3-Valued Abstraction: More Precision at Less Cost

Sharon Shoham   Orna Grumberg

Computer Science Department, Technion, Haifa, Israel

E-mail: {sharonsh,orna}@cs.technion.ac.il

Abstract

This paper investigates both the precision *and the model checking* efficiency *of abstract models designed to preserve branching time logics w.r.t. a 3-valued semantics. Current abstract models use ordinary transitions to over approximate the concrete transitions, while they use* hyper transitions *to under approximate the concrete transitions. In this work we refer to precision measured w.r.t. the choice of abstract states, independently of the formalism used to describe abstract models. We show that current abstract models do not allow maximal precision. We suggest a new class of models and a construction of an abstract model which is* most precise *w.r.t. any* choice of abstract states. As before, the construction of such models might involve an exponential blowup, which is inherent by the use of hyper transitions. We therefore suggest an efficient algorithm in which the abstract model is constructed* during *model checking, by need. Our algorithm achieves maximal precision w.r.t. the given property while remaining quadratic in the number of abstract states. To complete the picture, we incorporate it into an abstraction-refinement framework.*

1. Introduction

Abstraction is one of the most successful techniques for fighting the state explosion problem in model checking [3]. Abstractions hide some of the details of the verified system, thus result in smaller models. Most commonly used are state abstractions that collapse (possibly non disjoint) sets of concrete states into abstract states. As such, an abstraction consists of a set of abstract states S_A and a mapping (or concretization function) γ that defines the relation between abstract states and the concrete states that they represent. The rest of the components of the concrete model then also need to be lifted into the abstract world, in order to result in an abstract model. This can be done in various ways.

When using a 2-valued semantics, abstract models are usually designed to be *conservative* for *true*, meaning that truth of a formula is preserved from the abstract model to the concrete model. A greater advantage is obtained if the formula is interpreted w.r.t. a 3-valued semantics [1]. This semantics evaluates a formula to either *true*, *false* or *indefinite*. Abstract models can then be designed to be conservative for both *true* and *false*. Only if the value of a formula in the abstract model is indefinite, its value in the concrete model is unknown. We follow this approach.

The logic specifications we consider in this paper are formulas of the modal μ-*calculus* [15]. The modal μ-calculus is a powerful formalism for expressing properties of transition systems using fixpoint operators. In particular, it combines both existential and universal properties. As such, two transition relations are needed in an abstract model for it to be conservative w.r.t. the full μ-calculus (be it over a 2-valued or a 3-valued semantics). Examples of such abstract models are *modal transition systems* [18, 16] or *mixed transition systems* [7] that contain *may* transitions which over-approximate transitions of the concrete model, and *must* transitions, which under-approximate the concrete transitions. To ensure logic preservation, truth of universal formulas is then examined over may transitions, whereas truth of existential formulas is examined over must transitions. Dually for falsity when a 3-valued semantics is considered.

It was shown in [19, 8, 9] that must transitions are a source of incompleteness, in the sense that when limited to the use of must transitions, it is not always possible to construct a finite abstract model in which a property holds, even if it holds on the concrete model. Must transitions were also shown to behave badly in refinement in the sense of causing a loss of precision [23]. It was therefore suggested to model the must transitions of an abstract model as *hyper transitions*, which connect a single state to a *set* of state. Hyper transitions, first introduced in [17], were shown in [23] to prevent the loss of precision during refinement. They were also shown in [8, 9] to result in a *complete* abstraction framework for the fragment of the μ-calculus defined with greatest fixpoints only ([8] also introduces fairness and hence achieves completeness for the full μ-calculus). Following [23], we refer to such models, defined with may transitions and must hyper transitions, as *generalized kripke modal transition systems* (GTSs).

In this paper we investigate both the *precision* of ab-

stract models, and the *efficiency* of their model checking. We show that GTSs are not yet satisfactory in terms of precision. We suggest how to overcome their imprecision by using may hyper transitions. We then suggest an efficient abstract model checking algorithm that achieves the newly obtained maximal precision while avoiding the exponential blowup inherent by the use of hyper transitions.

Precision of an abstract model is measured by the extent to which it enables to verify or falsify formulas. Specifically, given an abstraction (S_A, γ), it is desirable to construct an abstract model over the states S_A in which as many formulas as possible have a definite value (true or false). With this purpose in mind, we address the allegedly non-problematic *may* transitions. We show that while being good enough for completeness purposes [8, 9], they are in fact a source of *imprecision*. This might sound surprising, yet the explanation is simple: when completeness is investigated, the choice of the abstraction (S_A, γ) is left open. On the other hand, when precision is investigated, one is interested in how precise the model is for a *given* abstraction.

In order to elaborate further on the imprecision problem we need a more detailed description of abstract models. Typically, to ensure logic preservation, may transitions in an abstract model have to be such that whenever there is a concrete transition from a concrete state s_c to a concrete state s'_c, then every abstract state that represents s_c has to have a may transition to some abstract state that represents s'_c. Now, consider the following example.

Example 1.1 Suppose that we are interested in verifying the formulas $\Box p$ ("all the successors satisfy p") and $\Box q$ ("all the successors satisfy q") in a concrete state s_c that has exactly one successor s'_c satisfying both p and q. Suppose further that we are given an abstraction in which s_c is represented by s_a, and no other concrete state is represented by s_a. Moreover suppose that s'_c is represented by two abstract states: s_{1a} that satisfies p but has an indefinite value on q, and s_{2a} that satisfies q but has an indefinite value on p. Fig. 1 illustrates this setting. Then at least one of the transitions (s_a, s_{1a}) or (s_a, s_{2a}) has to be included as a may transition in the abstract model to ensure logic preservation. However, choosing the first will enable verification of $\Box p$, but not $\Box q$, choosing the second will enable the opposite, and including both transitions will prevent verification of both properties. In other words, no choice of a may transition relation will enable verification of both $\Box p$ and $\Box q$. In particular, none of them will enable to verify $\Box p \wedge \Box q$.

Intuitively, in order to achieve the desired precision in the above example one has to consider both may transitions, but each of them has to be considered separately. We therefore suggest a new class of models, called *hyper kripke modal transition systems* (HTSs), in which may transitions are also replaced by hyper transitions, with the meaning that each
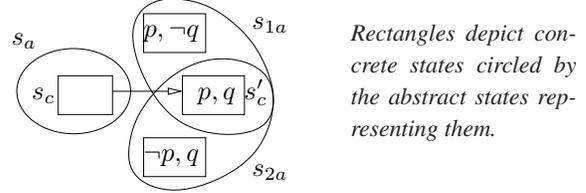
Figure 1. Illustration of Example 1.1

outgoing may hyper transition of an abstract state s_a over approximates *all* the concrete transitions of the states represented by s_a, but several different approximations (may hyper transitions) can be used. Other possible solutions involve changing the abstract state space, for example by some kind of completion that improves the states precision (e.g. [6]). However, we wish to "make the most" of the *given* abstract states.

Using HTSs as abstract models solves the problem demonstrated by Example 1.1, but one may wonder if there are other imprecision sources that HTSs do not address. To answer this question and justify the use of HTSs as abstract models we show how to construct, given *any* abstraction, an HTS which is as *precise* as the abstraction allows. We formalize this by introducing a new notion of precision which only depends on the abstraction (S_A, γ) itself and not on the class of abstract models. This enables us to claim that the constructed HTS is as precise as possible, among all possible abstract models with a standard 3-valued semantics.

HTSs therefore settle the issue of precision, as they allow maximal precision. Yet, in terms of efficiency, their use only increases the problem which already exists in GTSs due to the must hyper transitions: In general, the number of hyper-transitions might be exponential in the number of states in the abstract model. Thus, the need to handle hyper transitions makes both the construction of an abstract model and its model checking computationally expensive.

This problem was already addressed in [23] with respect to must hyper transitions. They suggested an automatic construction of abstract GTSs within an abstraction-refinement framework for CTL. Their algorithm starts with some initial model which consists of (mostly) ordinary transitions. Then, during refinement, when the abstract states are split, instead of computing all must hyper transitions, they "learn" must hyper transitions from must transitions (and hyper transitions) that existed in the previous iteration. Thus, in many cases they avoid the exponential blowup.

This approach suffers from several disadvantages. First, it only works as part of an abstraction-refinement loop. More importantly, the produced must hyper transitions are not necessarily the ones that are needed in practice for a specific proof. Some of them might be redundant, as they are irrelevant for proving the desired property, whereas others which are needed to verify the desired property might not

be produced, making the model not precise enough.

We wish to obtain efficiency without compromising the precision that an HTS enables to get. We achieve this goal for the alternation free fragment of the μ-calculus. The ability to do this results from the fact that the precise HTS is precise w.r.t. *every* μ-calculus formula, whereas we are only interested in *one* particular (alternation-free) formula. This can be exploited to save unnecessary efforts.

Suppose, for example, that we wish to check the formula $\Diamond p$ ("there is a successor that satisfies p") in an abstract state s_a, for which the number of outgoing must hyper-transitions in the precise HTS is exponential in the number of states. If we want the abstract model to be as precise as possible w.r.t every μ-calculus formula, we might need to consider all of the hyper transitions (or at least the minimal ones). However, for the verification of $\Diamond p$ in s_a it suffices to consider a single must hyper transition (under approximation), in which all the target states satisfy p. In other words, w.r.t. the particular formula, a HTS that contains only the relevant must hyper transition is as precise as the precise HTS. Similar reasoning applies to may hyper transitions. The question is how to find these designated hyper transitions and avoid the computation of the rest.

The key idea is to construct the HTS *during* the model checking, and thus avoid the (exponential) construction of the precise HTS. We use the model checking to guide the computation of hyper transitions, by checking for the existence of hyper transitions only when needed.

We obtain an automatic construction of an abstract model which is as precise as the precise HTS w.r.t. the property of interest, along with a model checking algorithm with complexity $O(|S_A|^2 \times |\varphi|)$. This is comparable to the model checking complexity of the alternation free μ-calculus over models limited to ordinary transitions (recall that the number of ordinary transitions over $|S_A|$ states is $O(|S_A|^2)$), except that our algorithm also ensures maximal precision. We believe that similar techniques can be used to develop precise abstract model checking algorithms for the full μ-calculus, with complexity comparable to model checking of ordinary transition systems.

We emphasize that while may hyper transitions are not always necessary for maximal precision, must hyper transitions are in fact mandatory for completeness. This demonstrates the importance of such an algorithm, which handles hyper transitions efficiently. Moreover, our approach can be beneficial even in cases where ordinary transitions suffice for the construction of a precise abstract model for a formula. This is because such constructions are usually expensive as they require finding best approximations of the concrete transitions (e.g. [7]). In our approach, instead of computing best approximations, the model checking algorithm wisely chooses candidates for which we perform the simpler task of checking if the *given* candidate is a correct

approximation – not necessarily the best one.

To complete the discussion, we show how to use our abstract model checking within an abstraction-refinement framework, and show that the refinement has the desirable property of monotonicity, meaning that the precision of an abstract model never decreases as a result of refinement.

To sum up, the main contributions of this paper are:

- New simple definition of precision of abstract models, which measures the precision w.r.t. the abstraction (S_A, γ), independently of the class of models used.
- New class of abstract models and a construction of an abstract model of this class which is precise w.r.t. any given abstraction.
- New abstract model checking algorithm for the alternation free μ-calculus that achieves maximal precision for a given formula, while remaining quadratic in the number of abstract states. This algorithm results in a more precise abstraction-refinement framework.

Related Work. Precision of modal (or mixed) transition systems, with ordinary may and must transitions, is studied in [4, 7, 21]. They suggest constructions of such abstract models which are most precise among all models from this *specific* class. In [23] GTSs are considered. They suggest a construction of an abstract GTS (with must hyper transitions) and show that it is most precise among all models produced by a *specific* construction method. In contrast to the above, we define a general notion of precision, which is independent not only of the construction method, but also of the class of abstract models.

A similar approach is taken in [13]. They refer to multivalued concrete models and use an abstract semantics which is more general than the 3-valued semantics. They also define precision w.r.t. the abstraction itself, but then use (multi-valued) transition systems as abstract models, which causes a loss of precision. Our work, on the other hand, suggests a class of models that achieves maximal precision for the case of 2-valued concrete models. Moreover, [13] defines precision within the framework of abstract interpretation [5] and assumes that every set of concrete states has a unique most precise abstract state that describes it. We do not impose any restrictions on the abstraction and provide a simple, "stand alone", definition of precision.

The work of [9] also measures the precision of an abstract model by comparison to the precision of the abstraction. They define the precision of an abstraction (S_A, γ) in terms of a game over the concrete model. Their definition considers abstract states as precise in less cases than our definition. In particular, the abstract state s_a from Example 1.1 is not considered precise for $\Box p$ by their definition (when translating it to logic terms), although as demonstrated by Example 1.1, it *does* carry enough information to verify $\Box p$ in the (only) concrete state it represents. Using

this stronger definition they show that the construction of an abstract GTS, which is also suggested in [23], results in a precise abstract model. This is in contrast to our result that shows that GTSs do not allow maximal precision, since we measure the precision of a model compared to a more general definition of precision of an abstraction. As a consequence, when pursuing precision w.r.t. our definition, we get abstract models which are strictly more precise.

[10] refers to precision with a different motivation. They suggest how to define the abstraction (S_A, γ) after refinement in order to maintain precision of an abstract model after refinement. Thus, they measure precision only w.r.t. the precision before refinement and not independently.

A different approach to precision pursued in [2, 11] uses a more precise 3-valued semantics, referred to as the *thorough semantics*. This semantics gives more definite answers than the standard 3-valued semantics, at the expense of increasing the complexity of model checking. Namely, the resulting model checking problem has the same complexity as satisfiability. We are interested in an effective framework, thus we use the standard 3-valued semantics, which is less precise, but enjoys a better model checking complexity. We note that the imprecision problem described in this paper still exists even if the thorough semantics is used.

May hyper transitions resemble the de-focus operations of [8]. We give them a new motivation and use.

In terms of model checking in the presence of hyper transitions, [9] shows that the model checking problem for GTSs is reducible to concrete model checking in linear time (and logarithmic space) in the size of the GTS. Yet, the GTS itself might be of size exponential in the size of the abstract state space S_A (due to the existence of hyper transitions). Thus the overall complexity is exponential.

Our approach in which we construct the abstract model during the model checking has some resemblance to the work of [20]. They perform reachability analysis, where they execute the concrete transitions, while storing abstract versions of the concrete states that are visited. Their approach is limited to falsification of safety properties, as they consider only an under approximation of the concrete model. Our work, on the other hand, is suitable for any property expressed in the alternation free μ-calculus, and is based on a 3-valued setting which enables both verification and falsification.

2. Preliminaries

μ-calculus. [15] Let AP be a finite set of atomic propositions and $\mathcal{V}$ a set of propositional variables. We define the set Lit of literals over AP to be the set $AP \cup \{\neg p : p \in AP\}$. We identify $\neg\neg p$ with p. The logic μ-*calculus* in *negation normal form* is defined as follows:

$$\varphi ::= l \mid Z \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \mu Z.\varphi \mid \nu Z.\varphi$$

where $l \in Lit$ and $Z \in \mathcal{V}$. μ denotes a least fixpoint, whereas ν denotes greatest fixpoint. Let $\mathcal{L}_\mu$ denote the set of *closed* formulas generated by the above grammar, where the fixpoint quantifiers μ and ν are variable binders. We will also write η for either μ or ν. Furthermore we assume that formulas are well-named, i.e. no variable is bound more than once in any formula. Thus, every variable Z *identifies* a unique subformula $fp(Z) = \eta Z.\psi$ of φ, where the set $Sub(\varphi)$ of *subformulas* of φ is defined in the usual way.

We also consider the *alternation-free* fragment of the μ-calculus, denoted $\mathcal{L}_\mu^0$, where no nesting of fixpoints is allowed. Namely, $\varphi \in \mathcal{L}_\mu^0$ if for every subformula $\eta Z.\psi \in Sub(\varphi)$, no variable other than Z occurs freely in ψ.

Concrete Semantics. Concrete systems are typically modelled as *Kripke structures*. A Kripke structure [3] is a tuple $M = (S, R, L)$, where S is a (possibly infinite) set of states, $R \subseteq S \times S$ is a transition relation, which must be *total*, and $L : S \to 2^{Lit}$ is a labeling function, such that for every state s and every $p \in AP$, exactly one of p and $\neg p$ is in $L(s)$.

The *concrete semantics* $[\![\varphi]\!]^M$ of a formula $\varphi \in \mathcal{L}_\mu$ w.r.t. a Kripke structure $M = (S, R, L)$ is a mapping from S to $\{tt, ff\}$. $[\![\varphi]\!]^M(s) = tt$ $(= ff)$ means that the formula φ is true (false) in the state s of the Kripke structure M. Intuitively, in this context $\Box$ stands for "all successors", whereas $\Diamond$ stands for "exists a successor".

2.1. Abstraction Framework

Let M_C be a concrete Kripke structure with a set of concrete states S_C. An *abstraction* (S_A, γ) for S_C consists of a finite set of *abstract states* S_A and a total *concretization function* $\gamma : S_A \to 2^{S_C}$ that maps each abstract state to the (nonempty) set of concrete states it represents. Every $s_c \in S_C$ is represented by some $s_a \in S_A$.

The abstract states provide descriptions of the concrete states. The other components of the model M_C then also need to be lifted into the abstract world. Several classes of abstract models have been suggested for this purpose.

A *class of models* consists of some form of a transition system. It is accompanied with a *semantics* for the logic of interest, in our case the μ-calculus, over models from the class, and some *preservation relation* $\preceq$ between states that ensures preservation of the logic. An *abstract model* for M_C is then a model M_A from the class, over S_A, in which $(M_C, s_c) \preceq (M_A, s_a)$ whenever $s_c \in \gamma(s_a)$.

We are particularly interested in classes of abstract models that use a 3-valued semantics. The *3-valued semantics* [1] of a formula in a model M enables preservation of both satisfaction (tt) and refutation (ff) from an abstract model to the concrete one. In addition, a new truth value, $\bot$, is introduced, meaning that the truth value over the concrete model is unknown and can be either tt or ff. Such a 3-valued semantics was suggested for various classes of

abstract models (e.g. [14, 12, 23]). We define a *generic 3-valued semantics* that generalizes these definitions. We refer to classes of models defined with such a 3-valued semantics, where the preservation relation ensures preservation of both tt and ff, as *3-valued classes*.

A 3-valued class defines, for each model M from the class, sets $l^M \in 2^S$, for every $l \in Lit$, and operators $\square^M, \diamondsuit^M : 2^S \to 2^S$. These definitions are given in terms of the components of M, with the requirements that l^M and $(\neg l)^M$ are disjoint and the operators $\square^M$ and $\diamondsuit^M$ are monotone w.r.t. set inclusion. The 3-valued semantics for the class is then defined by the following definition.

Definition 2.1 (Generic 3-Valued Semantics) *Let M be a model from a 3-valued class. The tt-set $[\![\varphi]\!]_{tt}^M \subseteq S$ and ff-set $[\![\varphi]\!]_{ff}^M \subseteq S$ of $\varphi \in \mathcal{L}_\mu$ over M are defined inductively in the style of [1] (based on Kleene's 3-valued logic for $\wedge$ and $\vee$, and with the standard definition for fixpoints [1]), except that the definition for formulas of the form $l \in Lit$, $\square\psi$, or $\diamondsuit\psi$ depends on the particular class of M, as follows.*

$$[\![l]\!]_{tt}^M = l^M, \qquad [\![l]\!]_{ff}^M = (\neg l)^M$$
$$[\![\square\psi]\!]_{tt}^M = \square^M([\![\psi]\!]_{tt}^M), \qquad [\![\square\psi]\!]_{ff}^M = \diamondsuit^M([\![\psi]\!]_{ff}^M)$$

and dually for $\diamondsuit\psi$ when exchanging $\square$ and $\diamondsuit$.

If for every $\varphi \in \mathcal{L}_\mu$, $[\![\varphi]\!]_{tt}^M \cap [\![\varphi]\!]_{ff}^M = \emptyset$, then M is consistent. The 3-valued semantics of $\varphi \in \mathcal{L}_\mu$ over M, denoted $[\![\varphi]\!]_3^M$, is then defined to be a mapping $S \to \{tt, ff, \bot\}$:

$$[\![\varphi]\!]_3^M(s) = \begin{cases} tt, & \text{if } s \in [\![\varphi]\!]_{tt}^M \\ ff, & \text{if } s \in [\![\varphi]\!]_{ff}^M \\ \bot, & \text{otherwise} \end{cases}$$

Note that if M is an abstract model, preservation of both tt and ff of the $\mathcal{L}_\mu$ from M to the concrete model guarantees that M is consistent.

An example of a 3-valued class of models is the class of Generalized Kripke Modal Transition Systems described below with generalized mixed simulation as a relation that ensures logic preservation.

Generalized Kripke Modal Transition Systems.

Definition 2.2 *Given a set of states S, a* hyper-transition *is a pair (s, A) where $s \in S$ and $A \subseteq S$ is a nonempty set.*

Definition 2.3 *[23] A Generalized Kripke Modal Transition System (GTS) is a tuple $M = (S, R^+, R^-, L)$, where S is defined as before, R^-, R^+ are may and must transition relations s.t. $R^- \subseteq S \times S$ is total and $R^+ \subseteq S \times 2^S$. $L : S \to 2^{Lit}$ is a labeling function s.t. for every state s and $p \in AP$, at most one of p and $\neg p$ is in $L(s)$.*

3-Valued Semantics for GTSs. For a GTS $M = (S, R^+, R^-, L)$, we define $l^M, \square^M, \diamondsuit^M$ as follows. For every $l \in Lit$, $l^M = \{s \mid l \in L(s)\}$. For every $U \subseteq S$: $\square^M(U) = \{s \mid \forall t \in S, \text{ if } sR^- t \text{ then } t \in U\}$, and

$\diamondsuit^M(U) = \{s \mid \exists A \subseteq S \text{ s.t. } sR^+ A \text{ and } A \subseteq U\}$. When integrated into Definition 2.1 this results in a 3-valued semantics. In particular, for a consistent GTS the definition for formulas of the form $l \in Lit$, $\square\psi$ or $\diamondsuit\psi$ results in

$[\![l]\!]_3^M(s) = tt$ if $l \in L(s), ff$ if $\neg l \in L(s)$, and $\bot$ otherwise.

$$[\![\square\psi]\!]_3^M(s) = \begin{cases} tt, & \text{if } \forall t \in S, \text{ if } sR^- t \text{ then } [\![\psi]\!]_3^M(t) = tt \\ ff, & \text{if } \exists A \subseteq S \text{ s.t. } sR^+ A \text{ and} \\ & \qquad \forall t \in A : [\![\psi]\!]_3^M(t) = ff \\ \bot, & \text{otherwise} \end{cases}$$

$[\![\diamondsuit\psi]\!]_3^M(s)$ is defined dually when exchanging tt with ff.

Definition 2.4 (Generalized Mixed Simulation) *[23] Let $M_1 = (S_1, R_1^+, R_1^-, L_1)$ and $M_2 = (S_2, R_2^+, R_2^-, L_2)$ be two GTSs. We say that $H \subseteq S_1 \times S_2$ is a* generalized mixed simulation *from M_1 to M_2 if $(s_1, s_2) \in H$ implies:*

1. $L_2(s_2) \subseteq L_1(s_1)$.
2. *if $s_1 R_1^- s_1'$, then there is some $s_2' \in S_2$ s.t. $s_2 R_2^- s_2'$ and $(s_1', s_2') \in H$.*
3. *if $s_2 R_2^+ A_2$, then there is some $A_1 \subseteq S_1$ s.t. $s_1 R_1^+ A_1$ and $(A_1, A_2) \in H^{\forall\exists}$, where $(A_1, A_2) \in H^{\forall\exists} \Leftrightarrow \forall s_1' \in A_1 \exists s_2' \in A_2 : (s_1', s_2') \in H$.*

If there is a generalized mixed simulation H such that $(s_1, s_2) \in H$, we write $(M_1, s_1) \preceq (M_2, s_2)$.

In particular, Definition 2.4 can be applied to a (concrete) Kripke structure M_C and an (abstract) GTS M_A, by viewing the Kripke structure as a GTS where $R^- = R$, $R^+ = \{(s, \{s'\}) \mid (s, s') \in R\}$. For a Kripke structure the 3-valued semantics agrees with the concrete semantics. Thus, preservation of $\mathcal{L}_\mu$ formulas is guaranteed by the following theorem, which is adapted from [23] to $\mathcal{L}_\mu$.

Theorem 2.5 *For GTSs M_1 and M_2 with states s_1 and s_2 resp., if $(M_1, s_1) \preceq (M_2, s_2)$ then for every $\varphi \in \mathcal{L}_\mu$: $s_2 \in [\![\varphi]\!]_{tt}^{M_2} \Rightarrow s_1 \in [\![\varphi]\!]_{tt}^{M_1}$, and $s_2 \in [\![\varphi]\!]_{ff}^{M_2} \Rightarrow s_1 \in [\![\varphi]\!]_{ff}^{M_1}$.*

Construction of an Abstract GTS. Let $M_C = (S_C, R, L_C)$ be a (concrete) Kripke structure and (S_A, γ) an abstraction for S_C. An abstract GTS $M_A = (S_A, R^+, R^-, L_A)$ can be constructed as follows [23].

The labeling of an abstract state is defined in accord with the labeling of all the concrete states it represents. For $l \in Lit$, $l \in L_A(s_a)$ only if $\forall s_c$ if $s_c \in \gamma(s_a)$ then $l \in L_C(s_c)$. It is thus possible that neither p nor $\neg p$ are in $L_A(s_a)$.

The *may* transitions are computed by an [∃∃] rule such that every concrete transition is represented by them:

$$\exists s_c \in \gamma(s_a) \, \exists s_c' \in \gamma(s_a') \text{ s.t. } s_c R s_c' \Longrightarrow s_a R^- s_a'$$

The *must* hyper transitions, on the other hand, represent concrete transitions that are common to all the concrete states represented by the source abstract state. They are computed by an [∀∃∃] rule:

$$\forall s_c \in \gamma(s_a) \, \exists s_a' \in A_a \, \exists s_c' \in \gamma(s_a') \text{ s.t. } s_c R s_c' \Longleftarrow s_a R^+ A_a$$

---

[1]We omit the use of an environment for simplicity of the presentation.

Exact GTS. If the three implications above are replaced by "iff", then the labeling, may transitions and must hyper transitions are *exact*, resulting in the *exact GTS*.

Other constructions of abstract GTSs can also be suggested. For example, the construction of a mixed transition system from [7] within the framework of abstract interpretation can be extended to GTSs as well.

All the above constructions assure us that whenever $s_c \in \gamma(s_a)$, then $(M_C, s_c) \preceq (M_A, s_a)$. The generalized mixed simulation $H \subseteq S_C \times S_A$ is induced by γ as follows: $(s_c, s_a) \in H$ iff $s_c \in \gamma(s_a)$. Therefore, Theorem 2.5 guarantees preservation of $\mathcal{L}_\mu$ from M_A to M_C.

3. Increasing Precision

Let M_C be a concrete Kripke structure. In this section we are interested in the *precision* of the abstract model constructed for M_C with a given abstraction (S_A, γ).

Specifically, in Section 2 we described GTSs as a class of abstract models, along with constructions of abstract models from this class. We now ask the following questions: (1) Do the constructions of GTSs from Section 2 produce the most precise abstract model that we can hope for, given an abstraction? and more fundamentally: (2) Does the use of GTSs *enable* to express the most precise abstract model?

Of course, to answer these questions we first need to define what the most precise abstract model that we can hope for is, given an abstraction. We measure precision with respect to a 3-valued semantics. We therefore restrict the discussion to abstract models from 3-valued classes.

3.1. Precision of Abstract Models

We wish to capture maximal precision within the boundaries of the inductive 3-valued semantics as defined in Definition 2.1. When using this semantics, the verification or refutation of any $\mathcal{L}_\mu$ formula over an abstract model M_A boils down to manipulations of l^{M_A}, $\square^{M_A}(U_A)$, and $\lozenge^{M_A}(U_A)$ for various $l \in Lit$ and $U_A \subseteq S_A$. We therefore view a set $U_A \subseteq S_A$ as a new formula with the following semantics. Let $\gamma(U_A)$ stand for $\bigcup_{s_a \in U_A} \gamma(s_a)$. Then in a concrete model M_C, $[\![U_A]\!]_{tt}^{M_C} = \{s_c \mid s_c \in \gamma(U_A)\}$. In an abstract model M_A (from a 3-valued class), $[\![U_A]\!]_{tt}^{M_A} = U_A$. This makes the tt-sets of formulas of the form l, $\square U_A$, and $\lozenge U_A$ over M_A the building blocks of any model checking problem over M_A. As such, the precision of M_A is determined by its precision w.r.t. truth of such formulas.

In the spirit of [9] we first define the precision of an *abstraction* w.r.t. such formulas. This is the precision that a precise abstract *model* will then be expected to match.

Definition 3.1 (Precision of Abstractions) *Given an abstraction* (S_A, γ) *for* M_C *and a state* $s_a \in S_A$*, we say that*

s_a *fulfills* $\varphi = l$, $\square U_A$ *or* $\lozenge U_A$*, for* $l \in Lit$ *and* $U_A \subseteq S_A$*, if* $\forall s_c \in \gamma(s_a) : [\![\varphi]\!]^{M_C}(s_c) = tt$.

Note that this definition is independent of the class of abstract models, as it is meant to capture the precision of the abstraction itself, in terms of the information carried within the abstract states. For example, for the abstraction to reflect the fact that $\square U_A$ holds in an abstract state s_a (meaning it holds in all the concrete states it represents), it has to be the case that *all* the concrete states in $\gamma(s_a)$ share the property that all of their outgoing (concrete) transitions are to $\gamma(U_A)$, which is the "description" of U_A in the concrete world.

Definition 3.2 (Precision of Models) *An abstract model* M_A *for* M_C *(from some 3-valued class) is* precise *w.r.t.* (S_A, γ) *if for all* $s_a \in S_A$*,* $l \in Lit$ *and* $U_A \subseteq S_A$*: whenever* s_a *fulfills* $\varphi = l$, $\square U_A$ *or* $\lozenge U_A$*, then* $s_a \in [\![\varphi]\!]_{tt}^{M_A}$.

Thus whenever the information about l, $\square U_A$, or $\lozenge U_A$ exists in the abstract states, a precise abstract model enables to see that. To formalize the generality of Definition 3.2, we extend Definition 3.1 to more complicated formulas and to falsification, following the 3-valued semantics. We then show that whenever an abstract model is precise w.r.t. truth of $l, \square U_A, \lozenge U_A$, it is also precise w.r.t. any other formula.

Definition 3.3 *Let* $\mathcal{A} = (S_A, \gamma)$ *be an abstraction. We define an* abstract semantics $[\![\varphi]\!]_3^{\mathcal{A}}$ *by using the generic 3-valued semantics (see Definition 2.1) with the following definitions of* $l^{\mathcal{A}} \in 2^{S_A}$*, and* $\square^{\mathcal{A}}, \lozenge^{\mathcal{A}} : 2^{S_A} \to 2^{S_A}$*. For* $l \in Lit$*:* $l^{\mathcal{A}} = \{s_a \mid s_a$ *fulfills* $l\}$*. For* $U_A \subseteq S_A$*:* $\square^{\mathcal{A}}(U_A) = \{s_a \mid s_a$ *fulfills* $\square U_A\}$*, and* $\lozenge^{\mathcal{A}}(U_A) = \{s_a \mid s_a$ *fulfills* $\lozenge U_A\}$*. We say that* $s_a \in S_A$ *enables verification (falsification) of* $\varphi \in \mathcal{L}_\mu$ *if* $[\![\varphi]\!]_3^{\mathcal{A}}(s_a) = tt$ (ff).

The abstract semantics is well defined since whenever $s_a \in [\![\varphi]\!]_{tt}^{\mathcal{A}}$ (resp. $[\![\varphi]\!]_{ff}^{\mathcal{A}}$), then $\forall s_c \in \gamma(s_a) : [\![\varphi]\!]^{M_C}(s_c) = tt$ (resp. ff). This ensures that $[\![\varphi]\!]_{tt}^{\mathcal{A}} \cap [\![\varphi]\!]_{ff}^{\mathcal{A}} = \emptyset$.

For example, by this definition s_a enables verification of $\varphi = \square \psi$ iff s_a fulfills $\square U_A$ for some $U_A \subseteq S_A$ such that every $s_a' \in U_A$ enables verification of ψ.

Theorem 3.4 *Let* M_A *be an abstract model for* M_C *(from some 3-valued class) which is* precise *w.r.t.* (S_A, γ)*. Then whenever* $s_a \in S_A$ *enables verification (falsification) of* $\varphi \in \mathcal{L}_\mu$*, then* $[\![\varphi]\!]_3^{M_A}(s_a) = tt$ (ff).

The following theorem ensures that an abstract model which is precise w.r.t. the abstraction is also most precise when compared to other abstract models, provided that their class has the following property. A 3-valued class of models is *structural* if its definitions of $\square^M, \lozenge^M : 2^{S_A} \to 2^{S_A}$ ensure that for every $U_A \subseteq S_A$, whenever $s_a \in \square^M(U_A)$, then for every $s_c \in \gamma(s_a)$ all the concrete successors of s_c

are in $\gamma(U_A)$. Similarly, whenever $s_a \in \Diamond^M(U_A)$, then every $s_c \in \gamma(s_a)$ has a successor in $\gamma(U_A)$. Intuitively, for $\Box^M$ and $\Diamond^M$ to maintain such consistency with the concrete world, they have to be based on some (structural) abstract description of the concrete transitions in the abstract model. For example, GTSs and their variants are such classes.

Theorem 3.5 *Let M_A, M'_A be two abstract models for M_C (from possibly different 3-valued classes) based on an abstraction (S_A, γ). If M_A is precise w.r.t. (S_A, γ) and the class of M'_A is structural, then for every $s_a \in S_A$ and every $\varphi \in \mathcal{L}_\mu$: $[\![\varphi]\!]_3^{M'_A}(s_a) \neq \bot \Rightarrow [\![\varphi]\!]_3^{M_A}(s_a) = [\![\varphi]\!]_3^{M'_A}(s_a)$.*

Now, equipped with formal definitions of precision, we go back to our questions about the precision of GTSs.

Theorem 3.6 *If the abstraction (S_A, γ) partitions the concrete states, i.e. for each $s_a, s'_a \in S_A : \gamma(s_a) \cap \gamma(s'_a) = \emptyset$, then the exact GTS from section 2 is precise w.r.t. (S_A, γ).*

However, in many cases it might be desirable to gather the concrete states into non-disjoint sets, as this can reduce the size of the abstract state space that enables verification or falsification of the desired property. We show that in this general setting, the answer to both questions is "no".

3.2. May Transitions as a Source of Imprecision

As demonstrated by Example 1.1, when the given abstract states do not represent disjoint sets of concrete states, the may transitions can become a source of imprecision. In this example there is *no* abstract GTS for M_C over S_A that will enable verification of both $\Box p$ and $\Box q$ in s_a. This is while the abstraction *does* enable verification of both $\Box p$ and $\Box q$ in s_a (see Definition 3.3). Thus, none of the possible GTSs is precise w.r.t. the given abstraction.

Theorem 3.7 *GTSs do not always suffice for the construction of a precise abstract model w.r.t. a given abstraction.*

We emphasize that this imprecision is not limited to a certain construction. Indeed, the construction of the exact GTS from Section 2 is simplistic, as it might introduce redundancy in the may transitions (for example, in Example 1.1 both may transitions would be included). Yet, Theorem 3.7 holds even for optimized constructions that avoid redundant may transitions (e.g. in the style of [7]).

It can be shown that the imprecision results from the may transitions and not from the other components of the GTS. This is because whenever the abstraction enables verification of $l \in Lit$ or $\Diamond U_A$, so does the exact GTS, which implies that the labeling and the must hyper transitions (used for verification of such formulas) are precise enough.

More than that, analyzing Example 1.1 shows that the imprecision arises when there is no "best" choice of may

transitions, in which case one needs to consider *all* of their (incomparable) possibilities to achieve maximal precision. Unfortunately, a GTS does not enable to do that. We therefore suggest to model the may transitions as hyper transitions as well, with the meaning that each may hyper transition $(s_a, A_a) \in S_A \times 2^{S_A}$ provides *some* over approximation of *all* the outgoing transitions of the concrete states represented by s_a.

3.3. Hyper Kripke Modal Transition Systems

This brings us to the new class of abstract models that we suggest to be used in order to obtain maximal precision.

Definition 3.8 *A* Hyper Kripke Modal Transition System *(**HTS**) is a tuple $M = (S, R^+, R^-, L)$, where S, L, R^+ are defined as before, and $R^- \subseteq S \times 2^S$ (not necessarily total).*

3-Valued Semantics for HTSs. To adapt the 3-valued semantics of $\mathcal{L}_\mu$ for HTSs we redefine $\Box^M$. For every $U \subseteq S$: $\Box^M(U) = \{s \mid \exists A \subseteq S \text{ s.t. } sR^-A \text{ and } \forall t \in A : t \in U\}$. This changes the definition for $\Box\psi$ in a consistent HTS to:

$$[\![\Box\psi]\!]_3^M(s) = \begin{cases} \text{tt,} & \text{if } \exists A \subseteq S \text{ s.t. } sR^-A \text{ and} \\ & \qquad \forall t \in A : [\![\psi]\!]_3^M(t) = \text{tt} \\ \text{ff,} & \text{if } \exists A \subseteq S \text{ s.t. } sR^+A \text{ and} \\ & \qquad \forall t \in A : [\![\psi]\!]_3^M(t) = \text{ff} \\ \bot, & \text{otherwise} \end{cases}$$

and dually for $[\![\Diamond\psi]\!]_3^M(s)$ when exchanging tt with ff.

Thus, in order to evaluate a $\Box\psi$ formula to tt, instead of requiring that *all* the may transitions are to states that satisfy ψ, we now require that there *exists* a may hyper transition such that *all* the states within the target set satisfy ψ.

A GTS, and thus also a Kripke structure, can be viewed as a HTS, where every state has exactly *one* outgoing may hyper transition, whose target set consists of the target states of *all* of its (ordinary) may transitions. Preservation of $\mathcal{L}_\mu$ between HTSs (and in particular between an HTS and a Kripke structure) is guaranteed by the following relation.

Definition 3.9 (Hyper Mixed Simulation) *Let $M_1 = (S_1, R_1^+, R_1^-, L_1)$ and $M_2 = (S_2, R_2^+, R_2^-, L_2)$ be two HTSs. $H \subseteq S_1 \times S_2$ is a* hyper mixed simulation *from M_1 to M_2 if $(s_1, s_2) \in H$ implies the requirements of Definition 2.4, except that requirement 2 is replaced by:*

2. *if $s_2 R_2^- A_2$, then there is some $A_1 \subseteq S_1$ s.t. $s_1 R_1^- A_1$ and $(A_1, A_2) \in H^{\forall\exists}$, where as before: $(A_1, A_2) \in H^{\forall\exists} \Leftrightarrow \forall s'_1 \in A_1 \exists s'_2 \in A_2 : (s'_1, s'_2) \in H$.*

If there is a hyper mixed simulation H such that $(s_1, s_2) \in H$, we write $(M_1, s_1) \preceq (M_2, s_2)$.

Intuitively, there can be less may hyper transitions in M_2 but each one has to over approximate *some* hyper transition in M_1. Thus, if some may hyper transition was used to verify $\Box\psi$ in M_2, then the may hyper transition that it over

approximates can be used to verify it in M_1. Note that a may hyper transition of M_1 that has no representation in M_2 can only cause formulas with a definite value in M_1 to be indefinite in M_2 and not vice versa.

Theorem 3.10 *For HTSs M_1 and M_2 with states s_1 and s_2 resp., if $(M_1, s_1) \preceq (M_2, s_2)$ then for every $\varphi \in \mathcal{L}_\mu$: $s_2 \in [\![\varphi]\!]_{\text{tt}}^{M_2} \Rightarrow s_1 \in [\![\varphi]\!]_{\text{tt}}^{M_1}$, and $s_2 \in [\![\varphi]\!]_{\text{ff}}^{M_2} \Rightarrow s_1 \in [\![\varphi]\!]_{\text{ff}}^{M_1}$.*

Construction of an Abstract HTS. Let $M_C = (S_C, R, L_C)$ be a (concrete) Kripke structure. Given an abstraction (S_A, γ) for it, an abstract model in the form of a HTS $M_A = (S_A, R^+, R^-, L_A)$, can be constructed as before with the exception that R^- now consists of hyper transitions, constructed as follows. A may hyper transition $s_a R^- A_a$ exists only if an [∀∀∃] condition holds:

$$\forall s_c \in \gamma(s_a) \; \forall s_c' \; [\; s_c R s_c' \Rightarrow \exists s_a' \in A_a \text{ s.t. } s_c' \in \gamma(s_a') \;]$$

That is, every outgoing may hyper transition of s_a over approximates *all* the concrete transitions of the states represented by s_a. An example of a "legal" may hyper transition is (s_a, A_a) for $A_a = \{s_a' \mid \exists s_c \in \gamma(s_a) \; \exists s_c' \in \gamma(s_a') \text{ s.t. } s_c R s_c'\}$. Note that the "only if" allows to include less hyper transitions than allowed by the rule. The following theorem formalizes the correctness of the construction.

Theorem 3.11 *Let M_C be a concrete Kripke structure over S_C, and let M_A be an HTS computed as described above based on an abstraction (S_A, γ) for S_C. Then whenever $s_c \in \gamma(s_a)$ then $(M_C, s_c) \preceq (M_A, s_a)$.*

For example, to verify $\Box p$ and $\Box q$ in Example 1.1, we include $(s_a, \{s_{1a}\})$ and $(s_a, \{s_{2a}\})$ as may hyper transitions.

Exact HTS. If the "only if" in the definition of may hyper transitions is replaced by "iff", the may hyper transitions are *exact*. If all components are exact, we get the *exact HTS*.

Theorem 3.12 *Let M_C be a Kripke structure and M_A^E the exact HTS computed as described above based on an abstraction (S_A, γ). Then M_A^E is precise w.r.t. (S_A, γ).*

4. Decreasing the Model Checking Cost

Using the exact HTS as an abstract model ensures maximal precision. Yet, it involves an exponential blowup. In this section we suggest an efficient model checking, which remains quadratic in the number of abstract states, and yet produces a result which is *as precise as possible* with respect to a specific property.

From now on, we restrict the discussion to the alternation free fragment of the μ-calculus. Let M_C be a concrete Kripke structure and $\varphi \in \mathcal{L}_\mu^0$ a formula that we wish to check in some state s_c of M_C. Moreover, suppose that we are given a (finite) abstraction (S_A, γ). All the abstract

$$\frac{s \vdash \psi_0 \vee \psi_1}{s \vdash \psi_i} : i \in \{0, 1\} \qquad \frac{s \vdash \psi_0 \wedge \psi_1}{s \vdash \psi_i} : i \in \{0, 1\}$$

$$\frac{s \vdash \eta Z.\psi}{s \vdash Z} \qquad \frac{s \vdash Z}{s \vdash \psi} : \text{if } fp(Z) = \eta Z.\psi$$

$$\frac{s \vdash \Diamond \psi}{t \vdash \psi} : s \widetilde{R} t \qquad \frac{s \vdash \Box \psi}{t \vdash \psi} : s \widetilde{R} t$$

Figure 2. Rules for product graph construction

states that represent s_c are candidates to enable verification or falsification of φ in s_c. We therefore refer to them as *designated* states. Our purpose is to evaluate φ in all these designated abstract states in the exact HTS M_A^E.

Our algorithm is based on a generalization of the game-based model checking suggested in [22] for CTL over abstract models with ordinary may and must transitions. We omit the details of the game, but continue with the game-graph, to which we refer as the *product graph*.

Product Graph. The product graph presents all the information "relevant" for the model checking: Every node in the graph is labeled by $s_a \vdash \psi$, where s_a is an abstract state and ψ is a subformula of φ, indicating that the value of ψ in s_a is relevant for determining the model checking result. The outgoing edges of a node $s_a \vdash \psi$ can be seen as defining "subgoals" for the goal of checking ψ in s_a.

Formally, let $\varphi \in \mathcal{L}_\mu^0$ be a formula, S_A a set of states, $S_d \subseteq S_A$ a set of designated states in which we want to evaluate φ, and $\widetilde{R} \subseteq S_A \times S_A$ a total transition relation. $\widetilde{R}$ is meant to provide a basic description of the possible transitions between states (we will soon see how it is obtained). The *product graph* $G_{S_d, \widetilde{R}, \varphi}$, or in short G, is a graph (N, E) with a set of nodes $N \subseteq S_A \times Sub(\varphi)$ and a set of edges $E \subseteq N \times N$, defined as follows. The *initial nodes* $N_0 \subseteq N$ consist of $S_d \times \{\varphi\}$. The (rest of the) nodes and the edges are defined by the rules of Fig. 2, with the meaning that whenever $n \in N$ is of the form of the upper part of the rule, then the result in the lower part of the rule is also a node $n' \in N$ and $(n, n') \in E$.

The nodes of G are classified as $\wedge, \vee, \Box, \Diamond$ nodes, based on their subformuals. Nodes whose subformula is a literal are *terminal nodes* (they have no outgoing edges). Nodes whose subformulas are of the form Z or $\eta Z.\psi$ are *deterministic* – they have exactly one son.

Each strongly connected component (SCC) in G which is non-trivial, i.e. has at least one edge, contains exactly one free fixpoint variable $Z \in \mathcal{V}$, called a *witness*. If $fp(Z) = \mu Z.\psi$, then Z is a μ-witness. Otherwise it is a ν-witness.

Coloring Algorithm. To determine the model checking result, a coloring algorithm is applied on the product graph with the purpose of labeling each node $n = s_a \vdash \psi$ in it by T, F, ? depending on the value of ψ in the state s_a in M_A^E.

The coloring algorithm of [22] processes the product

graph bottom-up by iterating two phases: In the sons-coloring phase, a node is colored based on the colors of its sons by rules which reflect the 3-valued semantics of the logic. In the witness-coloring phase a special procedure is applied to handle cycles (non trivial SCCs) in the graph.

As for our algorithm, for the sake of the explanation, suppose first that we construct the product graph based on M_A^E (of course, eventually the point will be to avoid the construction of M_A^E). $\widetilde{R}$ will then simply be the set $\widetilde{R}^E = \{(s_a, s_a') \mid s_a' \in A_a \text{ and } (s_a R^- A_a \text{ or } s_a R^+ A_a)\}$, where R^- and R^+ are the transition relations of M_A^E. In this case we also define may and must hyper-sons in G: If $n = s \vdash \heartsuit \psi \in N$ for $\heartsuit \in \{\Box, \Diamond\}$ and $sR^- A$ ($sR^+ A$), then $B = A \times \{\psi\} \subseteq N$ is a *may (must) hyper-son* of n.

The coloring can be extended to handle hyper sons in the same way that the 3-valued semantics is extended to handle hyper transitions. For example, a $\Box$-node will be colored by F iff it has a must hyper-son whose nodes are all colored by F. It will be colored by T iff it has a may hyper-son whose nodes are all colored by T. Otherwise it will be colored ?. Dually for a $\Diamond$-node. Yet, instead of considering *all* the hyper sons and checking if any of them justifies coloring the node, we suggest to use the information gathered so far in the bottom-up coloring to perform this check wisely.

For example, to color a $\Box$-node n by F, it suffices to check, whenever some son of n gets colored by F, if all of n's currently F-colored sons comprise a must hyper-son (i.e., their underlying states fulfill the $\forall \exists \exists$ rule). Similarly, to conclude that n should not be colored F, it suffices to check that n's currently F-colored sons along with the un-colored sons (if exist) do not form a must hyper-son. Thus, checking these candidates is as informative as checking *all* of the possible must hyper sons. Similar reasoning applies to may hyper sons. This leads us to the following algorithm, where M_A^E is *not* constructed in advance.

4.1. Optimized Abstract Model Checking

Let M_C be a concrete model, $s_c \in S_C$ a concrete state, $\varphi \in \mathcal{L}_\mu^0$ a formula that we wish to check in s_c, and (S_A, γ) an abstraction. The algorithm is as follows.

Product Graph Construction. Construct a *partial* HTS $\widetilde{M}_A = (S_A, \widetilde{R}, L_A)$, where L_A is defined as in the exact HTS, and $\widetilde{R} \subseteq S_A \times S_A$ is defined by $\widetilde{R} = \{(s_a, s_a') \mid \exists s_c \in \gamma(s_a) \exists s_c' \in \gamma(s_a') \text{ s.t. } s_c R s_c'\}$. This ensures that $\widetilde{R} \supseteq \widetilde{R}^E$. Construct the product graph $G_{S_d, \widetilde{R}, \varphi}$ based on $\varphi, S_A, \widetilde{R}$ as above, and $S_d = \{s_a \mid s_c \in \gamma(s_a)\}$.

Partition. $G_{S_d, \widetilde{R}, \varphi}$ is partitioned into Maximal Strongly Connected Components (MSCCs), denoted Q_i's, and a (to-tal) order $\leq$ is determined on them, s.t. for every $n \in Q_i$ and $n' \in Q_j$, $(n, n') \in E$ only if $Q_j \leq Q_i$. Such an order exists because the MSCCs form a directed acyclic graph.

Coloring. The following two phases are performed repeat-edly until all nodes are colored.

1. *Sons-coloring phase.* Apply the following rules until none is applicable.
   - A terminal node $s_a \vdash l$ is colored T if $l \in L_A(s_a)$, F if $\neg l \in L_A(s_a)$, and ? otherwise.
   - An $\wedge$-node ($\vee$-node) is colored by:
     – $T(F)$ if both its sons are colored $T(F)$.
     – $F(T)$ if it has a son that is colored $F(T)$.
     – ? if it has a son that is colored ? and the other is colored $\neq F(T)$.
   - A deterministic node is colored as its (only) son.
   - A $\Box$-node ($\Diamond$-node) is colored by:
     – $T(F)$ if its currently $T(F)$-colored sons form a may hyper son.
     – $F(T)$ if its currently $F(T)$-colored sons form a must hyper son.
     – ? if all of its sons are colored, yet none of the above holds.

2. *Witness-coloring phase.* If there are still uncolored nodes, let Q_i be the smallest MSCC w.r.t. $\leq$ that is not yet fully colored. Q_i is necessarily a non-trivial MSCC that has exactly one witness. Its uncolored nodes are colored according to the witness. For a μ-witness:

   (a) Repeatedly color ? each node in Q_i satisfying one of the following.
       - An $\wedge$-node ($\vee$-node) that both (at least one) of its sons are colored $\neq F$.
       - A deterministic node whose son is colored ?.
       - A $\Box$-node ($\Diamond$-node) whose F-colored sons along with its remaining uncolored sons do not form a must (may) hyper-son.

   (b) Color the remaining nodes in Q_i by F.

   The case where the witness is of type ν is dual, when exchanging F with T, $\wedge$ with $\vee$, and $\Box$ with $\Diamond$.

In each phase of the coloring, the rules will initially be checked once for every uncolored node, and later will only be checked when one of the sons of the node gets colored by an appropriate color. Several optimizations can be used.

Remark 4.1 *Checking if a set B of nodes forms a may or must hyper son of a $\Box$-node or a $\Diamond$-node n is performed by checking the $\forall \exists \exists$ or the $\forall \forall \exists$ condition (resp.) between the underlying states of the node n and the set of nodes B.*

Theorem 4.2 *Let M_A^E denote the exact HTS for M_C w.r.t. (S_A, γ). Let $G = G_{S_d, \widetilde{R}, \varphi}$ be the product graph produced by the algorithm. Then for every $n = s_a \vdash \varphi_1 \in G$ such that φ_1 is closed the following holds:*

1. *$[\![\varphi_1]\!]_3^{M_A^E}(s_a) = \text{tt}$ iff $n = s_a \vdash \varphi_1$ is colored by T.*

2. $[\![\varphi_1]\!]_3^{M_A^E}(s_a) = \mathrm{ff}$ *iff* $n = s_a \vdash \varphi_1$ *is colored by* F.

3. $[\![\varphi_1]\!]_3^{M_A^E}(s_a) = \perp$ *iff* $n = s_a \vdash \varphi_1$ *is colored by* ?.

Thus, for all nodes with closed formulas in the product graph, the coloring is as precise as model checking with M_A^E, even though M_A^E is *not* constructed by the algorithm. In particular, this is true for $N_0 = S_d \times \{\varphi\}$, and by the choice of S_d, we are guaranteed that whenever the abstraction is precise enough, at least one initial node will be colored by a definite color T or F, in which case by Theorems 4.2 and 3.10, $[\![\varphi]\!]^{M_C}(s_c) = \mathrm{tt}$ or ff respectively. Note, that it is impossible that some initial node will be colored T and another will be colored F. If all the initial nodes in the product graph are colored ?, then the result is indefinite.

Remark 4.3 *By considering the underlying hyper transitions of hyper sons computed by the algorithm, the final product graph induces an abstract HTS for M_C which is as precise as the exact HTS w.r.t. φ.*

Complexity. During all applications of the sons-coloring phase, the $\forall\exists\exists$ and the $\forall\forall\exists$ conditions are checked at most $|S_A|$ times for each node, as each node has at most $|S_A|$ sons, and between checks the set of candidates to comprise a hyper son is monotonically increasing. Similar analysis holds for phase 2a, with the difference that the sets of candidates to comprise a hyper son are monotonically decreasing. As the number of nodes in the product graph is $O(|S_A| \times |\varphi|)$, the total number of checks of the $\forall\exists\exists$ and the $\forall\forall\exists$ conditions is $O(|S_A|^2 \times |\varphi|)$. This is the dominant part which determines the model checking complexity.

5. Abstraction-Refinement

Our abstract model checking ensures maximal precision. Still, its result might be indefinite if the abstraction is not precise enough. In this case, refinement can be applied by splitting the abstract states, similarly to the refinement of [22] for models with ordinary transitions (with various optimizations that exploit the use of hyper transitions).

When refinement is introduced, monotonicity in the precision of the abstract models before and after the refinement is desirable, meaning that formulas that had a definite value before the refinement will not become indefinite after refinement [23]. This is guaranteed by the following theorem.

Theorem 5.1 (Monotonicity of HTSs) *Let M_A' and M_A be exact HTSs defined based on abstractions (S_A', γ') and (S_A, γ) resp., where (S_A', γ') is the result of splitting the states of (S_A, γ). Then whenever $s_a' \in S_A'$ is a substate of $s_a \in S_A$ then $(M_A', s_a') \preceq (M_A, s_a)$.*

Monotonicity implies that refinement of an exact HTS will never take us further from the (definite) result. In particular,

we will not "miss" the opportunity to get a definite result only due to excess refinement. Thus, our approach, which is as precise as using the exact HTS w.r.t. the desired property, will ensure the same. Recall that the same is not guaranteed when using ordinary must transitions [23].

If the concrete model is finite, an iterative abstraction-refinement is guaranteed to terminate with a definite answer.

References

[1] G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *CAV*, 1999.

[2] G. Bruns and P. Godefroid. Generalized model checking: Reasoning about partial state spaces. In *CONCUR*, 2000.

[3] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT press, 1999.

[4] R. Cleaveland, P. Iyer, and D. Yankelevich. Optimality in abstraction of model checking. In *SAS*, 1995.

[5] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, 1977.

[6] P. Cousot and R. Cousot. Abstract interpretation frameworks. *J. Log. Comput.*, 2(4), 1992.

[7] D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *TOPLAS*, 19(2), 1997.

[8] D. Dams and K. Namjoshi. The existence of finite abstractions for branching time model checking. In *LICS*, 2004.

[9] L. de Alfaro, P. Godefroid, and R. Jagadeesan. Three-valued abstractions of games: Uncertainty, but with precision. In *LICS*, 2004.

[10] P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based model checking using modal transition systems. In *CONCUR*, 2001.

[11] P. Godefroid and R. Jagadeesan. Automatic abstraction using generalized model checking. In *CAV*, 2002.

[12] P. Godefroid and R. Jagadeesan. On the expressiveness of 3-valued models. In *VMCAI*, 2003.

[13] A. Gurfinkel, O. Wei, and M. Chechik. Systematic construction of abstractions for model-checking. In *VMCAI*, 2006.

[14] M. Huth, R. Jagadeesan, and D. Schmidt. Modal transition systems: A foundation for three-valued program analysis. In *ESOP*, 2001.

[15] D. Kozen. Results on the propositional μ-calculus. *TCS*, 27, 1983.

[16] K. Larsen and B. Thomsen. A modal process logic. In *LICS*, 1988.

[17] K. Larsen and L. Xinxin. Equation solving using modal transition systems. In *LICS*, 1990.

[18] K. G. Larsen. Modal specifications. In *Automatic Verification Methods for Finite State Systems, Grenoble*, 1989.

[19] K. Namjoshi. Abstraction for branching time properties. In *CAV*, 2003.

[20] C. S. Pasareanu, R. Pelánek, and W. Visser. Concrete model checking with abstract matching and refinement. In *CAV'05*.

[21] D. A. Schmidt. Closed and logical relations for over- and under-approximation of powersets. In *SAS*, 2004.

[22] S. Shoham and O. Grumberg. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. In *CAV*, 2003. To appear in TOCL.

[23] S. Shoham and O. Grumberg. Monotonic abstraction-refinement for CTL. In *TACAS*, 2004.