{T},\mathbf{F},\bot\}$ can be arranged in an "information lattice" [Kle87] in which $\bot \sqsubseteq \mathbf{T}$ and $\bot \sqsubseteq \mathbf{F}$. We say that a concrete state c *satisfies* an abstract state a if for all $p \in AP$, we have $L_A(a,p) \sqsubseteq L_C(c,p)$ (equivalently, if $L_A(a,p) \neq \bot$ then $L_C(c,p) = L_A(a,p)$).

Let $C = \langle AP, S_C, I_C, \longrightarrow_C, L_C\rangle$ be a concrete transition system. A TMTS $A = \langle AP, S_A, I_A, \xrightarrow{may}_A, \xrightarrow{must^+}_A, \xrightarrow{must^-}_A, L_A\rangle$ is an *abstraction* of C if there exists a total and onto function $\rho\colon S_C \to S_A$ such that (i) for all $c \in S_C$, we have that c satisfies $\rho(c)$, and (ii) the transition relations $\xrightarrow{may}_A$, $\xrightarrow{must^+}_A$, and $\xrightarrow{must^-}_A$ satisfy the following:

- $a \xrightarrow{may}_A a'$ if there is a concrete state c with $\rho(c) = a$, there is a concrete state c' with $\rho(c') = a'$, and $c \longrightarrow_C c'$.
- $a \xrightarrow{must^+}_A a'$ only if for every concrete state c with $\rho(c) = a$, there is a concrete state c' with $\rho(c') = a'$ and $c \longrightarrow_C c'$.
- $a \xrightarrow{must^-}_A a'$ only if for every concrete state c' with $\rho(c') = a'$, there is a concrete state c with $\rho(c) = a$ and $c \longrightarrow_C c'$.

Note that may transitions over-approximate the concrete transitions. In particular, the abstract system can contain may transitions for which there is no corresponding concrete transition. Dually, $must^-$ and $must^+$ transitions under-approximate the concrete transitions. Thus, the concrete transition relation can contain transitions for which there are no corresponding $must$ transitions. Since ρ is onto, each abstract state corresponds to at least one concrete state, and so $\xrightarrow{must^+}_A \subseteq \xrightarrow{may}_A$ and $\xrightarrow{must^-}_A \subseteq \xrightarrow{may}_A$. On the other hand, $\xrightarrow{must^+}_A$ and $\xrightarrow{must^-}_A$ are incomparable. Finally, note that by letting $must$-transitions become may-transitions, and by adding superfluous may-transitions, we can have several abstractions of the same concrete system.

A *precision preorder* on TMTS defines when one TMTS is more abstract than another. For two TMTS $A = \langle AP, S_A, I_A, \xrightarrow{may}_A, \xrightarrow{must^+}_A, \xrightarrow{must^-}_A, L_A\rangle$ and $B = \langle AP, S_B, I_B, \xrightarrow{may}_B, \xrightarrow{must^+}_B, \xrightarrow{must^-}_B, L_B\rangle$, the precision preorder is the greatest relation $\mathcal{H} \subseteq S_A \times S_B$ such that if $\mathcal{H}(a,b)$ then

C0. for all $p \in AP$, we have $L_A(a,p) \sqsubseteq L_B(b,p)$,
C1. if $b \xrightarrow{may}_B b'$, then there is $a' \in S_A$ such that $\mathcal{H}(a',b')$ and $a \xrightarrow{may}_A a'$,
C2. if $b' \xrightarrow{may}_B b$, then there is $a' \in S_A$ such that $\mathcal{H}(a',b')$ and $a' \xrightarrow{may}_A a$,
C3. if $a \xrightarrow{must^+}_A a'$, then there is $b' \in S_B$ such that $\mathcal{H}(a',b')$ and $b \xrightarrow{must^+}_B b'$, and
C4. if $a' \xrightarrow{must^-}_A a$, then there is $b' \in S_B$ such that $\mathcal{H}(a',b')$ and $b' \xrightarrow{must^-}_B b$.

When $\mathcal{H}(a,b)$, we write $(A,a) \preceq (B,b)$, which indicates that A is more abstract (less defined) than B.

By viewing a concrete system as an abstract system whose may, $must^+$, and $must^-$ transition relations are equivalent to the transition relation of the concrete system, we can use the precision preorder to relate a concrete system and its abstraction. Formally, the precision preorder $\mathcal{H} \subseteq S_C \times S_A$ (also known as *mixed simulation* [DGG97,GJ02]) is such that $\mathcal{H}(c,a)$ iff $\rho(c) = a$.

2.2 A logical characterization

The logic *full-PML* is a propositional logic extended with the modal operators AX ("for all immediate successors") and AY ("for all immediate predecessors"). Thus, full-PML extends PML [Ben91] by the past-time operator AY. The syntax of full-PML is given by the rules $\theta ::= p \mid \neg\theta \mid \theta \wedge \theta \mid AX\theta \mid AY\theta$, for $p \in AP$.

We define a 3-*valued semantics* of full-PML formulas with respect to TMTS. The value of a formula θ in a state a of a TMTS $A = \langle S_A, I_A, \xrightarrow{may}_A, \xrightarrow{must^+}_A, \xrightarrow{must^-}_A, L_A\rangle$, denoted $[(A,a) \models \theta]$, is defined as follows:

$$[(A,a) \models p] = L_A(a,p).$$

$$[(A,a) \models \neg\theta] = \begin{cases} \mathbf{T} & \text{if } [(A,a) \models \theta] = \mathbf{F}. \\ \mathbf{F} & \text{if } [(A,a) \models \theta] = \mathbf{T}. \\ \bot & \text{otherwise.} \end{cases}$$

$$[(A,a) \models \theta_1 \wedge \theta_2] = \begin{cases} \mathbf{T} & \text{if } [(A,a) \models \theta_1] = \mathbf{T} \text{ and } [(A,a) \models \theta_2] = \mathbf{T}. \\ \mathbf{F} & \text{if } [(A,a) \models \theta_1] = \mathbf{F} \text{ or } [(A,a) \models \theta_2] = \mathbf{F}. \\ \bot & \text{otherwise.} \end{cases}$$

$$[(A,a) \models AX\theta] = \begin{cases} \mathbf{T} & \text{if for all } a',\ \text{if } may(a,a') \text{ then } [(A,a') \models \theta] = \mathbf{T}. \\ \mathbf{F} & \text{if exists } a' \text{ s.t. } must^+(a,a') \text{ and } [(A,a') \models \theta] = \mathbf{F}. \\ \bot & \text{otherwise.} \end{cases}$$

$$[(A,a) \models AY\theta] = \begin{cases} \mathbf{T} & \text{if for all } a',\ \text{if } may(a',a) \text{ then } [(A,a') \models \theta] = \mathbf{T}. \\ \mathbf{F} & \text{if exists } a' \text{ s.t. } must^-(a',a) \text{ and } [(A,a') \models \theta] = \mathbf{F}. \\ \bot & \text{otherwise.} \end{cases}$$

While PML logically characterizes the precision preorder on MTS [GJ02], full-PML characterizes the precision preorder on TMTS. It follows that the TMTS model is indeed stronger than the MTS model, because TMTS are logically characterized by a strictly more expressive modal logic which has the past operators AY and EY, in addition to AX and EX operators. Formally, we have the following.

Theorem 1. *Let $A = \langle AP, S_A, I_A, \xrightarrow{may}_A, \xrightarrow{must^+}_A, \xrightarrow{must^-}_A, L_A\rangle$ and $B = \langle AP, S_B, I_B, \xrightarrow{may}_B, \xrightarrow{must^+}_B, \xrightarrow{must^-}_B, L_B\rangle$ be two TMTS. For every two states $a \in S_A$ and $b \in S_B$, we have that $(A,a) \preceq (B,b)$ iff $[(A,a) \models \theta] \sqsubseteq [(B,b) \models \theta]$ for all full-PML formulas θ.*

Proof. Assume first that $(A,a) \preceq (B,b)$. We prove that $[(A,a) \models \theta] \sqsubseteq [(B,b) \models \theta]$ for all full-PML formulas θ. The proof proceeds by induction on the structure of θ. Let $\mathcal{H} \subseteq S_A \times S_B$ be the precision preorder such that $\mathcal{H}(a,b)$.

- For the induction base, consider the case $\theta = p$ for $p \in AP$. Since $(A,a) \preceq (B,b)$, then, by the definition of $\preceq$, we have that $L_A(a,p) \sqsubseteq L_B(b,p)$, so, by the semantics of full-PML, $[(A,a) \models p] \sqsubseteq [(B,b) \models p]$.
- The cases $\theta = \neg\theta_1$ or $\theta = \theta_1 \wedge \theta_2$ follows immediately from the semantics of full-PML and the induction hypothesis.
- Let $\theta = AX\theta_1$. We prove that if $[(A,a) \models AX\theta_1] = \mathbf{T}$, then $[(B,b) \models AX\theta_1] = \mathbf{T}$ too, and if $[(A,a) \models AX\theta_1] = \mathbf{F}$ then $[(B,b) \models AX\theta_1] = \mathbf{F}$ too. Assume first that $[(A,a) \models AX\theta_1] = \mathbf{T}$. By the semantics of full-PML, for all a' with $a \xrightarrow{may}_A a'$, we have $[(A,a') \models \theta_1] = \mathbf{T}$. Since $(A,a) \preceq (B,b)$, then, by the definition of $\preceq$, if $b \xrightarrow{may}_B b'$, then there is $a' \in S_A$ such that $\mathcal{H}(a',b')$ and $a \xrightarrow{may}_A a'$. By the

induction hypothesis, $[(B, b') \models \theta_1] = \mathbf{T}$, thus $[(B, b) \models AX\theta_1] = \mathbf{T}$ and we are done. Assume now that $[(A, a) \models AX\theta_1] = \mathbf{F}$. By the semantics of full-PML, there exists a' such that $a \xrightarrow{must^+}_A a'$ and $[(A, a') \models \theta_1] = \mathbf{F}$. Since $(A, a) \preceq (B, b)$, then, by the definition of $\preceq$, there is $b' \in S_B$ such that $\mathcal{H}(a', b')$ and $b \xrightarrow{must^+}_B b'$. By the induction hypothesis, $[(B, b') \models \theta_1] = \mathbf{F}$, thus $[(B, b) \models AX\theta_1] = \mathbf{F}$ and we are done.

– Let $\theta = AY\theta_1$. We prove that if $[(A, a) \models AY\theta_1] = \mathbf{T}$, then $[(B, b) \models AY\theta_1] = \mathbf{T}$ too, and if $[(A, a) \models AY\theta_1] = \mathbf{F}$ then $[(B, b) \models AY\theta_1] = \mathbf{F}$ too. Assume first that $[(A, a) \models AY\theta_1] = \mathbf{T}$. By the semantics of full-PML, for all a' with $a' \xrightarrow{may}_A a$, we have $[(A, a') \models \theta_1] = \mathbf{T}$. Since $(A, a) \preceq (B, b)$, then, by the definition of $\preceq$, if $b' \xrightarrow{may}_B b$, then there is $a' \in S_A$ such that $\mathcal{H}(a', b')$ and $a' \xrightarrow{may}_A a$. By the induction hypothesis, $[(B, b') \models \theta_1] = \mathbf{T}$, thus $[(B, b) \models AY\theta_1] = \mathbf{T}$ and we are done. Assume now that $[(A, a) \models AY\theta_1] = \mathbf{F}$. By the semantics of full-PML, there exists a' such that $a' \xrightarrow{must^-}_A a$ and $[(A, a') \models \theta_1] = \mathbf{F}$. Since $(A, a) \preceq (B, b)$, then, by the definition of $\preceq$, there is $b' \in S_B$ such that $\mathcal{H}(a', b')$ and $b' \xrightarrow{must^-}_B b$. By the induction hypothesis, $[(B, b') \models \theta_1] = \mathbf{F}$, thus $[(B, b) \models AY\theta_1] = \mathbf{F}$ and we are done.

For the other direction we prove that if $(A, a) \not\preceq (B, b)$, then there is a full-PML formula θ such that $[(A, a) \models \theta] \not\sqsubseteq [(B, b) \models \theta]$. The precision preorder is the greatest fixed-point of the following sequence of relations $\mathcal{H}_i \subseteq S_A \times S_B$:

– $\mathcal{H}_0 = \{\langle a, b \rangle :$ for all $p \in AP$, we have $L_A(a, p) \sqsubseteq L_B(b, p)\}$,
– $\mathcal{H}_{i+1} = \mathcal{H}_i \cap \{\langle a, b \rangle : \langle a, b \rangle$ is good with respect to $\mathcal{H}_i\}$, where a pair $\langle a, b \rangle$ is *good* with respect to a relation $\mathcal{H}_i \subseteq S_A \times S_B$ iff the four conditions **C1**-**C4** of a precision relation hold with respect to $\mathcal{H}_i$.

Assume that $(A, a) \not\preceq (B, b)$. Then, there is an index $i \geq 0$ such that $\langle a, b \rangle \notin \mathcal{H}_i$. We define the full-PML formula θ by induction on i.

– If $i = 0$, thus $\langle a, b \rangle \notin \mathcal{H}_0$, then there is $p \in AP$ for which $L_A(a, p) \not\sqsubseteq L_B(b, p)$, and we define θ to be p.
– Assume that $\langle a, b \rangle \in \mathcal{H}_i$ and $\langle a, b \rangle \notin \mathcal{H}_{i+1}$. Then, $\langle a, b \rangle$ is not good with respect to $\mathcal{H}_i$. Therefore, at least one of the four conditions **C1** - **C4** does not hold. We consider each of these cases and show that in all of them, we can point to a full-PML formula θ such that $[(A, a) \models \theta] \not\sqsubseteq [(B, b) \models \theta]$.

  Assume that **C1** does not hold. Thus, there is a state $b' \in S_B$ such that $b \xrightarrow{may}_B b'$ and there is no $a' \in S_A$ such that $\mathcal{H}(a', b')$ and $a \xrightarrow{may}_A a'$. Thus, for all $a' \in S_A$, if $a \xrightarrow{may}_A a'$, then $\langle a', b' \rangle \notin \mathcal{H}_i$. By the induction hypothesis, there is a full-PML formula $\varphi_i(a', b')$ such that $[(A, a') \models \varphi_i(a', b')] \not\sqsubseteq [(B, b') \models \varphi_i(a', b')]$. By the definition of the $\sqsubseteq$ order, $[(A, a') \models \varphi_i(a', b')]$ is either $\mathbf{T}$ or $\mathbf{F}$. Let $\varphi'_i(a', b')$ be $\varphi_i(a', b')$ in case $[(A, a') \models \varphi_i(a', b')] = \mathbf{T}$ and be $\neg\varphi_i(a', b')$ in case $[(A, a') \models \varphi_i(a', b')] = \mathbf{F}$. We define $\theta = AX \bigvee_{a' : a \xrightarrow{may}_A a'} \varphi'_i(a', b')$. It is not hard to see that $[(A, a) \models \theta] = \mathbf{T}$. Indeed, by the semantics of the AX operator, $[(A, a) \models \theta] = \mathbf{T}$ if for all a' with $may(a, a')$, we have that $[(A, a') \models \varphi'_i(a', b')] = \mathbf{T}$, which we have established by construction. Also, by the definition of $\varphi'_i(a', b')$, we have that

$[(B, b') \models \varphi'_i(a', b')] \neq \mathbf{T}$ for all a' with $a \xrightarrow{may}_A a'$. Hence, $[(A, a) \models \theta] \not\sqsubseteq [(B, b) \models \theta]$.

Assume that **C2** does not hold. Thus, there is a state $b' \in S_B$ such that $b' \xrightarrow{may}_B b$ and there is no $a' \in S_A$ such that $\mathcal{H}(a', b')$ and $a' \xrightarrow{may}_A a$. Thus, for all $a' \in S_A$, if $a' \xrightarrow{may}_A a$, then $\langle a', b' \rangle \notin \mathcal{H}_i$. Let $\varphi'_i(a', b')$ be as in the **C1** case. We define $\theta = AY \bigvee_{a' : a' \xrightarrow{may}_A a} \varphi'_i(a', b')$. It is not hard to see that $[(A, a) \models \theta] = \mathbf{T}$. Indeed, by the semantics of the AY operator, $[(A, a) \models \theta] = \mathbf{T}$ if for all a' with $may(a', a)$, we have that $[(A, a') \models \varphi'_i(a', b')] = \mathbf{T}$, which we have established by construction. Also, by the definition of $\varphi'_i(a', b')$, we have that $[(B, b') \models \varphi'_i(a', b')] \neq \mathbf{T}$ for all a' with $a' \xrightarrow{may}_A a$. Hence, $[(A, a) \models \theta] \not\sqsubseteq [(B, b) \models \theta]$.

Assume that **C3** does not hold. Thus, there is a state $a' \in S_A$ such that $a \xrightarrow{must^+}_A a'$ and there is no $b' \in S_B$ such that $\mathcal{H}(a', b')$ and $b \xrightarrow{must^+}_B b'$. Thus, for all $b' \in S_B$, if $b \xrightarrow{must^+}_B b'$, then $\langle a', b' \rangle \notin \mathcal{H}_i$. Let $\varphi'_i(a', b')$ be as in the **C1** case. We define $\theta = EX \bigwedge_{b' : b \xrightarrow{must^+}_B b'} \varphi'_i(a', b')$. By the semantics of the EX operator, $[(A, a) \models \theta] = \mathbf{T}$, as a' satisfies $must^+(a, a')$ and $[(A, a') \models \varphi'_i(a', b')] = \mathbf{T}$ for all b' with $b \xrightarrow{must^+}_B b'$. Also, by the definition of $\varphi'_i(a', b')$, we have that $[(B, b') \models \varphi'_i(a', b')] \neq \mathbf{T}$ for all b' with $b \xrightarrow{must^+}_B b'$. Hence, $[(A, a) \models \theta] \not\sqsubseteq [(B, b) \models \theta]$.

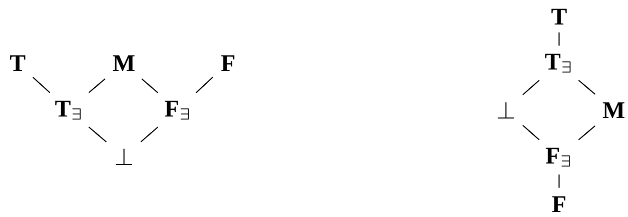
Assume that **C4** does not hold. Thus, there is a state $a' \in S_A$ such that $a' \xrightarrow{must^-}_A a$ and there is no $b' \in S_B$ such that $\mathcal{H}(a', b')$ and $b' \xrightarrow{must^-}_B b$. Thus, for all $b' \in S_B$, if $b' \xrightarrow{must^-}_B b$, then $\langle a', b' \rangle \notin \mathcal{H}_i$. Let $\varphi'_i(a', b')$ be as in the **C1** case. We define $\theta = EY \bigwedge_{b' : b' \xrightarrow{must^-}_B b} \varphi'_i(a', b')$. By the semantics of the EY operator, $[(A, a) \models \theta] = \mathbf{T}$ as a' satisfies $must^-(a', a)$ and $[(A, a') \models \varphi'_i(a', b')] = \mathbf{T}$ for all b' with $b' \xrightarrow{must^-}_B b$. Also, by the definition of $\varphi'_i(a', b')$, we have that $[(B, b') \models \varphi'_i(a', b')] \neq \mathbf{T}$ for all b' with $b' \xrightarrow{must^-}_B b$. Hence, $[(A, a) \models \theta] \not\sqsubseteq [(B, b) \models \theta]$.

2.3 Falsification Using TMTS

As shown in Section 2.2, the backwards nature of $must^-$ transitions makes them suitable for reasoning about the past. Thus, TMTS can be helpful in the verification setting for reasoning about specifications in full μ-calculus and other specification formalisms that contain past operators. We view this as a minor advantage of TMTS. In this section, we study their significant advantage: reasoning about specifications in a falsification setting[8].

Recall that the truth value **T** for a formula φ in an abstract state a indicates that all concrete states corresponding to a satisfy φ, and dually for **F**. A more informative truth value could have counted the number of corresponding concrete states that satisfy or violate φ. We could have had, for example, $[(A, a) \models \varphi] = \langle count\_T, count\_F \rangle$, for

[8] The specifications may contain both future and past operators. For simplicity, we describe the framework here for the μ-calculus, which does not contain past modalities. By letting the AY modality range over $must^+$ transitions, the framework can be used for falsification of full μ-calculus specifications.



information lattice   truth lattice

Fig. 1. The information and truth lattices for $\mathcal{L}_6$.

$count\_T, count\_F \in \{0, \ldots, |\rho^{-1}(a)|\}$, which indicates in how many corresponding concrete states we know that φ is satisfied and in how many we know it is violated. Such an informative reasoning, however, would lose much of the saving that the transition to an abstract system achieves. We suggest, instead, to approximate $count\_T$ and $count\_F$ by 0, at least 1, or $|\rho^{-1}(a)|$.

In addition to the truth values **T**, **F**, and $\bot$, we now allow formulas to have the values $\mathbf{T}_\exists$ (existential **true**), $\mathbf{F}_\exists$ (existential **false**), and **M** ("mixed" – both $\mathbf{T}_\exists$ and $\mathbf{F}_\exists$). Intuitively, the values $\mathbf{T}_\exists$, $\mathbf{F}_\exists$, and **M** refine the value $\bot$, and are helpful for falsification and testing, as they indicate that the abstract state corresponds to at least one concrete state that satisfies the property ($\mathbf{T}_\exists$), at least one concrete state that violates the property ($\mathbf{F}_\exists$), and at least one pair of concrete states in which one state satisfies the property and one violates it (**M**).

As described in Figure 1, the six values $\mathcal{L}_6 = \{\mathbf{T}, \mathbf{F}, \mathbf{M}, \mathbf{T}_\exists, \mathbf{F}_\exists, \bot\}$ can be ordered in the information lattice depicted on the left. The values can also be ordered in the "truth lattice" depicted on the right:

We allow the truth values of the (abstract) labeling function L_A to range over the six truth values.

A TMTS $A = \langle AP, S_A, I_A, \xrightarrow{may}_A, \xrightarrow{must^+}_A, \xrightarrow{must^-}_A, L_A\rangle$ is an abstraction of a concrete transition system $C = \langle AP, S_C, I_C, \longrightarrow_C, L_C\rangle$ if there exists a total and onto function $\rho\colon S_C \to S_A$ such that for all $a \in S_A$ and $p \in AP$:

- $L_A(a,p) = \mathbf{T}$ only if for all $c \in S_C$ such that $\rho(c) = a$, we have $L_C(c,p) = \mathbf{T}$;
- $L_A(a,p) = \mathbf{F}$ only if for all $c \in S_C$ such that $\rho(c) = a$, we have $L_C(c,p) = \mathbf{F}$;
- $L_A(a,p) = \mathbf{T}_\exists$ only if there exists $c \in S_C$ such that $\rho(c) = a$ and $L_C(c,p) = \mathbf{T}$;
- $L_A(a,p) = \mathbf{F}_\exists$ only if there exists $c \in S_C$ such that $\rho(c) = a$ and $L_C(c,p) = \mathbf{F}$;
- $L_A(a,p) = \mathbf{M}$ only if there exist $c, c' \in S_C$ such that $\rho(c) = \rho(c') = a$, $L_C(c,p) = \mathbf{T}$, and $L_C(c',p) = \mathbf{F}$.

In addition, ρ satisfies requirement (ii) relating the transitions of A and C, as defined in Section 2.1.

The complementation ($\neg\colon \mathcal{L}_6 \to \mathcal{L}_6$) and the conjunction ($\wedge\colon \mathcal{L}_6 \times \mathcal{L}_6 \to \mathcal{L}_6$) operations are defined as follows:

	¬
F	**T**
$\mathbf{F}_\exists$	$\mathbf{T}_\exists$
M	**M**
$\mathbf{T}_\exists$	$\mathbf{F}_\exists$
T	**F**
$\bot$	$\bot$

∧	**F**	$\mathbf{F}_\exists$	**M**	$\mathbf{T}_\exists$	**T**	$\bot$
F	**F**	**F**	**F**	**F**	**F**	**F**
$\mathbf{F}_\exists$	**F**	$\mathbf{F}_\exists$	$\mathbf{F}_\exists$	$\mathbf{F}_\exists$	$\mathbf{F}_\exists$	$\mathbf{F}_\exists$
M	**F**	$\mathbf{F}_\exists$	$\mathbf{F}_\exists$	$\mathbf{F}_\exists$	**M**	$\mathbf{F}_\exists$
$\mathbf{T}_\exists$	**F**	$\mathbf{F}_\exists$	$\mathbf{F}_\exists$	$\bot$	$\mathbf{T}_\exists$	$\bot$
T	**F**	$\mathbf{F}_\exists$	**M**	$\mathbf{T}_\exists$	**T**	$\bot$
$\bot$	**F**	$\mathbf{F}_\exists$	$\mathbf{F}_\exists$	$\bot$	$\bot$	$\bot$

We define a 6-*valued semantics* of PML formulas with respect to TMTS. The value of a formula θ in a state a of a TMTS $A = \langle AP, S_A, I_A, \xrightarrow{may}_A, \xrightarrow{must^+}_A, \xrightarrow{must^-}_A, L_A\rangle$, denoted $[(A,a) \models \theta]$, is a value in $\mathcal{L}_6$, and is defined, by induction on the structure of θ, as follows:

$[(A,a) \models p] = L_A(a,p)$.
$[(A,a) \models \neg\theta] = \neg([(A,a) \models \theta])$.
$[(A,a) \models \theta_1 \wedge \theta_2] = \wedge([(A,a') \models \theta_1], [(A,a') \models \theta_2])$.

$$[(A,a) \models AX\theta] = \begin{cases} \mathbf{T} & \text{if for all } a',\ \text{if } may(a,a') \text{ then } [(A,a') \models \theta] = \mathbf{T}, \\ \mathbf{F} & \text{if there exists } a' \text{ s.t. } must^+(a,a') \text{ and } [(A,a') \models \theta] = \mathbf{F}, \\ \mathbf{F}_\exists & \text{if there exists } a' \text{ s.t. } must^-(a,a') \text{ and } [(A,a') \models \theta] \sqsupseteq \mathbf{F}_\exists, \\ \bot & \text{otherwise.} \end{cases}$$

The ordering relation ($\sqsupseteq$) used in the definition of $AX\theta$ refers to the information lattice. Note that the conditions for the **F** and the $\mathbf{F}_\exists$ conditions are not mutually exclusive. If both conditions hold, we take the value to be the stronger **F** value.

For clarity, we give the semantics for the existential operator EX explicitly (an equivalent definition follows from the semantics of AX and ¬):

$$[(A,a) \models EX\theta] = \begin{cases} \mathbf{F} & \text{if for all } a',\ \text{if } may(a,a') \text{ then } [(A,a') \models \theta] = \mathbf{F}, \\ \mathbf{T} & \text{if there exists } a' \text{ s.t. } must^+(a,a') \text{ and } [(A,a') \models \theta] = \mathbf{T}, \\ \mathbf{T}_\exists & \text{if there exists } a' \text{ s.t. } must^-(a,a') \text{ and } [(A,a') \models \theta] \sqsupseteq \mathbf{T}_\exists, \\ \bot & \text{otherwise.} \end{cases}$$

Thus, the semantics of the next-time operators follows both $must^-$ and $must^+$ transitions (that is, a' is such that $must^-(a,a')$ or $must^+(a,a')$, where the first is used for obtaining a $\mathbf{T}_\exists$ value and the second for obtaining a **T** value). To understand why $must^-$ transitions are suitable for falsification, let us explain the semantics for the EX modality. The other cases are similar (and are detailed in the proof of Theorem 2). Consider a concrete transition system $C = \langle AP, S_C, I_C, \longrightarrow_C, L_C\rangle$, and an abstraction for it $A = \langle AP, S_A, I_A, \xrightarrow{may}_A, \xrightarrow{must^+}_A, \xrightarrow{must^-}_A, L_A\rangle$. Let $\rho\colon S_C \to S_A$ be the witness function for the abstraction.

We argue that if $[(A,a) \models EXp] = \mathbf{T}_\exists$, then there is a concrete state c such that $\rho(c) = a$ and $c \models EXp$. By the semantics of the EX operator, $[(A,a) \models EXp] = \mathbf{T}_\exists$ implies that there is $a' \in S_A$ such that $must^-(a,a')$ and $L_A(a',p) \sqsupseteq \mathbf{T}_\exists$. Let $\hat{c}$ be a concrete state with $\rho(\hat{c}) = a'$ and $L_C(\hat{c},p) = \mathbf{T}$ (by the definition of abstraction, at least one such $\hat{c}$ exists). Since $must^-(a,a')$, then for every concrete state c' such that $\rho(c') = a'$ there is a concrete state c such that $\rho(c) = a$ and $c \longrightarrow_C c'$. In particular, there is a concrete state c such that $\rho(c) = a$ and $c \longrightarrow_C \hat{c}$. Thus, $c \models EXp$ and we are done.

Let a and a' be abstract states. The (reflexive) transitive closure of $must^-$, denoted $[must^-]^*$ is defined in the expected manner as follows: $[must^-]^*(a, a'')$ if either $a = a''$ or there is an abstract state a' such that $[must^-]^*(a, a')$ and $must^-(a', a'')$. We say that an abstract state a' is *onto reachable* from an abstract state a if for every concrete state c' that satisfies a', there is a concrete state c that satisfies a and c' is reachable from c. Dually, we can define the transitive closure of $must^+$ transitions, denoted $[must^+]^*$. Thus, $[must^+]^*(a, a'')$ if either $a = a''$ or there is an abstract state a' such that $must^+(a, a')$ and $[must^+]^*(a', a'')$. We say that an abstract state a' is *total reachable* from an abstract state a if for every concrete state c that satisfies a, there is a concrete state c' that satisfies a' and c' is reachable from c. The onto nature of $must^-$ transitions is retained by their transitive closure, and similarly for the total nature of $must^+$ transitions: $[must^-]^*(a, a')$ only if a' is onto reachable from a, and $[must^+]^*(a, a')$ only if a' is total reachable from a [Bal04].

Adding transitive closure (fixed-point) operators increases the expressive power of PML and, by the above discussion, retains the intuitive meaning of the values in $\mathcal{L}_6$. By extending PML by fixed-point operators, one gets the logic μ-calculus [Koz83], which subsumes the branching temporal logics CTL and CTL*. The 3-valued semantics of PML can be extended to the μ-calculus [BG04]. Note that in the special case of CTL and CTL* formulas, this amounts to letting path formulas range over may and $must^+$ paths [SG03]. The fact that the "onto" nature of $must^-$ transitions is retained under transition closure enables us to extend the soundness argument for the 6-valued semantics described above for a single EX or AX modality to nesting of such modalities and thus, to PML and the μ-calculus.

Let us explain the extension of the 6-valued semantics to μ-calculus in more detail. A formula ψ can be viewed as a function $\psi : S_A \to \mathcal{L}_6$. Each operator of PML can be viewed as a mapping from such a function (or two such functions, in case the operator is binary) to a new function. For example, the EX operator maps the function that corresponds to some formula θ to a function that corresponds to the formula $EX\theta$. As detailed in [BG04], every operator $f : (S_A \to \mathcal{L}_6) \to (S_A \to \mathcal{L}_6)$ has a least fixed-point, which is equal to the meet (with respect to the truth lattice) of its fixed points. Moreover, since PML operators are monotonic, the least fixed-point can be calculated by repeated iterations of f. Let $f_{\mathbf{F}}$ be the valuation in which all states have value **F**. Let $\psi_1^*(f_{\mathbf{F}})$ denote the successive application of ψ_1 from $f_{\mathbf{F}}$ until a fixed point is reached. To obtain a 6-valued semantics of μ-calculus, we extend the PML semantics above with the following definition:

$$[(A, a) \models \mu Z.\psi_1(Z)] = [(A, a) \models \psi_1^*(f_{\mathbf{F}})]$$

We now state that the intuitive meaning of the six values in $\mathcal{L}_6$ corresponds to the formal semantics we have described:

Theorem 2. *Let C be a concrete system and let A be its abstraction according to an abstraction function ρ. For every μ-calculus formula ψ and abstract state a of A, the following holds.*

– *If $[(A, a) \models \psi] = \mathbf{T}$, then for all concrete states c such that $\rho(c) = a$, we have that $c \models \psi$.*

– *If* $[(A, a) \models \psi] = \mathbf{F}$*, then for all concrete states* c *such that* $\rho(c) = a$*, we have that* $c \not\models \psi$.
– *If* $[(A, a) \models \psi] = \mathbf{T}_\exists$*, then there is a concrete state* c *such that* $\rho(c) = a$ *and* $c \models \psi$.
– *If* $[(A, a) \models \psi] = \mathbf{F}_\exists$*, then there is a concrete state* c *such that* $\rho(c) = a$ *and* $c \not\models \psi$.
– *If* $[(A, a) \models \psi] = \mathbf{M}$*, then there is a pair of concrete states* c *and* c' *such that* $\rho(c) = \rho(c') = a$*,* $c \models \psi$*, and* $c' \not\models \psi$.

Proof. The proof proceeds by induction on the nesting depth of fixed-point operators in ψ. The proofs of both the induction base and step proceed by induction on the structure of ψ.

We start with the induction base, that is, when ψ has no fixed-point operators. For the (internal) induction base, consider the case where $\psi = p$ for some $p \in AP$. Then, $[(A, a) \models p] = L_A(a, p)$, and the claim follows from the definition of $L_A(a, p)$.

Let $\psi = \neg\psi_1$. Then, the proof follows immediately from the semantics of the $\neg$ operation, and the inductive hypothesis. Let $\psi = \psi_1 \wedge \psi_2$. We have to show that the truth table for the $\wedge$ operator follows the intuitive meaning of the six values. We distinguish between the various possible values for $[(A, a) \models \psi]$.

– If $[(A, a) \models \psi] = \mathbf{F}$, then, according to the truth table of $\wedge$, one of $\{\psi_1, \psi_2\}$ evaluates to **F**. Without loss of generality, assume it is ψ_1. By the inductive hypothesis for ψ_1, for all concrete states c such that $\rho(c) = a$ we have that $c \not\models \psi_1$. Thus, for all such c, we have that $c \not\models \psi_1 \wedge \psi_2$, regardless of the value of ψ_2 on c.
– If $[(A, a) \models \psi] = \mathbf{F}_\exists$, then one of $\{\psi_1, \psi_2\}$ evaluates to $\mathbf{F}_\exists$ or **M**. Without loss of generality, assume it is ψ_1. By the inductive hypothesis for ψ_1, in either case ($\mathbf{F}_\exists$ or **M**) there exists a concrete state c such that $\rho(c) = a$ and $c \not\models \psi_1$. Thus, whatever the value of ψ_2 on c is, $c \not\models \psi_1 \wedge \psi_2$.
– If $[(A, a) \models \psi] = \mathbf{M}$, then by definition of $\wedge$, either $[(A, a) \models \psi_1] = \mathbf{M}$ and $[(A, a) \models \psi_2] = \mathbf{T}$, or $[(A, a) \models \psi_2] = \mathbf{M}$ and $[(A, a) \models \psi_1] = \mathbf{T}$. Assume, without loss of generality, that the first case holds. By the inductive hypothesis for ψ_1, there exist two concrete states, c' and c'', such that $\rho(c') = \rho(c'') = a$, $c' \models \psi_1$, and $c'' \not\models \psi_1$. From the latter, it follows that $c'' \not\models \psi_1 \wedge \psi_2$. By the inductive hypothesis for ψ_2, for all concrete states c such that $\rho(c) = a$, $c \models \psi_2$. In particular, $c' \models \psi_2$, and therefore, $c' \models \psi_1 \wedge \psi_2$.
– If $[(A, a) \models \psi] = \mathbf{T}_\exists$, then either $[(A, a) \models \psi_1] = \mathbf{T}$ and $[(A, a) \models \psi_2] = \mathbf{T}_\exists$, or $[(A, a) \models \psi_2] = \mathbf{T}$ and $[(A, a) \models \psi_1] = \mathbf{T}_\exists$. Assume, without loss of generality, that the first case holds. Since $[(A, a) \models \psi_2] = \mathbf{T}_\exists$, then, by the induction hypothesis, there exists a concrete state c such that $\rho(c) = a$ and $c \models \psi_2$. Also, since $[(A, a) \models \psi_1] = \mathbf{T}$, then, by the induction hypothesis, $c \models \psi_1$ as well. Therefore, $c \models \psi_1 \wedge \psi_2$.
– If $[(A, a) \models \psi] = \mathbf{T}$, then both $[(A, a) \models \psi_1] = \mathbf{T}$ and $[(A, a) \models \psi_2] = \mathbf{T}$, and the claim follows immediately from the induction hypothesis.

Let $\psi = AX\psi_1$. In the cases when $[(A, a) \models \psi] = \mathbf{T}$ and $[(A, a) \models \psi] = \mathbf{F}$, the arguments are similar to those of the three-valued semantics of the AX operator. If $[(A, a) \models \psi] = \mathbf{F}_\exists$, we argue that there is a concrete state c such that $\rho(c) = a$ and $c \models AX\theta$. By the definition of the semantics, there is an abstract state a' such that

$must^{-}(a, a')$, and $[(A, a') \models \theta] \sqsupseteq$ **F**$_{\exists}$, i.e., it can be any of the values **F**$_{\exists}$, **M** or **F**. By the inductive hypothesis for θ, in all of these three cases, there exists a concrete state c' such that $\rho(c') = a'$ and $c \not\models \theta$. Since $must^{-}(a, a')$, we know that there exists a concrete state c with $\rho(c) = a$ such that $c \rightarrow c'$. Thus, $c \not\models AX\theta$. Note that the AX case is reflected in the fact that the transitive closure of $must^{-}$ transitions preserve the "onto" nature of $must^{-}$ transitions.

We now move to the induction step. That is, assume that the claim holds for formulas of nesting depth l and prove it for a formula ψ of nesting depth $l + 1$. The cases where ψ is p, $\neg\psi_1$, $\psi_1 \wedge \psi_2$, or $AX\psi_1$ are as in the induction base. Let $\psi = \mu Z.\psi_1(Z)$. Again the **T** and **F** cases are as in the three-valued case. If $[(A, a) \models \psi] =$ **T**$_{\exists}$, we argue that there exists a concrete state c such that $\rho(c) = a$ and $c \models \psi$. According to the semantics for ψ, we have $[(A, a) \models \psi_1^*(f_{\mathbf{F}})] =$ **T**$_{\exists}$. Let k be such that $\psi^*(f_{\mathbf{F}}) = \psi_1^k(f_{\mathbf{F}})$, where $\psi_1^k(f_{\mathbf{F}})$ denotes the operator defined by k successive applications of the operator ψ_1, starting from $f_{\mathbf{F}}$. Thus, $[(A, a) \models \psi_1^k(f_{\mathbf{F}})] =$ **T**$_{\exists}$. The nesting depth of fixed-point operators in the formula $\psi_1^k(f_{\mathbf{F}})$ is l, so the induction hypothesis applies to it. Hence, there exists a concrete state c such that $\rho(c) = a$ and $c \models \psi_1^k(f_{\mathbf{F}})$. Thus, $c \models \psi$. The proof for the other truth values is similar.

We note that when $[(A, a) \models \psi] = \perp$, it is possible to use the same techniques as in the verification setting [SG03], with $must^{-}$ transitions replacing $must^{+}$ transitions in order to refine the TMTS.

3 Weak Reachability

When reasoning about paths in the abstract system, one can often manage with an even weaker type of reachability (than transitive closure over $must^{-}$ transitions): we say that an abstract state a' is *weakly reachable* from an abstract state a if there is a concrete state c' that satisfies a', there is a concrete state c that satisfies a, and c' is reachable from c. The combination of $must^{+}$ and $must^{-}$ transitions turn out to be especially powerful when reasoning about weak reachability.

If there are three abstract states a_1, a_2, and a_3 such that a_2 is onto reachable from a_1 and a_3 is total reachable from a_2, then a_3 is weakly reachable from a_1. Hence, weak reachability can be concluded from the existence of a sequence of $must^{-}$ transitions followed immediately by a sequence of $must^{+}$ transitions:

Theorem 3. [Bal04] *If* $[must^{-}]^*(a_1, a_2)$ *and* $[must^{+}]^*(a_2, a_3)$*, then* a_3 *is weakly reachable from* a_1.

In this section, we show how reasoning about weak reachability can be made tighter in the context of predicate abstraction. In Section 4, we show applications of weak reachability for falsification of LTL properties and for generating testing goals for a concrete system by reasoning about its abstraction.

Remark 1. The argument that a sequence of $must^{-}$ transitions followed by a sequence of $must^{+}$ transitions is a sufficient condition for weak reachability can be generalized in order to obtain a more precise 6-valued semantics for the μ-calculus (more precise in the sense that formulas may be evaluated to greater values in the information lattice). For this purpose, one can attribute the truth values to a formula in which EX and

AX modalities are nested by a mode flag from the set $\{-, +\}$. A $-$ flag indicates that only $must^-$ transitions have been taken in the evaluation of the formula, and a $+$ flag indicates that the sequence of $must^-$ transitions is followed by a sequence of $must^+$ transitions. As long as the truth value is attributed by $-$, evaluation of the EX and AX modalities can proceed along either $must^-$ transitions (leaving the flag unchanged) or $must^+$ transitions (updating the flag to $+$). Once the attribution is $+$, evaluation should proceed only along $must^+$ transitions. This guarantees that the path generated in the abstract system corresponds to a real path in the concrete system.

3.1 Weak Reachability in Predicate Abstraction

We now focus on the case where the concrete system is a program, and its abstraction is obtained by predicate abstraction. We then show that weak reachability can be made tighter by parameterizing the abstract transitions by predicates. The predicates used in these transitions may be (and usually are) different from the predicates used for predicate abstraction.

Consider a program P. Let X be the set of variables appearing in the program and variables that encode the program location, and let D be the domain of all variables (for technical simplicity, we assume that all variables are over the same domain). We model P by a concrete transition system in which each state is labeled by a valuation in D^X. Let $\Phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$ be a set of predicates (quantifier-free formulas of first-order logic) on X. For a set $a \subseteq \Phi$ and an assignment $c \in D^X$, we say that c *satisfies* a iff c satisfies all the predicates in a. The satisfaction relation induces a total and onto function $\rho : D^X \rightarrow 2^\Phi$, where $\rho(c) = a$ for the unique a for which c satisfies a. An abstraction of the program P that is based on Φ is a TMTS with state space 2^Φ, thus each state is associated (and is labeled by) the set of predicates that hold in it. For a detailed description of predicate abstraction see [GS97,BMMR01].

Note that all the transitions of the concrete system in which only the variables that encode the program location are changed (all transitions associated with statements that are not assignments, c.f., conditional branches, skip, etc.) are both $must^+$ and $must^-$ transitions, assuming that Φ includes all conditional expressions in the program. We call such transitions *silent* transitions. The identification of silent transitions makes our reasoning tighter: if $a \xrightarrow{silent}_A a'$ we can replace the transition from a to a' with transitions from a's predecessors to a'. The type of a new transition is the same as the type of the transitions leading to a. [9] Such elimination of silent transitions result in an abstract system in which each transition is associated with an assignment statement.

For simplicity of exposition, we first present a toy example. (We later present a more realistic example.) Consider the program P appearing in Figure 2.

When describing an abstract system, it is convenient to describe an abstract state in S_A as a pair of program location and a Boolean vector describing which of the program predicates in Φ hold. Let $\phi_1 = (x < 6)$ and $\phi_2 = (x > 7)$. The abstraction of P that corresponds to the two predicates is described in the left-hand side of Figure 3. In the right-hand side, we eliminate the silent transitions.

[9] A transition from a' may also be silent, in which case we continue until the chain of silent transitions either reaches an end state or reaches an assignment statement. If the chain reaches an end state, we can make a an end state.

```
L0   if x < 6 then
L1       x := x + 3;
L2       if x > 7 then
L3           x := x − 3;
L4   end
```

Fig. 2. The program P.

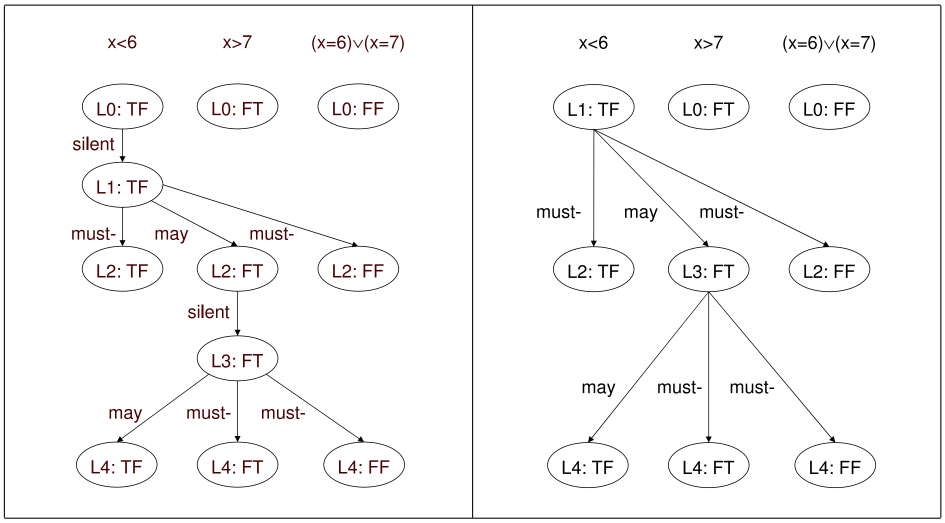


Fig. 3. The abstract transition system of the program P from Figure 2.

We now turn to study weak reachability in the abstract system. By Theorem 3, if $[must^{-}]^{*}(a_1, a_2)$ and $[must^{+}]^{*}(a_2, a_3)$, then a_3 is weakly reachable from a_1. While Theorem 3 is sound, it is not complete, in the sense that it is possible to have two abstract states a and a' such that a' is weakly reachable from a and still no sequence of transitions as specified in Theorem 3 exists in the abstract system. As an example, consider the abstract states $a = (\mathrm{L1} : \mathrm{TF})$ and $a' = (\mathrm{L4} : \mathrm{TF})$. While a' is weakly reachable from a; c.f., c' =(L4:$x = 5$) is reachable from c =(L0:$x = 5$), the only path from a to a' in the abstraction contains two may transitions, so Theorem 3 cannot be applied. In fact, the status of the abstract states (L4:FT) and (L4:FF) also is not clear, as the paths from a to these states do not follow the sequence specified in Theorem 3. Accordingly, Theorem 3 does not help us determining whether there is an input $x < 6$ to P such that the execution of P on x would reach location L4 with x that is strictly bigger than 7 or with x that is equal to 6 or 7. Our goal is to tighten Theorem 3, so that we end up with fewer such undetermined cases.

3.2 Parameterized Must Transitions

Recall that each abstract state is associated with a location of the program, and thus it is also associated with a statement. For a statement s and a predicate e over X, the *weakest precondition* $\mathrm{WP}(s, e)$ and the *strongest postcondition* $\mathrm{SP}(s, e)$ are defined as follows [Dij76]:

- The execution of s from every state that satisfies $\mathrm{WP}(s,e)$ results in a state that satisfies e, and $\mathrm{WP}(s,e)$ is the weakest predicate for which the above holds. For an assignment statement $x := v$, we have that $\mathrm{WP}(x := v, e) = e[x/v]$ (that is, e with all occurrences of x replaced by v).
- The execution of s from a state that satisfies e results in a state that satisfies $\mathrm{SP}(s,e)$, and $\mathrm{SP}(s,e)$ is the strongest predicate for which the above holds. For an assignment statement $x := v$, we have that $\mathrm{SP}(x := v, e) = \exists x'.(e[x/x'] \wedge x = v)$.

For example, in the program P, we have $\mathrm{WP}(x := x+3, x > 7) = x > 4$, $\mathrm{SP}(x := x+3, x < 6) = x < 9$, $\mathrm{WP}(x := x-3, x < 6) = x < 9$, and $\mathrm{SP}(x := x-3, x > 7) = x > 4$.

Let θ be a predicate over X. We parameterize $must^+$ and $must^-$ transitions by θ as follows:

- $must^+(\theta)(a, a')$ only if for every concrete state c that satisfies $a \wedge \theta$, there is a concrete state c' that satisfies a' and $c \longrightarrow_C c'$.
- $must^-(\theta)(a, a')$ only if for every concrete state c' that satisfies $a' \wedge \theta$, there is a concrete state c that satisfies a and $c \longrightarrow_C c'$.

Thus, a $must^+(\theta)$ transition is total from all states that satisfy θ, and a $must^-(\theta)$ transition is onto all states that satisfy θ. Note that when $\theta = \mathbf{T}$, we get usual $must^+$ and $must^-$ transitions. Parameterized transitions can be generated automatically (using WP and SP) while building the TMTS without changing the complexity of the abstraction algorithm.

Theorem 4. *Let a and a' be two abstract states, and s the statement executed in a. Then, $must^+(\mathrm{WP}(s,a'))(a,a')$ and $must^-(\mathrm{SP}(s,a))(a,a')$.*

Proof. *Consider a concrete state c that satisfies $a \wedge \mathrm{WP}(s,a')$. Since c satisfies $\mathrm{WP}(s,a')$, the execution of s from c results in a state that satisfies a'. Thus, every such c has a concrete successor state c' that satisfies a'; thus $must^+(\mathrm{WP}(s,a'))(a,a')$.*

Consider a concrete state c' that satisfies $a' \wedge \mathrm{SP}(s,a)$. Since c' satisfies $\mathrm{SP}(s,a)$, it is obtained by executing s in a state that satisfies a. Thus, every such c' has a concrete predecessor state c that satisfies a; thus $must^-(\mathrm{SP}(s,a))(a,a')$.

The good news about Theorem 4 is that it is complete in the sense that for all predicates θ, if there is a $must^+(\theta)$ transition from a to a', then $a \Rightarrow (\theta \Rightarrow \mathrm{WP}(s,a'))$, and similarly for $must^-$ transitions, as formalized below.

Lemma 1. *Let a and a' be two abstract states, and s the statement executed in a.*

- *If there is a $must^+(\theta)$ transition from a to a', then $a \Rightarrow (\theta \Rightarrow \mathrm{WP}(s,a'))$.*
- *If there is a $must^-(\theta)$ transition from a to a', then $a' \Rightarrow (\theta \Rightarrow \mathrm{SP}(s,a))$.*

Proof. *Let θ be such that there is a $must^+(\theta)$ transition from a to a'. Then, for every c that satisfies $\theta \wedge a$, the execution of s from c results in c' that satisfies a'. By definition, $\mathrm{WP}(s,a')$ contains exactly all states from which the execution of s results in a state that satisfies a'. Hence, $(\theta \wedge a) \Rightarrow \mathrm{WP}(s,a')$, or equivalently, $a \Rightarrow (\theta \Rightarrow \mathrm{WP}(s,a'))$.*

Let θ be such that there is a $must^-(\theta)$ transition from a to a'. Then, every c' that satisfies $\theta \wedge a'$ is obtained by executing s in a state that satisfies a. By definition $\mathrm{SP}(s,a)$ contains exactly all states obtained by executing s is a state that satisfies a. Hence, $(\theta \wedge a) \Rightarrow \mathrm{SP}(s,a)$, or equivalently, $a \Rightarrow (\theta \Rightarrow \mathrm{SP}(s,a))$.

Thus, the pre and post conditions, which can be generated automatically, are the strongest predicates that can be used. Note that using Theorem 4, it is possible to replace all may transitions by parameterized $must^-$ and $must^+$ transitions.

It is easy to see how parameterized transitions can help when we consider weak reachability. Indeed, if $must^-(\theta_1)(a, a')$, $must^+(\theta_2)(a', a'')$, and $\theta_1 \wedge \theta_2 \wedge a'$ is satisfiable, then a'' is weakly reachable from a, as formalized by the following lemma.

Lemma 2. *If $must^-(\theta_1)(a, a')$, $must^+(\theta_2)(a', a'')$, and $\theta_1 \wedge \theta_2 \wedge a'$ is safistiable, then there are concrete states c and c'' such that $a(c)$, $a''(c'')$, and c'' is reachable from c.*

Proof. We show that there are concrete states c and c'' for which there is a concrete state c' such that $c \longrightarrow_C c'$ and $c' \longrightarrow_C c''$. We choose c' as a satisfying assignment for $\theta_1 \wedge \theta_2 \wedge a'$. Since $must^-(\theta_1)(a, a')$ and c' satisfies $\theta_1 \wedge a'$, there is a concrete state c such that $c \longrightarrow_C c'$. Simlarly, since $must^+(\theta_2)(a', a'')$ and c' satisfies $\theta_2 \wedge a'$, there is a concrete state c'' such that $c' \longrightarrow_C c''$.

The completeness of Theorem 4 implies that when a' is weakly reachable from a via two transitions, this always can be detected by taking $\theta_1 = \mathrm{SP}(s, a)$ and $\theta_2 = \mathrm{WP}(s', a')$, where s and s' are the statements executed in the two transitions.

In our example, we have seen that the transitions from (L1:TF) to (L3:FT) and from (L3:FT) to (L4:TF) are both may transitions, and thus Theorem 3 cannot be applied. However, the fact that the first transition also is a $must^-(x < 9)$ transition and the second also is a $must^+(x < 9)$, together with the fact that $x > 7 \wedge x < 9$ is satisfiable, guarantee that there is a concrete state that corresponds to (L1:TF) and from which a concrete state that corresponds to (L4:TF) is reachable. Indeed, as we noted earlier, (L4:$x = 5$) is reachable from (L0:$x = 5$).

When a and a' are of distance greater than two transitions, parameterization is useful for composing the sequence of $must^-$ transitions with the sequence of $must^+$ transitions:

Theorem 5. *If $[must^-]^*(a_1, a_2)$, $must^-(\theta_1)(a_2, a_3)$, $must^+(\theta_2)(a_3, a_4)$, $[must^+]^*(a_4, a_5)$, and $a_3 \wedge \theta_1 \wedge \theta_2$ is satisfiable, then a_5 is weakly reachable from a_1.*

Proof. Immediate from Theorem 3 and Lemma 2.

Again, the predicates θ_1 and θ_2 are induced by the pre and postconditions of the statement leading to the abstract state in which the two sequences are composed.

The transitive closure of the parameterized $must$ transitions does not retain the reachability properties of a single transition and requires reasoning in an assume-guarantee fashion, where two predicates are associated with each transition.

3.3 Assume-guarantee Must Transitions

Let a and a' be abstract states, and let θ and θ' be predicates over X.

- $\langle\theta\rangle must^+\langle\theta'\rangle(a, a')$ only if for every concrete state c that satisfies $a \wedge \theta$, there is a concrete state that satisfies $a' \wedge \theta'$ and $c \longrightarrow_C c'$.
- $\langle\theta\rangle must^-\langle\theta'\rangle(a, a')$ only if for every concrete state c' that satisfies $a' \wedge \theta'$, there is a concrete state that satisfies $a \wedge \theta'$ and $c \longrightarrow_C c'$.

So, a $\langle\theta\rangle must^+\langle\theta'\rangle$ transition is total from all states that satisfy θ, and it is guaranteed that the transition from such states result in states that satisfy θ'. Similarly, a $\langle\theta\rangle must^-\langle\theta'\rangle$ transition is onto all states that satisfy θ', and is from states that satisfy θ. We can now define the transitive closure of the relations. Let a and a'' be abstract states, and let θ_1 and θ_3 be predicates.

- $[\langle\theta_1\rangle must^+\langle\theta_3\rangle]^*(a,a'')$ if either $a = a''$ or there is an abstract state a' and a predicate θ_2 such that $[\langle\theta_1\rangle must^+\langle\theta_2\rangle]^*(a,a')$ and $\langle\theta_2\rangle must^+\langle\theta_3\rangle(a',a'')$.
- $[\langle\theta_1\rangle must^-\langle\theta_3\rangle]^*(a,a'')$ if either $a = a''$ or there is an abstract state a' and a predicate θ_2 such that $\langle\theta_1\rangle must^-\langle\theta_2\rangle(a,a')$ and $[\langle\theta_2\rangle must^-\langle\theta_3\rangle]^*(a',a'')$.

The transitive closure retains the reachability properties of a single transition:

Theorem 6.

- $[\langle\theta\rangle must^+\langle\theta'\rangle]^*(a,a')$ *only if for every concrete state* c *that satisfies* $a \wedge \theta$*, there is a concrete state* c' *that satisfies* $a' \wedge \theta'$ *and* c' *is reachable from* c*.*
- $[\langle\theta\rangle must^-\langle\theta'\rangle]^*(a,a')$ *only if for every concrete state* c' *that satisfies* $a' \wedge \theta'$*, there is a concrete state* c *that satisfies* $a \wedge \theta$ *and* c' *is reachable from* c*.*

We can now use Theorem 6 for automatic reasoning about weak reachability:

Theorem 7. *Consider a path* $\pi = a_1, a_2, \ldots, a_n$ *in the TMTS. Let* s_i *be the statement executed along the transition from* a_i *to* a_{i+1}*. Let* $\theta_1 = a_1$*,* $\xi_n = a_n$*, and for all* $1 \le i \le n-1$*, let* $\theta_{i+1} = a_{i+1} \wedge \mathrm{SP}(s_i, \theta_i)$*, and* $\xi_i = a_i \wedge \mathrm{WP}(s_i, \xi_{i+1})$*. Then, for all* $1 \le i \le n-1$*, we have* $\langle\theta_i\rangle must^-\langle\theta_{i+1}\rangle(a_i, a_{i+1})$ *and* $\langle\xi_i\rangle must^+\langle\xi_{i+1}\rangle(a_i, a_{i+1})$*. Also, the following are equivalent:*

1. a_n *is weakly reachable from* a_1 *via* π*.*
2. *For all* $1 \le i \le n$*, we have that* $\theta_i \wedge \xi_i$ *is satisfiable.*
3. *There is* $1 \le i \le n$ *for which* $\theta_i \wedge \xi_i$ *is satisfiable.*

Proof. First, we show that (1) implies (3). Weak reachability via π *means that there exist concrete states* $c_1, \ldots, c_n$ *that satisfy* $a_1, \ldots, a_n$*, respectively, such that* c_{i+1} *is obtained by executing the statement* s_i *on* c_i*, for* $1 \le i \le n-1$*. Since* c_1 *satisfies* $a_1 = \theta_1$*,* c_i *satisfies* θ_i *for* $1 \le i \le n-1$*, in particular* c_n *satisfies* θ_n*. From the fact that* c_n *satisfies* $a_n = \xi_n$*, we conclude that* c_n *satisfies* $\theta_n \wedge \xi_n$*.*

Second, we show that (3) implies (2). For the sake of contradiction, assume that there exist i *such that* $\theta_i \wedge \xi_i$ *is satisfiable, and (without loss of generality) assume that* θ_{i+1} *is unsatisfiable. Let* c *be a concrete state that satisfies* $\theta_i \wedge \xi_i$*. By definition of* ξ_i*,* c *satisfies* $WP(s_i, \xi_{i+1})$*, that is, there exists a concrete store* c' *that satisfies* a_{i+1} *(because* ξ_{i+1} *is implied by* a_{i+1}*) and that is reachable from* c *using* s_i*. Since* c *satisfies* θ_i*,* c' *satisfies* $SP(s_i, \theta_i)$*, thus* c' *satisfies* $\theta_{i+1} = a_{i+1} \wedge SP(s_i, \theta_i)$*, and a contradiction is obtained.*

Finally, we can show that (2) implies (1) using Theorem 6 and the fact that there is a $[\langle\theta_1 \wedge \xi_1\rangle must^+\langle\theta_n \wedge \xi_n\rangle]^*$ *transition from* a_1 *to* a_n*.*

The reasoning in Theorem 7 is similar to known methods where the iterative application of pre- and post-conditions are applied in order to test reachability [HJMM04]. The new feature of our approach is the fact we allow reasoning in both forward and backwards directions, and we intersect the intermediate predicates with the current abstract state of π. That way, we can reason not only about a_n being weakly reachable from a_1, but about a_n being weakly reachable from a_1 along π. As we discuss in Section 4, this is useful when studying path coverage in the abstract system or in counterexample-guided abstraction refinement, when the feasibility of a path needs to be checked. Note that, being intersected with the intermediate abstract states, the predicates we get are tighter than these obtained by only iterating pre and post conditions. Thus, they can also be used for a finer partition of states in case one wishes to eliminate infeasible paths from the abstraction [CGJ$^+$03,HJMM04].

Remark 2. Several refinement techniques are based on partitioning an abstract state a to two abstract states $a \wedge \theta$ and $a \wedge \neg\theta$ [CGJ$^+$03]. Our parameterization method, on the other hand, aims at answering weak reachability queries, and parameterization of a $must^+$ transition from a (or a $must^-$ transition to a) with θ essentially replaces a by $a \wedge \theta$ and ignores $a \wedge \neg\theta$. Also, θ is local, depends on the transition to/from a and is induced by the pre- and post-condition of the statement executed along this transition. Reaching a with several different transitions leads to different parameters, which are independent of each other. Most importantly, our parameterization method provides a way to refine the abstraction without increasing the number of state, thus preventing state-space explosion as a result of refinement.

4 Applications

This section describes application of weak reachability for linear-time falsification and for abstraction-guided test generation.

4.1 Linear-time Falsification

In *linear-time model checking*, we check whether all the computations of a given program P satisfy a specification ψ, say an LTL formula. In the automata-theoretic approach to model checking [Kur94,VW94], one constructs an automaton $\mathcal{A}_{\neg\psi}$ for the negation of ψ. The automaton $\mathcal{A}_{\neg\psi}$ is usually a nondeterministic Büchi automaton, where a run is accepting iff it visits a set of designated states infinitely often. The program P is faulty with respect to ψ if the product of $\mathcal{A}_{\neg\psi}$ with the program contains a fair path – one that visits the set of designated states infinitely often. The product of $\mathcal{A}_{\neg\psi}$ with an abstraction of P may contain fair paths that do not correspond to computations of P, thus again there is a need to check for weak reachability.

Let Φ_{spec} be the set of predicates induced by the LTL formula. For example, if $\psi = G((x \geq 4) \Rightarrow F(x = 0))$, then $\Phi_{spec} = \{x \geq 4, x = 0\}$, and let $\mathcal{A}_{\neg\psi} = \langle 2^{\Phi_{spec}}, Q, Q_{in}, \delta, \alpha \rangle$. Also, let $P_A = \langle \Phi_{prog} \cup \Phi_{spec}, S_A, I_A, \xrightarrow{may}_A, \xrightarrow{must^+}_A, \xrightarrow{must^-}_A, L_A \rangle$, be the abstraction of P according to the union of Φ_{prog} with Φ_{spec}. We define the product of P_A with $\mathcal{A}_{\neg\psi}$ as a TMTS $\mathcal{P} = \langle \emptyset, S_A \times Q, I_A \times Q_{in}, \xrightarrow{may}, \xrightarrow{must^+}, \xrightarrow{must^-}, L \rangle$, where for all $\gamma \in \{may, must^+, must^-\}$, we have that $\langle a, q \rangle \xrightarrow{\gamma} \langle a', q' \rangle$ iff $a \xrightarrow{\gamma}_A a'$ and $q' \in \delta(q, L_A(a) \cap \Phi_{spec})$. Thus, the product TMTS contains behaviors that are joined to P_A and $\mathcal{A}_{\neg\psi}$. When reasoning about concrete systems, emptiness of the product automaton can be reduced to a search for an accepting state that is reachable from both an initial state and itself. In the context of abstraction, we should make sure that the path from the accepting state to itself can be repeated, thus weak reachability is too weak here[10], and instead we need the following.

Theorem 8. *If there are abstract states $a_{init} \in I_A \times Q_{in}$ and $a_{acc} \in S_A \times \alpha$ such that a_{acc} is onto reachable from a_{init} and from itself, or a_{acc} is weakly reachable from a_{init} and total reachable from itself, then P violates ψ.*

4.2 Testing

Falsification methods are related to *testing*, where the system is actually executed. The infeasible task of executing the system with respect to all inputs is replaced by checking a test suite consisting of a finite subset of inputs. It is very important to measure the exhaustiveness of the test suite, and indeed, there has been an extensive research in the testing community on *coverage metrics*, which provide such a measure [CKKV01].

Some coverage metrics are defined with respect to an abstraction of the system. For example, in *predicate-complete testing* [Bal04], the goal is to cover all the reachable observable states (evaluation of the system's predicates under all reachable states), and reachability is studied in an abstract system whose state space consists of an overapproximation of the reachable observable states. The observable states we want our test suite to cover are abstract states that are weakly reachable.

The fundamental question in this setting is how to determine which abstract states are weakly reachable. As we have seen, TMTS provide a sufficient condition for determining weak reachability (via a sequence of $must^-$ transitions followed by a sequence of $must^+$ transitions). The parameterization method makes this condition tighter, and its combination with an assume-guarantee reasoning makes it complete. In Section 4.3 we demonstrate the usefulness of our approach: we manage to identify all the weakly reachable states in a non-trivial example.

4.3 Example

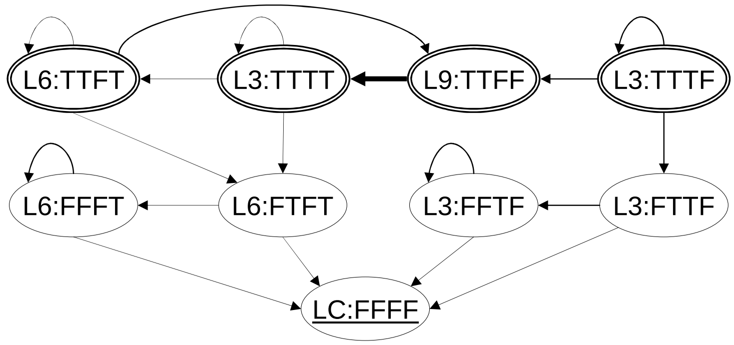
In this section we demonstrate the usefulness of our parameterization method with respect to a more interesting function. The function we consider is from [Bal04], where it is used to demonstrate how $must^-$ transitions improve reasoning about weak reachability. Still, some of the weakly reachable abstract states in the example cannot be detected using the techniques in [Bal04], where the problem of making the technique tighter is left open. We show that our parameterization method can identify all the weakly reachable states in this example. Also, it can show that certain paths are infeasible in the concrete system, thus one need not worry about the fact they are not covered by the test suite.

Figure 4(a) presents a (buggy) example of the QuickSort's `partition` function, a classic example that has been used to study test generation. The goal of the function is to permute the elements of the input array so that the resulting array has two parts: the values in the first part are less than or equal to the chosen pivot value `a[0]`; the values in the second part are greater than the pivot value.

[10] When ψ is a safety property, $\mathcal{A}_{\neg\psi}$ is an automaton accepting finite bad prefixes [KV01], and weak reachability is sufficient.

```
void partition(int a[], int n)  {
     assume(n>2);
     int p := a[0];
     int lo := 1;
     int hi := n-1;
L0:  while (lo <= hi) {
L2:    while (a[lo] <= p) {
L3:      lo:=lo+1;
       }
L5:    while (a[hi] > p) {
L6:      hi:=hi-1;
       }
       if (lo < hi) {
L9:      swap(a,lo,hi);
       }
     }
LC: ;
}
```

(a)



(b)

Fig. 4. (a) The `partition` function and (b) its TMTS after removing silent transitions.

There is an array bound check missing in the code that can lead to an array bounds error: the check at the **while** loop at label `L2` should be `(lo<=hi && a[lo]<=p)`.[11] This error only can be uncovered by executing the statement "`lo:=lo+1;`" at label `L3` at least twice.

We consider the TMTS generated by predicate abstraction with respect to the four predicates that appear in the conditional guards of the function: `(lo<hi)`, `(lo<=hi)`, `(a[lo]<=p)`, and `(a[hi]>p)`. An observed state thus is a bit vector of length four $(\mathbf{lt}, \mathbf{le}, \mathbf{al}, \mathbf{ah})$, where **lt** corresponds to `(lo<hi)`, **le** corresponds to `(lo<=hi)`, **al** corresponds to `(a[lo]<=p)`, and **ah** corresponds to `(a[hi]>p)`. There only are ten

[11] The loop at `L5` cannot decrement `hi` to take a value less than zero because the value of variable `p` is fixed to be the value of `a[0]`. One could argue that one would want to put a bounds check in anyway.

feasible valuations for this vector, as six are infeasible because of correlations between the predicates.

Figure 4(b) shows the TMTS of the `partition` function with respect to the four observed predicates. We have removed silent transitions, thus only states associated with locations in assignment statements appear in the TMTS. The initial states are denoted by double-lined ovals. The sole final state is denoted with underlined text. All the transitions in the TMTS are may transitions (solid lines), except for the one from (L9:TTFF) to (L3:TTTT), which is both $must^-$ and $must^+$ (as such, we could have regarded it as a silent transition and removed it as well; we kept it as L9 is associated with an assignment). This transition is denoted by a bold line in the figure.

Due to the preponderance of may transitions in the TMTS, no weak reachability information follows from Theorem 3, except for the initial states. However, parameterized transitions can be used in order to conclude that all the states are weakly reachable. Let us consider the may-transition (L3:TTTF) $\xrightarrow{may}_A$ (L3:FTTF). By Theorem 4, this transition can also be viewed as a parameterized $must^-(\theta_1)$ transition with the predicate θ_1 defined by $\mathrm{SP}(\text{lo:=lo+1}, \mathrm{TTTF})$, where TTTF denotes the formula $(lo < hi) \wedge (lo \leq hi) \wedge (a[lo] \leq p) \wedge \neg(a[hi] > P)$. Thus, θ_1 is $(lo - 1 < hi) \wedge (a[lo-1] < p) \wedge \neg(a[hi] > p)$.

Consider now the may transition (L3:FTTF) $\xrightarrow{may}_A$ (L3:FFTF). By Theorem 4, this transition can also be viewed as a parameterized $must^+(\theta_2)$ transition with the predicate θ_2 defined by $\mathrm{WP}(\text{lo:=lo+1}, \mathrm{FFTF})$. That is, θ_2 is $(lo + 1 > hi) \wedge (a[lo+1] < p) \wedge (a[hi] \leq p)$.

By Theorem 5, the state (L3: FFTF) is weakly reachable from (L3:TTTF) through the state (L3:FTTF) if $\theta_1 \wedge \theta_2 \wedge$ FTTF is satisfiable. This condition can be simplified into the equivalent formula $(lo = hi) \wedge (a[lo] \leq p) \wedge (a[lo-1] < p) \wedge (a[lo+1] < p)$ which is indeed satisfiable. A test that observes this state will cause an array bounds violation (by incrementing the variable lo until $lo > hi$ without decrementing hi and then accessing $a[lo]$).

Similarly, it can be shown that the state (LC:FFFF) is weakly reachable from (L3:TTTF) through (L3:FTTF). First, there is a parameterized $must^+(\theta_3)$ transition from (L3:FTTF) to (LC:FFFF), where θ_3 is $\mathrm{WP}(\text{lo:=lo+1}, \mathrm{FFFF})$, that is $(lo + 1 > hi) \wedge (a[lo+1] > p) \wedge (a[hi] \leq p)$. Then, it can be shown that $\theta_1 \wedge \theta_3 \wedge$ FTTF is equivalent to $(lo = hi) \wedge (a[lo] \leq p) \wedge (a[lo+1] > p) \wedge (a[lo-1] < p)$, which is satisfiable.

It remains to be shown that the state (L6:FFFT) is weakly reachable from one of the initial states. Consider the path from (L3:TTTT) to (L6:FFFT) through (L6:FTFT). By Theorem 4, there is a $must^-(\mathrm{SP}(lo := lo + 1, \mathrm{TTTT}))$ transition from (L3:TTTT) to (L6:FTFT) and a $must^+(\mathrm{WP}(hi := hi - 1, \mathrm{FFFT}))$ transition from (L6:FTFT) to (L6:FFFT). However, we cannot apply Theorem 5, as $\mathrm{SP}(lo := lo + 1, \mathrm{TTTT}) \wedge \mathrm{WP}(hi := hi - 1, \mathrm{FFFT}) \wedge$ FTFT implies that $a[lo-1] > p \wedge a[lo-1] \leq p$, which is unsatisfiable. In fact, since every $must^-(\theta)(a, a')$ transition is also a $\langle a \rangle must^- \langle \theta \wedge a' \rangle$ transition, and every $must^+(\theta)(a', a'')$ transition is also a $\langle \theta \wedge a' \rangle must^+ \langle a'' \rangle$ transition, Theorem 7 it can be used to show that this path is infeasible. Weak reachability of the state (L6:FFFT) can then be proved via another path, and indeed it can be shown using Theorem 5 that (L6:FFFT) is weakly reachable from another initial state (L6:TTFT) through the state (L6:FTFT).

5 Conclusion

The notions of $must^-$ transitions and weak reachability, which were introduced in [Bal04], where elaborated and extended in the present paper. We described an abstraction framework that contains $must^-$ transitions, the backwards version of $must$ transitions, and showed how $must^-$ transitions enable reasoning about past-time modalities as well as future-time modalities in a six-valued semantics, which is suitable for both verification and falsification. We showed that the falsification setting allows for a stronger type of abstraction and described applications in falsification of temporal properties and testing.

A general idea in our work is that by replacing $must^+$ by $must^-$ transitions, abstraction frameworks that are sound for verification become abstraction frameworks that are sound (and more precise) for falsification. We demonstrated it with model checking and refinement, and we believe that several other ideas in verification can be lifted to falsification in the same way. This includes generalized model checking [GJ02], making the framework complete [DN05], and its augmentation with hyper-transitions [LX90,SG04]. Another interesting direction is to use $must^-$ transitions in order to strengthen abstractions in the verification setting: the ability to move both forward and backwards across the transition relation has proven helpful in the concrete setting (c.f. , [INH96,BGS00]). Using $must^-$ transitions, this also can be done in the abstraction setting. Finally, the sequential TMTS model can be extended to a concurrent one. Beyond the usual compositionality questions in the concurrent setting, there is a need to define and reason about the compositions of models that have different types of under-approximating transitions.

References

[Bal04] T. Ball. A theory of predicate-complete test coverage and generation. In *3rd International Symposium on Formal Methods for Components and Objects*, 2004.

[Ben91] J. Benthem. Languages in actions: categories, lambdas and dynamic logic. *Studies in Logic*, 130, 1991.

[BG99] G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *Computer Aided Verification*, pages 274–287, 1999.

[BG04] G. Bruns and P. Godefroid. Model checking with 3-valued temporal logics. In *31st International Colloquium on Automata, Languages and Programming*, volume 3142 of *Lecture Notes in Computer Science*, pages 281–293, 2004.

[BGS00] R. Bloem, H.N. Gabow, and F. Somenzi. An algorithm for strongly connected component analysis in $n \log n$ symbolic steps. In *Formal Methods in Computer Aided Design*, volume 1954 of *Lecture Notes in Computer Science*, pages 37–54. Springer-Verlag, 2000.

[BMMR01] T. Ball, R. Majumdar, T. D. Millstein, and S. K. Rajamani. Automatic predicate abstraction of c programs. In *PLDI*, pages 203–213, 2001.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for the static analysis of programs by construction or approximation of fixpoints. In *POPL 77: Principles of Programming Languages*, pages 238–252. ACM, 1977.

[CE81] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.

[CGJ+03] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, 50(5):752–794, 2003.

[CK02] H. Chockler and O. Kupferman. ω-regular languages are testable with a constant number of queries. In J.P. Rolim and S. Vadhan, editors, *Proceedings of 6th International Workshop on Randomization and Approximation Techniques*, volume 2483 of *Lecture Notes in Computer Science*, pages 26–38, Cambridge, MA, USA, 2002. Springer-Verlag.

[CKKV01] H. Chockler, O. Kupferman, R.P. Kurshan, and M.Y. Vardi. A practical approach to coverage in model checking. In *Proc. 13th International Conference on Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, pages 66–78. Springer-Verlag, 2001.

[DGG97] D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Trans. Program. Lang. Syst.*, 19(2):253–291, 1997.

[Dij76] E.W. Dijksta. *A Discipline of Programming*. Prentice-Hall, 1976.

[DN05] D. Dams and K. S. Namjoshi. Automata as abstractions. In *VMCAI 2005*, Paris, 2005. to appear, LNCS, Springer-Verlag.

[FKZ+00] R. Fraer, G. Kamhi, B. Ziv, M. Vardi, and L. Fix. Prioritized traversal: efficient reachability analysis for verication and falsification. In *Proc. 12th Conference on Computer Aided Verication*, volume 1855 of *Lecture Notes in Computer Science*, pages 389–402, Chicago, IL, USA, July 2000. 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. Springer-Verlag, 2001.

[GJ02] P. Godefroid and R. Jagadeesan. Automatic abstraction using generalized model checking. In *Computer Aided Verification*, pages 137–150, 2002.

[GLST05] O. Grumberg, F. Lerda, O. Strichman, and M. Theobald. Proof-guided underapproximation-widening for multi-process systems. In *POPL*, pages 122–131, 2005.

[GS97] S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In *CAV 97: Computer-aided Verification*, LNCS 1254, pages 72–83. Springer-Verlag, 1997.

[HJMM04] T. A. Henzinger, R. Jhala, R. Majumdar, and K. L. McMillan. Abstractions from proofs. In *POPL*, pages 232–244, 2004.

[HJS01] M. Huth, R. Jagadeesan, and D. Schmidt. Model checking partial state spaces with 3-valued temporal logics. In *ESOP*, pages 155–169, 2001.

[INH96] H. Iwashita, T. Nakata, and F. Hirose. CTL model checking based on forward state traversal. In *Proc. IEEE/ACM International Conference on Computer Aided Design*, pages 82–87, 1996.

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

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[Kur94] R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.

[KV01] O. Kupferman and M.Y. Vardi. Model checking of safety properties. *Formal methods in System Design*, 19(3):291–314, November 2001.

[LT88] K.G. Larsen and G.B. Thomsen. A modal process logic. In *Proc. 3th Symp. on Logic in Computer Science*, Edinburgh, 1988.

[LX90] K. G. Larsen and L. Xinxin. Equation solving using modal transition systems. In *LICS*, pages 108–117, 1990.

[PDV01] C. S. Pasareanu, M. B. Dwyer, and W. Visser. Finding feasible counter-examples when model checking abstracted java programs. In *TACAS*, pages 284–298, 2001.

[QS81] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Proc. 5th International Symp. on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, 1981.

[SG03] S. Shoham and O. Grumberg. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. In *Computer Aided Verification*, pages 275–287, 2003.

[SG04] S. Shoham and O. Grumberg. Monotonic abstraction-refinement for CTL. In *Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 2988 of *Lecture Notes in Computer Science*, pages 546–560. Springer-Verlag, 2004.

[Sip99] H.B. Sipma. *Diagram-based Verification of Discrete, Real-time and Hybrid Systems*. PhD thesis, Stanford University, Stanford, California, 1999.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.