3-Valued Abstraction: More Precision at Less Cost ⋆

Sharon Shoham and Orna Grumberg

Computer Science Department, Technion, Haifa, Israel

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 [2]. 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

⋆ A preliminary version of this paper appeared in [1].

Email addresses: sharonsh@cs.technion.ac.il (Sharon Shoham), orna@cs.technion.ac.il (Orna Grumberg).

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 [3]. 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* [4]. 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* [5,6] 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 [8–10] 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 [11]. 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 [12], were shown in [11] to prevent the loss of precision during refinement. They were also shown in [9,10] to result in a *complete* abstraction framework for the fragment of the μ-calculus defined with greatest fixpoints only ([9] also introduces fairness and hence achieves completeness for the full μ-calculus). Following [11], 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 abstract 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

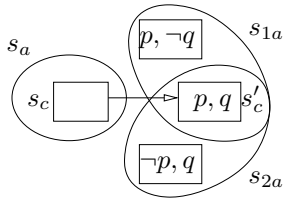
may hyper transitions. We then suggest an efficient abstract model checking algorithm that achieves the newly obtained maximal precision while avoiding the exponential blowup inherent by the use of hyper transitions.

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 [9,10], 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. This is because the may transitions are used to over approximate the concrete transitions. 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 in order to over approximate the concrete transition from s_c to s'_c. However, choosing the first transition 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 outgoing may hyper transition of an abstract state s_a over approximates *all* the concrete transitions of the



Rectangles depict concrete states circled by the abstract states representing them.

Fig. 1. Illustration of Example 1.1

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. [13,14]). However, in this work we do not follow such solutions since 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 [11] 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.

The approach of [11] 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 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 both may and must 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 [15,7,16]. They suggest constructions of such abstract models which are most precise among all models from this *specific* class. In [11] 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 [17]. They refer to multi-valued 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, [17] defines precision within the framework of abstract interpretation [18] 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 [10] 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 [11], 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. They thus allow to verify and falsify more properties of the concrete model.

[19] 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 [20,21], 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. Namely, the thorough semantics evaluates a formula in an abstract model depending on its value in all possible (consistent) concretizations of the abstract model. This is in general more precise than the standard (inductive) semantics which might implicitly consider inconsistent underlying concrete models. However, the described problem results from the imprecision of the abstract model itself, meaning that undesired concrete models are included as part of its "real" concretizations. Thus, even the thorough semantics, which considers only the real concretizations, does not help to overcome the imprecision.

May hyper transitions resemble the de-focus operations of [9], just like must hyper transitions resemble the focus operations. However, the focus and de-focus operations of [9] are used in the evaluation of $\vee$ and $\wedge$-formulas. We use the standard semantics for $\vee$ and $\wedge$, which does not depend on the underlying model. Instead, we use may and must hyper transitions in the evaluation of $\Box$ and $\Diamond$-formulas, where the transitions of the underlying model need to be considered. In addition, in [9] the authors are interested in completeness and do not refer to the precision or model checking cost of the suggested class of models.

In terms of model checking in the presence of hyper transitions, [10] 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 [22]. 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.

Organization of the paper The μ-calculus and the 3-valued abstraction framework are introduced in Section 2. In Section 3, precision of 3-valued abstract models is defined and investigated. The class of *hyper kripke modal transition systems* (HTSs) is introduced and a precise construction of a HTS is provided. Section 4 refers to the model checking efficiency of HTSs. An efficient abstract model checking algorithm for the alternation-free μ-calculus, which avoids the exponential blowup inherent by the use of hyper transitions, is suggested. This algorithm is incorporated into a monotonic abstraction-refinement framework in Section 5. We conclude in Section 6.

In Appendix A, we consider concrete systems with multiple initial states, and show that similar imprecision and efficiency questions arise, and are settled by similar techniques.

2 Preliminaries

μ-calculus [4] 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:

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

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.

For a formula $\eta Z.\psi$, we denote by ψ^i the *unwinding* of the fixpoint formula i times. Formally,

$$\psi^0 = \begin{cases} false, \text{ if } \eta = \mu \\ true, \quad \text{if } \eta = \nu \end{cases}$$

and $\psi^{i+1} = \psi[Z := \psi^i]$.

Concrete Semantics Concrete systems are typically modelled as *Kripke structures*. A Kripke structure [2] 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 $\varphi \in \mathcal{L}_\mu$ w.r.t. a Kripke structure $M = (S, R, L)$ is an element of 2^S. The semantics is defined inductively. To handle subformulas which are not closed, an *environment* $\rho : \mathcal{V} \to 2^S$, which explains the meaning of free variables, is introduced. $[\![\varphi]\!]^{M,\rho}$ is defined inductively, for every μ-calculus formula.

For $g \in 2^S$, we denote with $\rho[Z \mapsto g]$ the environment that maps Z to g and agrees with ρ on all other arguments. In the following definition $f = \lambda g.[\![\varphi]\!]^{M,\rho[Z\mapsto g]}$ is an element of $2^S \to 2^S$ and $gfp(f)$, $lfp(f)$ stand for the greatest and least fixpoints of f. These fixpoints exist according to [23], since the elements in 2^S form a complete lattice under set inclusion ordering and the functional f is monotone w.r.t. this ordering.

$$
\begin{array}{lcl}
[\![true]\!]^{M,\rho} & := & S \\
[\![false]\!]^{M,\rho} & := & \emptyset \\
[\![l]\!]^{M,\rho} & := & \{s \mid l \in L(s)\} \\
[\![\Box\varphi]\!]^{M,\rho} & := & \{s \mid \forall s', \text{ if } sRs' \text{ then } s' \in [\![\varphi]\!]^{M,\rho}\} \\
[\![\Diamond\varphi]\!]^{M,\rho} & := & \{s \mid \exists s' \text{ s.t. } sRs' \text{ and } s' \in [\![\varphi]\!]^{M,\rho}\} \\
[\![\varphi_1 \wedge \varphi_2]\!]^{M,\rho} & := & [\![\varphi_1]\!]^{M,\rho} \cap [\![\varphi_2]\!]^{M,\rho} \\
[\![\varphi_1 \vee \varphi_2]\!]^{M,\rho} & := & [\![\varphi_1]\!]^{M,\rho} \cup [\![\varphi_2]\!]^{M,\rho} \\
[\![Z]\!]^{M,\rho} & := & \rho(Z) \\
[\![\mu Z.\varphi]\!]^{M,\rho} & := & \mathit{lfp}(\lambda g.[\![\varphi]\!]^{M,\rho[Z\mapsto g]}) \\
[\![\nu Z.\varphi]\!]^{M,\rho} & := & \mathit{gfp}(\lambda g.[\![\varphi]\!]^{M,\rho[Z\mapsto g]})
\end{array}
$$

Intuitively, in this context $\Box$ stands for "all successors", whereas $\Diamond$ stands for "exists a successor".

Note that for a closed formula φ, $[\![\varphi]\!]^{M,\rho} = [\![\varphi]\!]^{M,\rho'}$, for any environments ρ, ρ'. Thus, when closed formulas are considered, we drop the environment from the semantic brackets, and simply refer to $[\![\varphi]\!]^M$.

$[\![\varphi]\!]^M \subseteq S$ can also be viewed as a mapping $S \to \{\mathrm{tt}, \mathrm{ff}\}$. As such, for a closed formula $\varphi \in \mathcal{L}_\mu$, we sometimes write $[\![\varphi]\!]^M(s) = \mathrm{tt}$ instead of $s \in [\![\varphi]\!]^M$. Similarly we write $[\![\varphi]\!]^M(s) = \mathrm{ff}$ instead of $s \notin [\![\varphi]\!]^M$. $[\![\varphi]\!]^M(s) = \mathrm{tt}$ $(= \mathrm{ff})$ means that the formula φ is true (false) in the state s of the Kripke structure M.

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* [3] 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, $\perp$, 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. [24,25,11]). 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 $\Box^M, \Diamond^M : 2^S \to 2^S$. These definitions are given in terms of the components of M (e.g. abstract transitions and labeling), with the requirements that l^M and $(\neg l)^M$ are disjoint and the operators $\Box^M$ and $\Diamond^M$ are monotone w.r.t. set inclusion. The 3-valued semantics for the class is then defined based on Kleene's 3-valued logic for $\wedge$ and $\vee$, and with the standard definition for fixpoints. Only the definition for formulas of the form $l \in Lit$, $\Box\psi$, and $\Diamond\psi$ depends on the particular class of M.

Definition 2.1 (Generic 3-Valued Semantics) *Let M be a model from a 3-valued class. The* tt-set $[\![\varphi]\!]^{M,\rho}_{\mathrm{tt}} \subseteq S$ *of a μ-calculus formula φ over M and an environment $\rho : \mathcal{V} \to 2^S$ is defined inductively similarly to the concrete semantics, except that the definition for formulas of the form $l \in Lit$, $\Box\psi$, or $\Diamond\psi$ depends on the particular class of M. The* ff-set $[\![\varphi]\!]^{M,\rho}_{\mathrm{ff}} \subseteq S$ *over M and an environment ρ is defined dually. Specifically,*

$$
\begin{array}{lcl@{\qquad}lcl}
[\![true]\!]^{M,\rho}_{\mathrm{tt}} & := & S & [\![true]\!]^{M,\rho}_{\mathrm{ff}} & := & \emptyset \\
[\![false]\!]^{M,\rho}_{\mathrm{tt}} & := & \emptyset & [\![false]\!]^{M,\rho}_{\mathrm{ff}} & := & S \\
[\![l]\!]^{M,\rho}_{\mathrm{tt}} & := & l^M & [\![l]\!]^{M,\rho}_{\mathrm{ff}} & := & (\neg l)^M \\
[\![\Box\varphi]\!]^{M,\rho}_{\mathrm{tt}} & := & \Box^M([\![\varphi]\!]^{M,\rho}_{\mathrm{tt}}) & [\![\Box\varphi]\!]^{M,\rho}_{\mathrm{ff}} & := & \Diamond^M([\![\varphi]\!]^{M,\rho}_{\mathrm{ff}}) \\
[\![\Diamond\varphi]\!]^{M,\rho}_{\mathrm{tt}} & := & \Diamond^M([\![\varphi]\!]^{M,\rho}_{\mathrm{tt}}) & [\![\Diamond\varphi]\!]^{M,\rho}_{\mathrm{ff}} & := & \Box^M([\![\varphi]\!]^{M,\rho}_{\mathrm{ff}}) \\
[\![\varphi_1 \wedge \varphi_2]\!]^{M,\rho}_{\mathrm{tt}} & := & [\![\varphi_1]\!]^{M,\rho}_{\mathrm{tt}} \cap [\![\varphi_2]\!]^{M,\rho}_{\mathrm{tt}} & [\![\varphi_1 \wedge \varphi_2]\!]^{M,\rho}_{\mathrm{ff}} & := & [\![\varphi_1]\!]^{M,\rho}_{\mathrm{ff}} \cup [\![\varphi_2]\!]^{M,\rho}_{\mathrm{ff}} \\
[\![\varphi_1 \vee \varphi_2]\!]^{M,\rho}_{\mathrm{tt}} & := & [\![\varphi_1]\!]^{M,\rho}_{\mathrm{tt}} \cup [\![\varphi_2]\!]^{M,\rho}_{\mathrm{tt}} & [\![\varphi_1 \vee \varphi_2]\!]^{M,\rho}_{\mathrm{ff}} & := & [\![\varphi_1]\!]^{M,\rho}_{\mathrm{ff}} \cap [\![\varphi_2]\!]^{M,\rho}_{\mathrm{ff}} \\
[\![Z]\!]^{M,\rho}_{\mathrm{tt}} & := & \rho(Z) & [\![Z]\!]^{M,\rho}_{\mathrm{ff}} & := & \rho(Z) \\
[\![\mu Z.\varphi]\!]^{M,\rho}_{\mathrm{tt}} & := & \mathrm{lfp}(\lambda g.[\![\varphi]\!]^{M,\rho[Z\mapsto g]}_{\mathrm{tt}}) & [\![\mu Z.\varphi]\!]^{M,\rho}_{\mathrm{ff}} & := & \mathrm{gfp}(\lambda g.[\![\varphi]\!]^{M,\rho[Z\mapsto g]}_{\mathrm{ff}}) \\
[\![\nu Z.\varphi]\!]^{M,\rho}_{\mathrm{tt}} & := & \mathrm{gfp}(\lambda g.[\![\varphi]\!]^{M,\rho[Z\mapsto g]}_{\mathrm{tt}}) & [\![\nu Z.\varphi]\!]^{M,\rho}_{\mathrm{ff}} & := & \mathrm{lfp}(\lambda g.[\![\varphi]\!]^{M,\rho[Z\mapsto g]}_{\mathrm{ff}})
\end{array}
$$

If φ is a closed formula, then $[\![\varphi]\!]^{M,\rho}_{\mathrm{tt}} = [\![\varphi]\!]^{M,\rho'}_{\mathrm{tt}}$, and $[\![\varphi]\!]^{M,\rho}_{\mathrm{ff}} = [\![\varphi]\!]^{M,\rho'}_{\mathrm{ff}}$, for any environments ρ, ρ'. Thus, when closed formulas are considered, we drop the environment from the semantic brackets.

If for every $\varphi \in \mathcal{L}_\mu$, $[\![\varphi]\!]^M_{\mathrm{tt}} \cap [\![\varphi]\!]^M_{\mathrm{ff}} = \emptyset$, then M is consistent*. The 3-valued semantics of $\varphi \in \mathcal{L}_\mu$ over M, denoted $[\![\varphi]\!]^M_3$, is then defined to be a mapping $S \to \{\mathrm{tt}, \mathrm{ff}, \bot\}$:*

$$
[\![\varphi]\!]^M_3(s) = \begin{cases} \mathrm{tt}, \textit{ if } s \in [\![\varphi]\!]^M_{\mathrm{tt}} \\ \mathrm{ff}, \textit{ if } s \in [\![\varphi]\!]^M_{\mathrm{ff}} \\ \bot, \textit{ otherwise} \end{cases}
$$

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 *[11] 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 $l^M, \Box^M, \Diamond^M$ as follows. For every $l \in Lit$, $l^M = \{s \mid l \in L(s)\}$. For every $U \subseteq S$: $\Box^M(U) = \{s \mid \forall t \in S, \text{ if } sR^-t \text{ then } t \in U\}$, and $\Diamond^M(U) = \{s \mid \exists A \subseteq S \text{ s.t. } sR^+A \text{ and } A \subseteq U\}$. When integrated into Definition 2.1 this results in a 3-valued semantics. In particular, for a consistent GTS the definition for closed formulas of the form $l \in Lit$, $\Box\psi$ or $\Diamond\psi$ results in

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

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

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

Definition 2.4 (Generalized Mixed Simulation) *[11] 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 [11] 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* $\varphi \in \mathcal{L}_\mu$: $s_2 \in [\![\varphi]\!]_{\text{tt}}^{M_2} \Rightarrow s_1 \in [\![\varphi]\!]_{\text{tt}}^{M_1}$, *and* $s_2 \in [\![\varphi]\!]_{\text{ff}}^{M_2} \Rightarrow s_1 \in [\![\varphi]\!]_{\text{ff}}^{M_1}$.

Construction of an Abstract 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 [11].

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 $[\exists\exists]$ 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 $[\forall\exists\exists]$ rule:

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

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* [1].

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 (see Section 3.2).

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.

For example, the exact GTS for the (partial) Kripke structure from Example 1.1 includes two may transitions: (s_a, s_{1a}) and (s_a, s_{2a}), computed by the $\exists\exists$ rule, and four must hyper transitions: $(s_a, \{s_{1a}\})$, $(s_a, \{s_{2a}\})$, $(s_a, \{s_{1a}, s_{2a}\})$ and $(s_a, \{s_{1a}, s_{2a}, s_a\})$, computed by the $\forall\exists\exists$ rule. Moreover, s_{1a} is labeled p, while s_{2a} is labeled q. This construction ensures the existence of a generalized mixed simulation relation $H \subseteq S_C \times S_A$ such that $\{(s_c, s_a), (s'_c, s_{1a}), (s'_c, s_{2a})\} \subseteq H$. For example, for the pair $(s_c, s_a) \in H$, requirement 2 of the generalized mixed simulation requires that s_a has a may transition that corresponds to the transition $s_c R s'_c$. Either one of the may transitions $s_a R^- s_{1a}$ or $s_a R^- s_{2a}$ fulfills this requirement. Requirement 3 requires that each of the must hyper

[1] The term "exact" reflects the fact that the implications are exact. It should not be confused with the notion of precision.

transitions of s_a has a corresponding (hyper) transition in the concrete model. The (only) transition of s_c to s'_c satisfies this requirement, when we view it as a hyper transition whose target set is the singleton consisting of s'_c.

Remark 2.6 (Consistency) In the definition of the generic 3-valued semantics (Definition 2.1), the need to first separately define the tt-sets and the ff-sets of μ-calculus formulas arises since in the general case, if the 3-valued model at hand is inconsistent, the value of a formula in a state of the model can be both tt and ff, resulting in a 4-valued semantics.

In some cases, consistency is ensured by some "syntactic" condition which is added to the 3-valued class and prevents such a scenario. In such cases, the 3-valued semantics can immediately be defined as a mapping $S \to \{\mathrm{tt}, \mathrm{ff}, \bot\}$, without the need to first define the tt-sets and the ff-sets separately.

For example, when talking about GTSs, a requirement that the must (hyper) transitions are included in the may transitions is sometimes added. This means that if sR^+A, then for every $s' \in A$, sR^-s' holds too. Such a requirement ensures consistency. More generally, this requirement ensures that if $U_1, U_2 \subseteq S$ are disjoint sets, then $\Box^M(U_1) \cap \Diamond^M(U_2) = \emptyset$, which ensures (by induction) that for every $\varphi \in \mathcal{L}_\mu$, $[\![\varphi]\!]^M_{\mathrm{tt}} \cap [\![\varphi]\!]^M_{\mathrm{ff}} = \emptyset$.

One could think of requiring an equivalent requirement from any 3-valued class in order to ensure consistency. However, such a requirement also restricts the expressiveness of the models. For example, when extending the constructions of [7] to GTSs, the resulting abstract models do not maintain this requirement, even though they are consistent. These constructions only ensure that if U_1 and U_2 represent disjoint sets of *concrete states* then $\Box^M(U_1) \cap \Diamond^M(U_2) = \emptyset$. This is a sufficient condition for consistency. Yet, since it involves the underlying concrete states, it cannot be used in the more general context.

Thus, we do not add such restrictions. This allows the consideration of more expressive classes of models, at the price of complicating the semantics and allowing the value of a formula in a state to be both tt and ff. However, as explained above, when considering an abstract model, consistency is ensured, and a 3-valued semantics is obtained.

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}, $\Box^{M_A}(U_A)$, and $\Diamond^{M_A}(U_A)$ 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, $[\![U_A]\!]^{M_C} = \{s_c \mid s_c \in \gamma(U_A)\}$. In an abstract model M_A (from a 3-valued class), $[\![U_A]\!]^{M_A}_{\mathrm{tt}} = U_A$. As such, $\Box^{M_A}(U_A) = [\![\Box U_A]\!]^{M_A}_{\mathrm{tt}}$, and $\Diamond^{M_A}(U_A) = [\![\Diamond U_A]\!]^{M_A}_{\mathrm{tt}}$. In addition, recall that $l^{M_A} = [\![l]\!]^{M_A}_{\mathrm{tt}}$. This makes the tt-sets of formulas of the form l, $\Box U_A$, and $\Diamond U_A$ 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 [10] 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 *$\varphi = l$, $\Box U_A$ or $\Diamond U_A$, for $l \in Lit$ and $U_A \subseteq S_A$, if $\forall s_c \in \gamma(s_a) : [\![\varphi]\!]^{M_C}(s_c) = \mathrm{tt}$ (i.e., $s_c \in [\![\varphi]\!]^{M_C}$).*

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 $\Box U_A$ 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 *Let M_A be an abstract model (from some 3-valued class) over*

a set of abstract states S_A, and let $U_A \subseteq S_A$. We say that U_A is definable *by $\mathcal{L}_\mu$ in M_A if $U_A = [\![\varphi]\!]^{M_A}_{\mathrm{tt}}$ or $U_A = [\![\varphi]\!]^{M_A}_{\mathrm{ff}}$ for some $\varphi \in \mathcal{L}_\mu$.*

Definition 3.3 (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 $U_A \subseteq S_A$ which is definable by $\mathcal{L}_\mu$ in M_A: whenever s_a fulfills $\varphi = l$, $\Box U_A$ or $\Diamond U_A$, then $s_a \in [\![\varphi]\!]^{M_A}_{\mathrm{tt}}$.*

Thus whenever the information about l, $\Box U_A$, or $\Diamond U_A$ exists in the abstract states, a precise abstract model enables to see that. Note that we restrict the requirements of precision to sets U_A which are definable by some μ-calculus formula, since these are the sets that arise in the verification or falsification of $\mathcal{L}_\mu$ formulas. To formalize the generality of Definition 3.3, 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 $l, \Box U_A, \Diamond U_A$, it is also precise w.r.t. any other formula.

Definition 3.4 *Let $\mathcal{A} = (S_A, \gamma)$ be an abstraction. We define an* abstract semantics $[\![\varphi]\!]^{\mathcal{A}}_3$ *by using the generic 3-valued semantics (see Definition 2.1) with the following definitions of $l^{\mathcal{A}} \in 2^{S_A}$, and $\Box^{\mathcal{A}}, \Diamond^{\mathcal{A}} : 2^{S_A} \to 2^{S_A}$. For $l \in Lit$: $l^{\mathcal{A}} = \{s_a \mid s_a \text{ fulfills } l\}$. For $U_A \subseteq S_A$: $\Box^{\mathcal{A}}(U_A) = \{s_a \mid s_a \text{ fulfills } \Box U_A\}$, and $\Diamond^{\mathcal{A}}(U_A) = \{s_a \mid s_a \text{ fulfills } \Diamond U_A\}$. We say that $s_a \in S_A$* enables verification (falsification) *of $\varphi \in \mathcal{L}_\mu$ if $[\![\varphi]\!]^{\mathcal{A}}_3(s_a) = \mathrm{tt}$ (ff).*

Note that $l^{\mathcal{A}}$, $\Box^{\mathcal{A}}$ and $\Diamond^{\mathcal{A}}$ satisfy the requirements of Definition 2.1. Namely, $l^{\mathcal{A}}$ and $(\neg l)^{\mathcal{A}}$ are disjoint and the operators $\Box^{\mathcal{A}}$ and $\Diamond^{\mathcal{A}}$ are monotone w.r.t. set inclusion. The latter holds due to the monotonicity of the concrete $\Box$ and $\Diamond$ operators w.r.t. $\subseteq$ and since for $U_A \subseteq U'_A$, we have that $\gamma(U_A) \subseteq \gamma(U'_A)$. This ensures that if s_a fulfills $\Box U_A$ and $U_A \subseteq U'_A$, then s_a also fulfills $\Box U'_A$, and similarly for $\Diamond$.

Recall that by Definition 2.1, $[\![\varphi]\!]^{\mathcal{A}}_3(s_a) = \mathrm{tt}$ if $s_a \in [\![\varphi]\!]^{\mathcal{A}}_{\mathrm{tt}}$, and $[\![\varphi]\!]^{\mathcal{A}}_3(s_a) = \mathrm{ff}$ if $s_a \in [\![\varphi]\!]^{\mathcal{A}}_{\mathrm{ff}}$. The abstract semantics is well defined since whenever $s_a \in [\![\varphi]\!]^{\mathcal{A}}_{\mathrm{tt}}$ (resp. $[\![\varphi]\!]^{\mathcal{A}}_{\mathrm{ff}}$), then $\forall s_c \in \gamma(s_a) : [\![\varphi]\!]^{M_C}(s_c) = \mathrm{tt}$ (resp. ff). This ensures that $[\![\varphi]\!]^{\mathcal{A}}_{\mathrm{tt}} \cap [\![\varphi]\!]^{\mathcal{A}}_{\mathrm{ff}} = \emptyset$.

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

Theorem 3.5 *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 $[\![\varphi]\!]^{M_A}_3(s_a) = \mathrm{tt}$ (ff).*

Note that $[\![\varphi]\!]^{M_A}_3$ is well-defined since M_A is an abstract model for M_C, thus it is consistent.

Proof: We prove that if s_a enables verification of φ, i.e. $[\![\varphi]\!]^{\mathcal{A}}_{\mathrm{tt}}(s_a) = \mathrm{tt}$ then $[\![\varphi]\!]^{M_A}_3(s_a) = \mathrm{tt}$. The proof for falsification is implied since the 3-valued semantics ensures that $[\![\varphi]\!]^{M_A}_3(s_a) = \mathrm{ff}$ iff $[\![\neg\varphi]\!]^{M_A}_3(s_a) = \mathrm{tt}$, and similarly for the abstract semantics, where $\neg\varphi$ stands for the formula resulting by pushing the negation to the literals, while exchanging *true* with $false$, $\wedge$ with $\vee$, $\Box$ with $\Diamond$, and μ with ν.

More specifically, we prove that if $s_a \in [\![\varphi]\!]^{\mathcal{A}}_{\mathrm{tt}}$, then $s_a \in [\![\varphi]\!]^{M_A}_{\mathrm{tt}}$. We refer to (closed) fixpoint-free formulas. This is justified by the property that the abstract set of states S_A is finite: A variation of the Knaster-Tarski theorem [23] implies that when the set of states is finite, then for a formula $\eta Z.\psi$ and an environment ρ, there exists $j \in \mathbb{N}$ such that for every $i \geq j$: $[\![\eta Z.\psi]\!]^{M_A,\rho}_{\mathrm{tt}} = [\![\psi^i]\!]^{M_A,\rho}_{\mathrm{tt}}$, where ψ^i denotes the unwinding of the fixpoint formula i times. Similarly, for the abstract semantics, it holds that there exists $j' \in \mathbb{N}$ such that for every $i \geq j'$: $[\![\eta Z.\psi]\!]^{\mathcal{A},\rho}_{\mathrm{tt}} = [\![\psi^i]\!]^{\mathcal{A},\rho}_{\mathrm{tt}}$. j and j' might be different, but both are bounded by $|S_A|$. Applying this argument recursively with a sufficiently large number of unwindings (e.g. $i = |S_A|$) for each fixpoint subformula implies that any formula $\varphi \in \mathcal{L}_\mu$ is equivalent to a (closed) fixpoint-free formula φ' w.r.t. both M_A and the abstract semantics, in the sense that $[\![\varphi]\!]^{M_A}_{\mathrm{tt}} = [\![\varphi']\!]^{M_A}_{\mathrm{tt}}$, and in addition $[\![\varphi]\!]^{\mathcal{A}}_{\mathrm{tt}} = [\![\varphi']\!]^{\mathcal{A}}_{\mathrm{tt}}$. Therefore, it suffices to refer to fixpoint-free formulas in the proof.

The proof is by induction on the structure of fixpoint-free μ-calculus formulas. The interesting cases are when $\varphi = l \in Lit$, $\Box\psi$, or $\Diamond\psi$. The remaining cases are immediate as both $[\![\varphi]\!]^{\mathcal{A},\rho}_{\mathrm{tt}}$ and $[\![\varphi]\!]^{M_A,\rho}_{\mathrm{tt}}$ are defined according to the generic 3-valued semantics.

- If $\varphi = l \in Lit$ and $s_a \in [\![l]\!]^{\mathcal{A}}_{\mathrm{tt}}$, then since $[\![l]\!]^{\mathcal{A}}_{\mathrm{tt}} = l^{\mathcal{A}}$ and by the definition of $l^{\mathcal{A}}$ we conclude that s_a fulfills l. Thus by the definition of a precise model $s_a \in [\![l]\!]^{M_A}_{\mathrm{tt}}$.
- Suppose $s_a \in [\![\Box\psi]\!]^{\mathcal{A}}_{\mathrm{tt}}$. This means that $s_a \in \Box^{\mathcal{A}}([\![\psi]\!]^{\mathcal{A}}_{\mathrm{tt}})$. Let $U_A = [\![\psi]\!]^{\mathcal{A}}_{\mathrm{tt}}$ and $U'_A = [\![\psi]\!]^{M_A}_{\mathrm{tt}}$. By the induction hypothesis for ψ, for every such $s'_a \in U_A = [\![\psi]\!]^{\mathcal{A}}_{\mathrm{tt}}$, we have that $s'_a \in [\![\psi]\!]^{M_A}_{\mathrm{tt}} = U'_A$. Thus $U_A \subseteq U'_A$. Recall that $s_a \in \Box^{\mathcal{A}}([\![\psi]\!]^{\mathcal{A}}_{\mathrm{tt}}) = \Box^{\mathcal{A}}(U_A)$. By the monotonicity of $\Box^{\mathcal{A}}$ w.r.t. set inclusion, we conclude that $s_a \in \Box^{\mathcal{A}}(U'_A)$. This means that s_a fulfills $\Box U'_A$. Moreover, by its definition, U'_A is definable by $\mathcal{L}_\mu$ in M_A (since $U'_A = [\![\psi]\!]^{M_A}_{\mathrm{tt}}$). Thus, by the definition of a precise model $s_a \in [\![\Box U'_A]\!]^{M_A}_{\mathrm{tt}}$, meaning that $s_a \in \Box^{M_A}([\![U'_A]\!]^{M_A}_{\mathrm{tt}}) = \Box^{M_A}(U'_A) = \Box^{M_A}([\![\psi]\!]^{M_A}_{\mathrm{tt}})$. Thus, by the 3-valued semantics $s_a \in [\![\Box\psi]\!]^{M_A}_{\mathrm{tt}}$ as well. The case of $\varphi = \Diamond\psi$ is similar.

□

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.

Definition 3.6 *A 3-valued class of models is* structural *if its definitions of* $\Box^M, \Diamond^M : 2^{S_A} \to 2^{S_A}$ *ensure that for every* $U_A \subseteq S_A$*, whenever* $s_a \in \Box^M(U_A)$*, 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* $\gamma(U_A)$*.*

Note that for every $U_A \subseteq S_A$ which is equal to $[\![\varphi]\!]^M_{\mathrm{tt}}$ for some $\varphi \in \mathcal{L}_\mu$, the conditions of Definition 3.6 are guaranteed to hold, since in this case $\Box^M(U_A) = [\![\Box\varphi]\!]^M_{\mathrm{tt}}$, and similarly $\Diamond^M(U_A) = [\![\Diamond\varphi]\!]^M_{\mathrm{tt}}$. Thus the conditions are implied by the preservation guarantee of the class. However, for a class to be structural, we require that these conditions hold for *every* $U_A \subseteq S_A$. 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 3.7 *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* $\varphi \in \mathcal{L}_\mu$*:* $[\![\varphi]\!]_3^{M'_A}(s_a) \neq \bot \Rightarrow [\![\varphi]\!]_3^{M_A}(s_a) = [\![\varphi]\!]_3^{M'_A}(s_a)$*.*

Proof: Let M'_A be *some* abstract model as described in the theorem, and M_A a precise model w.r.t. (S_A, γ). We prove that for every $s_a \in S_A$, if $[\![\varphi]\!]_3^{M'_A}(s_a) = \mathrm{tt}$, then $[\![\varphi]\!]_3^{M_A}(s_a) = \mathrm{tt}$. The proof for falsification is implied since the 3-valued semantics ensures that $[\![\varphi]\!]_3^{M'_A}(s_a) = \mathrm{ff}$ iff $[\![\neg\varphi]\!]_3^{M'_A}(s_a) = \mathrm{tt}$ and similarly for M_A, where $\neg\varphi$ stands for the formula resulting by pushing the negation to the literals as in the proof of Theorem 3.5. More specifically, we prove that if $s_a \in [\![\varphi]\!]^{M'_A}_{\mathrm{tt}}$, then $s_a \in [\![\varphi]\!]^{M_A}_{\mathrm{tt}}$. As in the proof of Theorem 3.5, we refer to (closed) fixpoint-free formulas. This is justified by the property that the abstract set of states S_A of both models is finite (see the proof of Theorem 3.5 for further details). The proof is by induction on the structure of fixpoint-free μ-calculus formulas. As before, we present the interesting cases where $\varphi = l \in Lit$, $\Box\psi$, or $\Diamond\psi$. The remaining cases are immediate as both $[\![\varphi]\!]^{M'_A,\rho}_{\mathrm{tt}}$ and $[\![\varphi]\!]^{M_A,\rho}_{\mathrm{tt}}$ are defined with the generic 3-valued semantics.

- For $\varphi = l \in Lit$, if $s_a \in [\![l]\!]^{M'_A}_{\mathrm{tt}}$, then by the preservation guarantee of M'_A, we conclude that $\forall s_c \in \gamma(s_a)$, $s_c \in [\![l]\!]^{M_C}$ (i.e. $[\![l]\!]^{M_C}(s_c) = \mathrm{tt}$), thus s_a fulfills l. Since M_A is precise w.r.t. (S_A, γ), we conclude that $s_a \in [\![l]\!]^{M_A}_{\mathrm{tt}}$.
- Suppose $\varphi = \Box\psi$, and $s_a \in [\![\Box\psi]\!]^{M'_A}_{\mathrm{tt}}$. Let $U_A = [\![\psi]\!]^{M_A}_{\mathrm{tt}}$. To show that $s_a \in [\![\Box\psi]\!]^{M_A}_{\mathrm{tt}}$, we need to show that $s_a \in \Box^{M_A}(U_A)$. Since $s_a \in [\![\Box\psi]\!]^{M'_A}_{\mathrm{tt}}$, then by the 3-valued semantics, $s_a \in \Box^{M'_A}([\![\psi]\!]^{M'_A}_{\mathrm{tt}})$. By the induction hypothesis, $[\![\psi]\!]^{M'_A}_{\mathrm{tt}} \subseteq [\![\psi]\!]^{M_A}_{\mathrm{tt}} = U_A$. Thus, by monotonicity of $\Box^{M'_A}$ w.r.t. $\subseteq$, we conclude

that $\Box^{M'_A}([\![\psi]\!]^{M'_A}_{\mathrm{tt}}) \subseteq \Box^{M'_A}(U_A)$, thus $s_a \in \Box^{M'_A}(U_A)$. Since M'_A belongs to a structural class, this ensures us that for every $s_c \in \gamma(s_a)$ all the concrete successors of s_c are in $\gamma(U_A)$, and thus belong to $[\![U_A]\!]^{M_C}$. Thus $\forall s_c \in \gamma(s_a) : s_c \in [\![\Box U_A]\!]^{M_C}$ (i.e., $[\![\Box U_A]\!]^{M_C}(s_c) = \mathrm{tt}$). Thus by definition s_a fulfills $\Box U_A$. Since M_A is precise w.r.t. (S_A, γ) and $U_A = [\![\psi]\!]^{M_A}_{\mathrm{tt}}$ is definable by $\mathcal{L}_\mu$ in M_A, this ensures that $s_a \in [\![\Box U_A]\!]^{M_A}_{\mathrm{tt}}$. Thus by the 3-valued semantics $s_a \in \Box^{M_A}(U_A) = \Box^{M_A}([\![\psi]\!]^{M_A}_{\mathrm{tt}})$, and $s_a \in [\![\Box\psi]\!]^{M_A}_{\mathrm{tt}}$. The proof for $\varphi = \Diamond\psi$ is similar.

□

3.2 Precision of GTSs

Equipped with formal definitions of precision, we go back to our questions about the precision of GTSs. We first observe that if the abstraction partitions the concrete states, then the answer to both questions is "yes":

Theorem 3.8 *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).*

Proof: Let M_A denote the exact GTS from section 2.

- Suppose that $s_a \in S_A$ fulfills $l \in Lit$. This means that $\forall s_c \in \gamma(s_a) : [\![l]\!]^{M_C}(s_c) = \mathrm{tt}$ (i.e. $s_c \in [\![l]\!]^{M_C}$), and by the concrete semantics this implies that $\forall s_c \in \gamma(s_a) : l \in L_C(s_c)$. Therefore by the construction of the exact GTS, $l \in L_A(s_a)$ and hence $s_a \in \{s \mid l \in L_A(s)\} = l^{M_A} = [\![l]\!]^{M_A}_{\mathrm{tt}}$.
- Suppose that $s_a \in S_A$ fulfills $\varphi = \Box U_A$ (for some U_A which is definable by $\mathcal{L}_\mu$ in M_A). Thus, $\forall s_c \in \gamma(s_a) : [\![\Box U_A]\!]^{M_C}(s_c) = \mathrm{tt}$ (i.e. $s_c \in [\![\Box U_A]\!]^{M_C}$). This means that $\forall s_c \in \gamma(s_a)\ \forall s'_c$, if $s_c R s'_c$ then $s'_c \in [\![U_A]\!]^{M_C}$. In other words, $\forall s_c \in \gamma(s_a)\ \forall s'_c$, if $s_c R s'_c$ then $s'_c \in \gamma(U_A)$ (1). Now, consider an outgoing may transition of s_a to some s'_a in M_A. It was computed based on the $\exists\exists$ condition, meaning that $\exists s_c \in \gamma(s_a)\ \exists s'_c \in \gamma(s'_a)$ s.t. $s_c R s'_c$. By (1), this also ensures that $s'_c \in \gamma(U_A)$. Thus there exists $s''_a \in U_A$ such that $s'_c \in \gamma(s''_a)$. Since we have a partition, it implies that $s'_a = s''_a$ (since also $s'_c \in \gamma(s'_a)$). Thus $s'_a \in U_A$ and as such $s'_a \in [\![U_A]\!]^{M_A}_{\mathrm{tt}}$. As this is true for every outgoing may transition of s_a, we conclude that $s_a \in \Box^{M_A}([\![U_A]\!]^{M_A}_{\mathrm{tt}}) = [\![\Box U_A]\!]^{M_A}_{\mathrm{tt}}$.
- Suppose that $s_a \in S_A$ fulfills $\varphi = \Diamond U_A$ (for some U_A which is definable by $\mathcal{L}_\mu$ in M_A). Thus, $\forall s_c \in \gamma(s_a) : [\![\Diamond U_A]\!]^{M_C}(s_c) = \mathrm{tt}$ (i.e., $s_c \in [\![\Diamond U_A]\!]^{M_C}$). This means that $\forall s_c \in \gamma(s_a)\ \exists s'_c$ such that $s_c R s'_c$ and $s'_c \in [\![U_A]\!]^{M_C}$, i.e. $s'_c \in \gamma(U_A)$. In other words, $\forall s_c \in \gamma(s_a)\ \exists s'_c \in \gamma(U_A)$ such that $s_c R s'_c$. Thus, by the construction there exists a must hyper transition in M_A from s_a to U_A, where all the states belong to $[\![U_A]\!]^{M_A}_{\mathrm{tt}}$ (by definition). Thus $s_a \in$

$$\Diamond^{M_A}(U_A) = \Diamond^{M_A}([\![U_A]\!]^{M_A}_{\mathrm{tt}}) = [\![\Diamond U_A]\!]^{M_A}_{\mathrm{tt}}.$$

□

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. In this case, the exact GTS is not necessarily precise (e.g. Example 1.1). Still, if the abstraction satisfies the *existence of a best approximation assumption* [13], then a precise abstract model in the form of a GTS can be constructed by an optimized version of the exact HTS. This is the case, for example, when the abstraction (S_A, γ) is a part of a Galois connection.

In our terminology, an abstraction (S_A, γ) satisfies the *existence of a best approximation assumption* if for every $s_c \in S_C$ there exists $s_a \in S_A$ such that $s_c \in \gamma(s_a)$ and for every $s'_a \in S_A$, if $s_c \in \gamma(s_a)$ then $\gamma(s_a) \subseteq \gamma(s'_a)$. s_a is called the best approximation of s_c. For such an abstraction, the construction of the may transitions of the exact GTS can be optimized, following [7], resulting in the *optimal GTS*. The may transitions of the optimal GTS are computed by the following optimized [$\exists\exists$] rule:

$$s_a R^- s'_a \iff \exists s_c \in \gamma(s_a)\ \exists s'_c \in \gamma(s'_a) \text{ s.t. } s_c R s'_c \text{ and } s'_a \text{ is the best approximation of } s'_c$$

Namely, may transitions whose target state is not the best approximation of the target of any corresponding concrete transition are removed. Note that the optimized rule is only well defined if the existence of a best approximation assumption holds. This optimization maintains the property that if $s_c \in \gamma(s_a)$, then $(M_C, s_c) \preceq (M_A, s_a)$. As before, the generalized mixed simulation $H \subseteq S_C \times S_A$ is induced by γ as follows: $(s_c, s_a) \in H$ iff $s_c \in \gamma(s_a)$.

Theorem 3.9 *If the abstraction (S_A, γ) satisfies the existence of a best approximation assumption, then the optimal GTS defined above is precise w.r.t. (S_A, γ).*

Proof: Let M_A denote the optimal GTS defined above. We first note that the optimal GTS has the property that if $\gamma(s_a) \subseteq \gamma(s'_a)$ then whenever $s'_a \in [\![\psi]\!]^{M_A}_{\mathrm{tt}}$ (resp. $s'_a \in [\![\psi]\!]^{M_A}_{\mathrm{ff}}$) for some $\psi \in \mathcal{L}_\mu$, then $s_a \in [\![\psi]\!]^{M_A}_{\mathrm{tt}}$ (resp. $s_a \in [\![\psi]\!]^{M_A}_{\mathrm{ff}}$) as well. This follows by induction on the structure of μ-calculus formulas, based on the construction of the optimal GTS, and on the definitions of l^{M_A}, $\Box^{M_A}$ and $\Diamond^{M_A}$ in a GTS. The main point in the proof is that for such s_a and s'_a in the optimal GTS, (1) if s'_a is labeled $l \in Lit$, so is s_a, meaning that if $s'_a \in l^{M_A}$, then $s_a \in l^{M_A}$ as well, (2) the set of outgoing may transitions of s'_a is a superset of the set of outgoing may transitions of s_a, meaning that if $s'_a \in \Box^{M_A}(U_A)$, then $s_a \in \Box^{M_A}(U_A)$ as well, and (3) the set of outgoing must hyper transitions

of s'_a is a subset of the set of outgoing must hyper transitions of s_a, meaning that if $s'_a \in \Diamond^{M_A}(U_A)$, then $s_a \in \Diamond^{M_A}(U_A)$ as well.

We now return to the proof of the theorem. The cases where $s_a \in S_A$ fulfills $l \in Lit$ or $\Diamond U_A$ are exactly as in the proof of theorem 3.8 (note that the proof of these cases did not rely on the fact that we had a partition of the concrete states). We refer to the remaining case, which is different.

- Suppose that $s_a \in S_A$ fulfills $\varphi = \Box U_A$ for some U_A which is definable by $\mathcal{L}_\mu$ in M_A. Thus, $\forall s_c \in \gamma(s_a) : [\![\Box U_A]\!]^{M_C}(s_c) = \mathrm{tt}$ (i.e. $s_c \in [\![\Box U_A]\!]^{M_C}$). This means that $\forall s_c \in \gamma(s_a)\ \forall s'_c$, if $s_c R s'_c$ then $s'_c \in [\![U_A]\!]^{M_C}$. In other words, $\forall s_c \in \gamma(s_a)\ \forall s'_c$, if $s_c R s'_c$ then $s'_c \in \gamma(U_A)$ (1). Now, consider an outgoing may transition of s_a to some s'_a in M_A. It was computed based on the optimized $\exists\exists$ condition, meaning that $\exists s_c \in \gamma(s_a)\ \exists s'_c \in \gamma(s'_a)$ s.t. $s_c R s'_c$ and s'_a is the best approximation of s'_c. By (1), this also ensures that $s'_c \in \gamma(U_A)$. Thus there exists $s''_a \in U_A$ such that $s'_c \in \gamma(s''_a)$. Recall that s'_a is the best approximation of s'_c, which ensures that $\gamma(s'_a) \subseteq \gamma(s''_a)$. Moreover, recall that U_A is definable by $\mathcal{L}_\mu$ in M_A, i.e., $U_A = [\![\psi]\!]^{M_A}_{\mathrm{tt}}$ or $U_A = [\![\psi]\!]^{M_A}_{\mathrm{ff}}$ for some $\psi \in \mathcal{L}_\mu$. Thus, by the previous property of the optimal GTS, we have that $s'_a \in U_A$ as well. As such, $s'_a \in [\![U_A]\!]^{M_A}_{\mathrm{tt}}$. As this is true for every outgoing may transition of s_a, we conclude that $s_a \in \Box^{M_A}([\![U_A]\!]^{M_A}_{\mathrm{tt}}) = [\![\Box U_A]\!]^{M_A}_{\mathrm{tt}}$.

□

In the next subsection, however, we show that in the most general setting, when no restriction is imposed on the abstraction, the answer to both questions is "no".

3.3 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, and do not satisfy the existence of a best approximation assumption (in this example, the state s'_c does not have a best approximation since it is abstracted by both s_{1a} and s_{2a} which are incomparable), 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 3.4). Thus, none of the possible GTSs is precise w.r.t. the given abstraction.

Theorem 3.10 *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 3.10 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. Basically, in order to obtain a generalized mixed simulation relation between s_c and s_a, the abstract model has to over approximate each concrete transition $s_c R s'_c$ by at least one may transition leaving s_a. When the abstraction forms a partition of the concrete states, there is exactly one possibility to over approximate each such transition since s'_c is abstracted by exactly one abstract state s'_a. Therefore, the exact GTS is precise in this case (see Theorem 3.8). When the abstraction does not form a partition, we might have several candidates to over approximate each concrete transition. Still, if the abstraction satisfies the existence of a best approximation assumption, then for each concrete transition there is a best overapproximation (may transition), namely, the one where the target state is the best approximation of s'_c. This results in the optimal GTS, which is precise in this case (see Theorem 3.9). However, when the existence of a best approximation assumption is eliminated as well, there might not be a best choice of may transitions (see also [13]), 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.

3.4 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 3.11 *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 $\Box^M$. The definitions of l^M and $\Diamond^M$ are the same as for GTSs. 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:

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

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 of s are to states that satisfy ψ, we now require that there *exists* a may hyper transition of s such that *all* the states within the target set satisfy ψ. This is justified by the fact that in an abstract HTS, *each* may hyper transition of s (as opposed to *all* the may transitions of s together in an abstract GTS) will over approximate *all* the concrete transitions leaving the concrete states represented by s.

Note that an HTS might be inconsistent. For example, a state s of an HTS M might have both a may hyper-transition to $[\![l]\!]_{\text{tt}}^M = \{s' \mid l \in L(s')\}$ and a must hyper-transition to $[\![l]\!]_{\text{ff}}^M = \{s' \mid \neg l \in L(s')\}$. This means that $s \in [\![\Box l]\!]_{\text{tt}}^M \cap [\![\Box l]\!]_{\text{ff}}^M$. Yet, we are interested in abstract HTSs, which are always consistent.

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. This encoding preserves the logical semantics of the models. Preservation of $\mathcal{L}_\mu$ between HTSs (and in particular between an HTS and a Kripke structure) is then guaranteed by the following relation.

Definition 3.12 (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* $(M_1, s_1) \preceq (M_2, s_2)$.

Instead of requiring that for each may transition of M_1, there exists a corresponding may transition in M_2 such that the target states satisfy $(s'_1, s'_2) \in H$, i.e. s'_2 over approximates s'_1, we now require that for each may hyper transition of M_2 there exists a corresponding may hyper transition in M_1 (note that the indices are swapped), such that the target sets satisfy $(A_1, A_2) \in H$, i.e. A_2 over approximates A_1.

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 3.13 *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* $\varphi \in \mathcal{L}_\mu$*:* $s_2 \in [\![\varphi]\!]^{M_2}_{\mathrm{tt}} \Rightarrow s_1 \in [\![\varphi]\!]^{M_1}_{\mathrm{tt}}$*, and* $s_2 \in [\![\varphi]\!]^{M_2}_{\mathrm{ff}} \Rightarrow s_1 \in [\![\varphi]\!]^{M_1}_{\mathrm{ff}}$.

Proof: The proof is obtained by induction on the structure of μ-calculus formulas, similarly to the proof of Theorem 2.5. The only changes occur in cases where the semantics was changed, i.e. where may hyper transitions are used instead of (ordinary) may transitions.

- Suppose $s_2 \in [\![\Box\psi]\!]^{M_2,\rho}_{\mathrm{tt}}$. Then by the definition of the semantics there exists a may hyper transition from s_2 to A_2 such that for each $s'_2 \in A_2$: $s'_2 \in [\![\psi]\!]^{M_2,\rho}_{\mathrm{tt}}$. Moreover, since $(s_1, s_2) \in H$, we know that there exists A_1 such that s_1 has a may hyper transition to A_1 and $(A_1, A_2) \in H^{\forall\exists}$, meaning that $\forall s'_1 \in A_1\ \exists s'_2 \in A_2 : (s'_1, s'_2) \in H$. Let s'_1 be such a state in A_1 and s'_2 the corresponding state from A_2. Since $s'_2 \in A_2$, we know that $s'_2 \in [\![\psi]\!]^{M_2,\rho}_{\mathrm{tt}}$. By the induction hypothesis, this implies that $s'_1 \in [\![\psi]\!]^{M_1,\rho}_{\mathrm{tt}}$. That is, $\forall s'_1 \in A_1$: $s'_1 \in [\![\psi]\!]^{M_1,\rho}_{\mathrm{tt}}$. Thus $s_1 \in [\![\Box\psi]\!]^{M_1,\rho}_{\mathrm{tt}}$. The treatment of the case where $s_2 \in [\![\Diamond\psi]\!]^{M_2,\rho}_{\mathrm{ff}}$ is dual.

□

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 a GTS (see Section 2) 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 [$\forall\forall\exists$] 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. In other words, each of the target sets of the outgoing may hyper transitions of s_a over approximates *all* the targets of the concrete transitions leaving the concrete states represented by s_a. An example of a "legal" may hyper transition that satisfies the $\forall\forall\exists$ condition is (s_a, A_a) for every $s_a \in S_A$ and $A_a = \{s'_a \mid \exists s_c \in \gamma(s_a)\ \exists s'_c \in \gamma(s'_a) \text{ s.t. } s_c R s'_c\}$. 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 3.14 *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 $(M_C, s_c) \preceq (M_A, s_a)$.*

Proof: We show that $H \subseteq S_C \times S_A$ defined by $(s_c, s_a) \in H$ iff $s_c \in \gamma(s_a)$ is a hyper mixed simulation. Let $s_c \in \gamma(s_a)$. Requirements 1 and 3 regarding the labeling and the must hyper transitions are fulfilled as in a GTS. We now refer to requirement 2. When viewing a Kripke structure as a HTS, every state $s_c \in S_C$ has exactly one outgoing may hyper transition $s_c R^- A_c$ where A_c consists of all the destination states of the ordinary transitions of s_c, i.e., $A_c = \{s'_c : s_c R s'_c\}$. Now, let $A_a \subseteq S_A$ be such that $s_a R^- A_a$. Since $s_c R^- A_c$ is the only may hyper transition of s_c in M_C, we need to show that $(A_c, A_a) \in H^{\forall\exists}$. Since $s_a R^- A_a$, this means (by the construction) that $\forall s_c \in \gamma(s_a)\ \forall s'_c\ [\ s_c R s'_c \Rightarrow \exists s'_a \in A_a \text{ s.t. } s'_c \in \gamma(s'_a)\ ]$. In particular, for our s_c, we have that $\forall s'_c\ [\ s_c R s'_c \Rightarrow \exists s'_a \in A_a \text{ s.t. } s'_c \in \gamma(s'_a)\ ]$, and in particular, $\forall s'_c \in A_c\ \exists s'_a \in A_a \text{ s.t. } s'_c \in \gamma(s'_a)$. This is because by the definition of A_c, every $s'_c \in A_c$ is a successor of s_c, i.e. $s_c R s'_c$ holds for it. $s'_c \in \gamma(s'_a)$ implies that $(s'_c, s'_a) \in H$. Thus, $(A_c, A_a) \in H^{\forall\exists}$. □

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. Note that both of these hyper transitions satisfy the $\forall\forall\exists$ condition, which ensures that each of them over approximates *all* the concrete transitions of the concrete state represented by s_a (in this case there is only one such concrete transition). In addition, the labeling function defines $L_A(s_{1a}) = \{p\}$, and $L_A(s_{2a}) = \{q\}$. Now, the may hyper transition $(s_a, \{s_{1a}\})$ enables to verify $\Box p$. Similarly, the may hyper transition $(s_a, \{s_{2a}\})$ enables to verify $\Box q$. Thus, $\Box p \wedge \Box q$ is verified.

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 3.15 *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).*

Proof: The cases where $s_a \in S_A$ fulfills $l \in Lit$ or $\Diamond U_A$ are exactly as in the proof of theorem 3.8 (note that the proof of these cases did not rely on the fact that we had a partition of the concrete states). We refer to the remaining case, which is different.

- Suppose that $s_a \in S_A$ fulfills $\varphi = \Box U_A$ (for some U_A which is definable by $\mathcal{L}_\mu$ in M_A). This means that $\forall s_c \in \gamma(s_a) : [\![\Box U_A]\!]^{M_C}(s_c) = \mathrm{tt}$ (i.e. $s_c \in [\![\Box U_A]\!]^{M_C}$). In other words, $\forall s_c \in \gamma(s_a)\ \forall s'_c,$ if $s_c R s'_c$ then $s'_c \in [\![U_A]\!]^{M_C}$, or equivalently $\forall s_c \in \gamma(s_a)\ \forall s'_c,$ if $s_c R s'_c$ then $s'_c \in \gamma(U_A)$, meaning that, $\forall s_c \in \gamma(s_a)\ \forall s'_c\ [\ s_c R s'_c \Rightarrow \exists s'_a \in U_a$ s.t. $s'_c \in \gamma(s'_a)\ ]$. Thus, by the construction of the exact HTS there exists a may hyper transition in M_A from s_a to U_A. In addition, all the (abstract) states in U_A belong to $[\![U_A]\!]^{M_A}_{\mathrm{tt}}$ (by definition), thus by the 3-valued semantics over HTS $s_a \in \Box^{M_A}(U_A) = \Box^{M_A}([\![U_A]\!]^{M_A}_{\mathrm{tt}}) = [\![\Box U_A]\!]^{M_A}_{\mathrm{tt}}$.

□

Optimization As suggested in [11,10] for must hyper transitions, a HTS can be reduced without damaging its precision by discarding may and must hyper-transitions (s_a, A_a) that are not *minimal*, meaning that there is another hyper transition (s_a, A'_a) of the same type where $A'_a \subset A_a$. In particular, Theorem 3.15 still holds after this optimization is applied.

For example, in the exact HTS constructed for Example 1.1, R^- (and also R^+) includes in addition to the hyper transitions $(s_a, \{s_{1a}\})$ and $(s_a, \{s_{2a}\})$, the hyper transition $(s_a, \{s_{1a}, s_{2a}\})$, which is not minimal. The latter may hyper transition indicates that the set $\{s_{1a}, s_{2a}\}$ over approximates all the targets of the concrete transitions leaving the concrete state represented by s_a (in this example the only concrete target is the state s'_c). However, the may hyper transitions $(s_a, \{s_{1a}\})$ and $(s_a, \{s_{2a}\})$ represent tighter overapproximations of s'_c. Thus the same precision is achieved when the may hyper transition $(s_a, \{s_{1a}, s_{2a}\})$ is omitted. In particular, the overapproximation provided by the may hyper transition $(s_a, \{s_{1a}, s_{2a}\})$ does not help in the verification of $\Box p$, nor does it help in the verification of $\Box q$. This is because for a may hyper transition to witness that $\Box\psi$ holds in an abstract state, *all* of the target states of the hyper transition must satisfy ψ, which is not the case for

s_{1a} and s_{2a} w.r.t. neither p nor q. The same reasoning applies to other non-minimal hyper transitions which are included in R^- and R^+, leaving us with $R^+ = R^- = \{(s_a, \{s_{1a}\}), (s_a, \{s_{2a}\})\}$.

Note that in a HTS, both the 3-valued semantics and the preservation relation (hyper mixed simulation) treat may and must hyper-transitions in the same way, rather than dually. Still, may hyper-transitions and must hyper-transitions have different roles in an abstract model: the first provides an overapproximation of the concrete transitions, and the latter provides an underapproximation for them. This difference is captured by the fact that when viewing a Kripke structure as a HTS, the may and must hyper-transitions are defined differently. Namely, each concrete transition is considered a must hyper-transition, whereas all the concrete transitions together form a single may hyper-transition. As such, the may and must hyper-transitions of an abstract HTS, which is related to the concrete Kripke structure by a hyper mixed simulation, are each required to satisfy different rules w.r.t. the concrete transitions. This is demonstrated by the construction of an abstract HTS, where the may and must hyper-transitions are defined differently.

3.5 Discussion: Precision versus Completeness

Our definition of precision should not be confused with the notion of completeness in abstract interpretation.

Completeness in abstract interpretation [26,27] means that no additional loss of information is accumulated when computing the semantics in the abstract model over the abstract states. The standard notion of completeness requires that $[\![\psi]\!]^{M_A}_{\mathrm{tt}} = \alpha([\![\psi]\!]^{M_C})$, where α denotes an abstraction function (or relation) that maps each set of concrete states to an abstract state that represents it. This means that the result of computing the abstract semantics over the abstract states coincides with the result of computing the concrete semantics over the concrete states and then applying abstraction on the result [2].

It is shown in [29] that completeness in abstract interpretation is equivalent to a variant of *strong preservation*, called *best preservation*. Strong preservation in model checking means that the same formulas can be verified on the concrete model and on the abstract model. The notion of strong preservation is generalized in [30,31,29] to abstract interpretation-based models and related

[2] [28] also defines a second form of completeness, called *forward completeness*, which requires that $\gamma([\![\psi]\!]^{M_A}_{\mathrm{tt}}) = [\![\psi]\!]^{M_C}$. This notion of completeness differs from our notion of precision as well by similar arguments (although the two notions of completeness do not coincide).

to completeness. Specifically, completeness is shown to be equivalent to *best preservation*, which requires that $s_a \in [\![\psi]\!]^{M_A}_{\mathrm{tt}}$ iff $\gamma(s_a) \subseteq [\![\psi]\!]^{M_C}$. This means that whenever all of the concrete states represented by an abstract state s_a satisfy ψ, then the abstract semantics enables to verify ψ in s_a.

Although our definition of precision w.r.t. $l \in Lit$, $\Box U_A$ and $\Diamond U_A$ (see Definition 3.3) resembles the notion of completeness (or best preservation), the precision of a precise abstract model guaranteed by our definition is rather different. In particular, our definition of precision does not ensure that $s_a \in [\![\psi]\!]^{M_A}_{\mathrm{tt}}$ whenever all of the concrete states represented by s_a satisfy ψ. Instead, only if s_a *enables verification* of ψ do we ensure that $s_a \in [\![\psi]\!]^{M_A}_{\mathrm{tt}}$. Enabling verification is a stronger property that takes into account the inductiveness of the semantics.

For example, even when using the exact HTS, which is precise, we are unable to verify $\Box(p \wedge q)$ using the given abstraction in Example 1.1, since s_a does *not* enable verification of $\Box(p \wedge q)$ (see Definition 3.4), i.e., the abstraction itself is not precise enough. Intuitively, this results from the fact that there is no abstract state that represents the concrete states that satisfy both p and q in this example: In fact, there is only one such concrete state, s'_c, in this example, but every abstract state that represents it also represents additional concrete states that do not satisfy either p or q. Thus, there is no abstract state that enables verification of $p \wedge q$. Once precision is lost w.r.t. $p \wedge q$, the inductive semantics is unable to recover this information and verify $\Box(p \wedge q)$. Note, however, that in order to obtain completeness, or equivalently best preservation, s_a should satisfy $\Box(p \wedge q)$, since the only concrete state represented by s_a satisfies $\Box(p \wedge q)$.

The fact that we do not obtain completeness (or strong preservation) by our precision definition is not surprising. When using an inductive definition of the semantics, as we do in this paper, completeness w.r.t. the semantics requires pointwise completeness, which means completeness w.r.t. each of the operators. As shown in [27,28], the ability to obtain pointwise completeness is a property of the *abstraction* (i.e., the abstract states). Intuitively, it requires that the abstraction enables to express not only the property we aim to verify, but also the intermediate properties that lead to it (such as $p \wedge q$ in the above example), as captured by our definition of "enabling verification". However, our precision does not require anything of the abstraction. In particular, we do not require that the abstraction enables to verify any property. We simply refer to the precision of an abstract model w.r.t. the *given* abstraction (S_A, γ). We wish to make the most of the *given* abstraction, in contrast to e.g. applying completion methods on the abstraction in order to make the abstraction itself more precise (or complete).

4 Decreasing the Model Checking Cost

Using the exact HTS as an abstract model ensures maximal precision. Yet, it involves an exponential blowup (even with the suggested optimization). In this section we suggest an efficient model checking for the alternation-free μ-calculus, 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 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 [32] 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, $true$ or $false$ are *terminal nodes* (they have no outgoing edges). Nodes whose subformulas are of the form Z or $\eta Z.\psi$

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

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

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

Fig. 2. Rules for product graph construction

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, based on the 3-valued semantics.

The coloring algorithm of [32] 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.

The witness-coloring phase analyzes nodes that remained uncolored after iterating the rules of the sons-coloring phase. Such nodes are necessarily a part of a non-trivial SCC which has a witness. Depending on the witness, one of the definite colors (T or F) is ruled out for the remaining uncolored nodes, yet another phase is needed to decide between ? and the remaining definite color. For example, a μ-witness rules out the T-color, as infinite paths cannot contribute to satisfaction of a μ (least fixpoint) formula. Thus, for an uncolored node n in such an SCC it remains to be checked if the condition for coloring n by F, which depends on n's type, can still hold for it, and if not color it ?. This is done similarly to the sons-coloring phase, except that the rules are now aimed at checking that n has no potential to be colored F. The remaining nodes are colored F.

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. That is, $\widetilde{R}^E$ contains all the (ordinary) transitions that participate in some hyper-transition in 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 $\heartsuit \in \{\Box, \diamondsuit\}$ 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 $\diamondsuit$-node. Thus, the coloring algorithm can be seen, in a sense, as exhaustively trying to find the justification for coloring each 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). This is because must hyper-sons whose nodes are not all colored F will not justify coloring n by F, and thus need not be checked. Moreover, if some subset of the F-colored sons of n comprises a must hyper-son, then so does the full set. Similarly, to conclude that n should not be colored F (as is done in the witness-coloring phase), it suffices to check that n's currently F-colored sons along with the uncolored sons (if exist) do not form a must hyper-son. If they do not, then the same holds for any of their subsets, and clearly other sets of nodes cannot form a must hyper-son whose nodes are all colored F. This means that n has no potential to have a must hyper-son whose nodes are all colored by F, and it is safe to conclude that it cannot be colored F. 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 φ, 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 (total) 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 repeatedly until all nodes are colored.

(1) *Sons-coloring phase.* Apply the following rules until none is applicable.
- A terminal node $s_a \vdash true$ ($s_a \vdash false$) is colored $T(F)$.
- 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. For example, during phase 1 it is possible to color a $\Box$-node ($\Diamond$-node) by ? before all of its sons are colored by checking that all of its $T(F)$-colored sons along with its uncolored sons do not form a may hyper son, and in addition

all of its $F(T)$-colored sons along with its uncolored sons do not form a must hyper son. Note that if one of these conditions holds at one time then it will remain valid, thus it need not be checked again.

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 $\forall\forall\exists$ or the $\forall\exists\exists$ condition (resp.) between the underlying states of the node n and the set of nodes B.*

The following theorem formalizes the correctness of the algorithm, by relating the colors of nodes in the product graph to truth values of the corresponding formulas in the corresponding states. To refer to formulas in the product graph which are not closed, we use the following notation. For a (possibly not closed) alternation free formula φ_1, φ_1^* denotes the result of replacing every free occurrence of $Z \in \mathcal{V}$ in φ_1 by $fp(Z)$. φ_1^* is always a closed formula.

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 $n = s_a \vdash \varphi_1 \in G$ the following holds:*

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

Proof: [sketch] The proof is by induction on the run of the coloring algorithm. For any node which is colored within the sons-coloring phase the correctness follows directly from the 3-valued semantics, combined with the fact that if *all* the T-colored sons of a node n do not comprise a must (may) hyper-son, then n does not have a must (may) hyper son whose nodes are all colored T. Similarly for F. Thus it is sufficient to check if the selected candidates comprise hyper sons, rather than considering all the possible subsets of sons, as described before.

As for nodes which are colored in the witness coloring phase, the proof consists of several steps. We demonstrate the idea of the proof for a Q_i with a witness Z of type μ (the proof for the case of a ν-witness is similar).

In this case nodes are either colored by ? (in phase 2a) or F (in phase 2b). The first step is thus to show that the remaining uncolored nodes in Q_i at the beginning of this phase should indeed not be colored T. Let $n = s_a \vdash \varphi_1$ be a node in Q_i. We show that if $[\![\varphi_1^*]\!]_3^{M_A^E}(s_a) = \mathrm{tt}$, i.e. n should be colored T, then n must have already been colored T in the sons-coloring phase. Thus none of the uncolored nodes should be colored T.

Suppose that $fp(Z) = \mu Z.\psi$. Since the abstract state space is finite, there exists

i such that $[\![\mu Z.\psi]\!]^{M_A^E}_{\mathrm{tt}} = [\![\psi^i]\!]^{M_A^E}_{\mathrm{tt}}$, where ψ^i denotes the unwinding of the fixpoint formula i times (see Section 2). Let φ_1 and s_a be such that $[\![\varphi_1^*]\!]^{M_A^E}_3(s_a) = \mathrm{tt}$, i.e. $s_a \in [\![\varphi_1^*]\!]^{M_A^E}_{\mathrm{tt}}$. By the definition of φ_1^*, $[\![\varphi_1^*]\!]^{M_A^E}_{\mathrm{tt}} = [\![\varphi_1[Z := \mu Z.\psi]]\!]^{M_A^E}_{\mathrm{tt}} = [\![\varphi_1]\!]^{M_A^E,\rho[Z:=[\![\mu Z.\psi]\!]^{M_A^E}_{\mathrm{tt}}]}_{\mathrm{tt}} = [\![\varphi_1]\!]^{M_A^E,\rho[Z:=[\![\psi^i]\!]^{M_A^E}_{\mathrm{tt}}]}_{\mathrm{tt}} = [\![\varphi_1[Z := \psi^i]]\!]^{M_A^E}_{\mathrm{tt}}$. Note that $\varphi_1[Z := \psi^i]$ is fixpoint-free, and it does not contain any free variable. Thus, one can follow the inductive definition of the 3-valued semantics for $\wedge$, $\vee$, $\Box$ and $\Diamond$ and construct a finite tree over pairs of states and formulas that explains why $s_a \in [\![\varphi_1[Z := \psi^i]]\!]^{M_A^E}_{\mathrm{tt}}$. The root of the tree is $(s_a, \varphi_1[Z := \psi^i])$. All the pairs (s'_a, φ') in the proof tree will be such that $s'_a \in [\![\varphi']\!]^{M_A^E}_{\mathrm{tt}}$. For example, for a pair of the form $(s'_a, \varphi'_1 \wedge \varphi'_2)$, both (s'_a, φ'_1) and (s'_a, φ'_2) will be included as sons in the proof tree. For a pair $(s'_a, \Box\varphi')$ some set $A'_a \times \{\varphi'\}$ such that $s'_a R^- A'_a$ is a may hyper-transition in M_A^E and $A'_a \subseteq [\![\varphi']\!]^{M_A^E}_{\mathrm{tt}}$ will be included. The property that all the pairs (s'_a, φ') in the tree are such that $s'_a \in [\![\varphi']\!]^{M_A^E}_{\mathrm{tt}}$ ensures that ψ^0 will not be included in the tree, as $[\![\psi^0]\!]^{M_A^E}_{\mathrm{tt}} = [\![false]\!]^{M_A^E}_{\mathrm{tt}} = \emptyset$. In addition, since no free variables or fixpoints exist in the tree, the subformulas become strictly shorter along paths of the tree. These two properties ensure that every path in the tree eventually reaches a formula that does not contain ψ^j as a subformula for any $j \geq 0$. These will be the leaves of the tree, which makes the tree finite (this results from the fact that we have explicitly unwound the fixpoint).

We now map every formula in the proof tree back to the original formula that produced it (by replacing ψ^j by Z), i.e. if $\varphi' = \varphi[Z := \psi^j]$, then we define $\sigma(\varphi') = \varphi$. This defines a mapping $\tilde{\sigma}$ from the nodes of the proof tree to the nodes of the product graph: $\tilde{\sigma}(s'_a, \varphi') = s'_a \vdash \sigma(\varphi')$. Since the leaves of the proof tree do not contain formulas of the form ψ^j for any $j \geq 0$, we are guaranteed that all the leaves of the proof tree are mapped to nodes in the product graph that do not contain Z, thus they belong to smaller Q_j's. This is the crucial observation as it means that these nodes were already colored by the time the witness coloring phase is applied on Q_i and by the induction hypothesis their coloring is correct, i.e. they are colored T. This provides the basis for an inductive argument (on the depth of the proof tree) that shows that for every node (s'_a, φ') in the proof tree, the corresponding node $\tilde{\sigma}(s'_a, \varphi')$ in the product graph could be colored T in the sons-coloring phase. The induction step follows by a case analysis on the type of subformulas, and results from the relation between the 3-valued semantics and the rules of the coloring in the sons-coloring phase. Since the sons-coloring phase is iterated as long as some rule is applicable, the corresponding nodes must have been indeed already colored T. For example, consider some pair $(s'_a, \Box\varphi')$ in the proof tree for which the tree contains as witness the pairs $A'_a \times \{\varphi'\}$ for some may hyper transition $s'_a R^- A'_a$ such that $A'_a \subseteq [\![\varphi']\!]^{M_A^E}_{\mathrm{tt}}$. Then first by the definition of $\widetilde{R}$, and since $\sigma(\Box\varphi') = \Box\sigma(\varphi')$, all of the nodes $A'_a \times \{\sigma(\varphi')\}$

to which $A'_a \times \{\varphi'\}$ are mapped by $\tilde{\sigma}$ are sons of the node $s'_a \vdash \sigma(\Box\varphi')$ that corresponds to $(s'_a, \Box\varphi')$ in the product graph. By the induction hypothesis, all of these sons get colored T in the sons-coloring phase. Thus, at latest when the last of them gets colored T, then the algorithm finds out that the set of currently T-colored sons of $s'_a \vdash \sigma(\Box\varphi')$, which is a superset of $A'_a \times \{\sigma(\varphi')\}$, is a may hyper son of $s'_a \vdash \sigma(\Box\varphi') = \tilde{\sigma}(s'_a \vdash \Box\varphi')$, and therefore colors it T during the sons-coloring phase. In particular, for the root of the proof tree, $(s_a, \varphi_1[Z := \psi^i])$, we have that $\tilde{\sigma}(s_a, \varphi_1[Z := \psi^i]) = s_a \vdash \varphi_1$, which ensures that $n = s_a \vdash \varphi_1$ already got colored T in the sons-coloring phase.

Now, for nodes which are colored in phase 2a, a similar analysis as in the sons-coloring phase, following the 3-valued semantics, combined with the fact that if *all* the F-colored sons along with the uncolored sons of a node n do not comprise a must (may) hyper-son then n does not have a must (may) hyper son whose nodes are all colored F, shows that the nodes colored in phase 2a should not be colored F. Together with the previous argument saying that they should not be colored T, this ensures the correctness of their coloring by ?.

To complete the proof it remains to show that the nodes $s_a \vdash \varphi_1$ which are colored in phase 2b should indeed be colored F. Alternatively, it suffices to show that every node $s_a \vdash \varphi_1$ that should *not* be colored F, i.e. $[\![\varphi_1^*]\!]_3(s_a) \neq \text{ff}$, is indeed already colored by a different color (T or ?) when this phase is reached. Again, since the state space is finite, there exists i such that $[\![\mu Z.\psi]\!]_{\text{ff}}^{M_A^E} = [\![\psi^i]\!]_{\text{ff}}^{M_A^E}$, where ψ^i denotes the unwinding of the fixpoint formula i times. In particular, $[\![\varphi_1^*]\!]_{\text{ff}}^{M_A^E} = [\![\varphi_1[Z := \psi^i]]\!]_{\text{ff}}^{M_A^E}$. The proof again uses a construction of a proof tree, except that now this is a proof tree that explains why $s_a \notin [\![\varphi_1[Z := \psi^i]]\!]_{\text{ff}}^{M_A^E}$. Here again the crucial observation is that ψ^0 cannot be part of the proof tree, since $[\![\psi^0]\!]_{\text{ff}}^{M_A^E} = [\![false]\!]_{\text{ff}}^{M_A^E} = S_A$, whereas all the nodes (s'_a, φ') in the proof tree are such that $s'_a \notin [\![\varphi']\!]_{\text{ff}}^{M_A^E}$. Thus all the leaves of the proof tree are mapped to nodes in smaller Q_i's, which are already colored correctly (i.e. $\neq F$) before phase 2b and by induction on the depth of the proof tree so are the rest of the nodes of the product graph which are mapped to internal nodes of the proof tree. For example, for a pair $(s'_a, \Box\varphi')$ the proof tree contains as witness a set $A'_a \times \{\varphi'\}$ such that A'_a contains at least one state s_a^* such that $s_a^* \notin [\![\varphi']\!]_{\text{ff}}^{M_A^E}$ from every must hyper transition $s'_a R^+ A_a^*$ in M_A^E. Thus, at latest in phase 2a, after the last of the nodes $s_a^* \vdash \sigma(\varphi')$ for $s_a^* \in A'_a$ gets colored $\neq F$ (this happens by the induction hypothesis), it holds that the set B of F-colored sons of $s'_a \vdash \sigma(\Box\varphi')$ along with its uncolored sons does not form a must hyper son, since for every must hyper transition of M_A^E at least one target state comprises a node which belongs to $A'_a \times \{\sigma(\varphi')\}$, and is thus colored $\neq F$ at this point, and does not belong to B. Thus, $s'_a \vdash \sigma(\Box\varphi')$ gets colored ?. $\square$

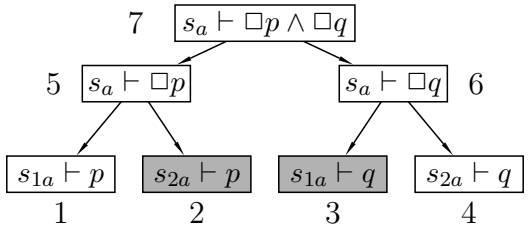


Fig. 3. A colored product graph. Grey nodes are colored ?, while white nodes are colored T.

Thus, for all the nodes 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. Note that if φ_1 is closed then $\varphi_1^* = \varphi_1$. Thus, for a node $n = s_a \vdash \varphi_1$ whose formula is closed the theorem immediately implies that the color of n in G matches the truth value of φ_1 in the state s_a of M_A^E. 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 3.13, $[\![\varphi]\!]^{M_C}(s_c) = \mathrm{tt}$ 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.

Example 4.4 Consider Example 1.1, where the purpose is to verify $\Box p \wedge \Box q$ in the concrete state s_c, abstracted by s_a (see Figure 1). This makes s_a the designated state. In this case $\widetilde{R} = \{(s_a, s_{1a}), (s_a, s_{2a})\}$. Thus, we obtain the product graph depicted in Figure 3, where $s_a \vdash \Box p \wedge \Box q$ is the initial node. Each node in the product graph comprises a (trivial) MSCC. Figure 3 also determines an order on the MSCCs, as indicated by the numbering of the nodes. The nodes $s_{1a} \vdash p$, $s_{2a} \vdash p$, $s_{1a} \vdash q$ and $s_{2a} \vdash q$ are colored as terminal nodes in the sons-coloring phase (in some arbitrary order). Their coloring is depicted in Figure 3. For example, $s_{1a} \vdash p$ is colored T since $p \in L_A(s_{1a})$, but

$s_{2a} \vdash p$ is colored ? since both $p \notin L_A(s_{2a})$ and $\neg p \notin L_A(s_{2a})$. Once $s_{1a} \vdash p$ is colored T, it is checked if the set $\{s_{1a} \vdash p\}$, which consists of the currently T-colored sons of the node $s_a \vdash \Box p$, forms a may hyper son of $s_a \vdash \Box p$. This is checked by checking if the $\forall\forall\exists$-condition holds for s_a and the set $\{s_{1a}\}$. Since the condition holds, $s_a \vdash \Box p$ is colored T. Similarly, once $s_{2a} \vdash q$ is colored T, it is checked if the set $\{s_{2a} \vdash q\}$ forms a may hyper son of $s_a \vdash \Box q$ (by checking the $\forall\forall\exists$-condition). Since the condition holds, $s_a \vdash \Box q$ is colored T. Thereafter, $s_a \vdash \Box p \wedge \Box q$ is colored T, and we conclude that the value of $\Box p \wedge \Box q$ in s_a is tt, thus s_c satisfies $\Box p \wedge \Box q$. In fact, the abstract model checking has "discovered" the two may hyper transitions $(s_a, \{s_{1a}\})$ and $(s_a, \{s_{2a}\})$, which are the ones needed for the verification of the formula in this example.

5 Abstraction-Refinement

Our abstract model checking ensures maximal precision w.r.t. the given abstraction. 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 [32] for models with ordinary transitions (with various optimizations that exploit the use of hyper transitions).

Definition 5.1 (Split) *Let S_C be a set of concrete states, let S_A and S'_A be two sets of abstract states and let $\gamma : S_A \to 2^{S_C}$, $\gamma' : S'_A \to 2^{S_C}$ be the corresponding concretization functions. We say that (S'_A, γ') is a* split of *(S_A, γ) iff there exists a (total) function $\rho : S'_A \to S_A$ such that for every $s_a \in S_A$: $\left(\bigcup_{\rho(s'_a)=s_a} \gamma'(s'_a)\right) = \gamma(s_a)$. If $\rho(s'_a) = s_a$ then s'_a is a* substate *of s_a.*

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 [11]. This is guaranteed by the following theorem.

Theorem 5.2 (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 a split of (S_A, γ). Then whenever $s'_a \in S'_A$ is a substate of $s_a \in S_A$ then $(M'_A, s'_a) \preceq (M_A, s_a)$.*

Proof: Suppose that $s'_a \in S'_A$ is a substate of $s_a \in S_A$. We show that $(M'_A, s'_a) \preceq (M_A, s_a)$. For this purpose we show that $H \subseteq S'_A \times S_A$ defined by $(s'_a, s_a) \in H$ iff $\rho(s'_a) = s_a$ (i.e. s'_a is a substate of s_a) is a hyper mixed simulation. Let $(s'_a, s_a) \in H$. We show that the three requirements hold.

(1) Suppose $l \in L_A(s_a)$. Then by the construction scheme, $\forall s_c \in \gamma(s_a) : l \in L_C(s_c)$. Since s'_a is a substate of s_a, then $\gamma'(s'_a) \subseteq \gamma(s_a)$, thus in particular

$\forall s_c \in \gamma'(s'_a) : l \in L_C(s_c)$ and by the construction scheme $l \in L_{A'}(s'_a)$. Thus $L_A(s_a) \subseteq L_{A'}(s'_a)$.

(2) Suppose $s_a R_A^- A_a$. Then by the construction, $\forall s_c \in \gamma(s_a)\ \forall s'_c\ [\ s_c R s'_c \Rightarrow \exists s_{a1} \in A_a$ s.t. $s'_c \in \gamma(s_{a1})\ ]$, i.e. $\forall s_c \in \gamma(s_a)\ \forall s'_c\ [\ s_c R s'_c \Rightarrow s'_c \in \gamma(A_a)\ ]$. Since s'_a is a substate of s_a, then $\gamma'(s'_a) \subseteq \gamma(s_a)$, thus in particular $\forall s_c \in \gamma'(s'_a)\ \forall s'_c\ [\ s_c R s'_c \Rightarrow s'_c \in \gamma(A_a)\ ]$. Let $A'_a \subseteq S'_A$ be the set consisting of all the substates of states in A_a. By definition of a split, $\left(\bigcup_{\rho(s'_a)=s_a} \gamma'(s'_a)\right) = \gamma(s_a)$, meaning that $\gamma'(A'_a) = \gamma(A_a)$. Therefore the following holds: $\forall s_c \in \gamma'(s'_a)\ \forall s'_c\ [\ s_c R s'_c \Rightarrow s'_c \in \gamma'(A'_a)\ ]$. This implies that $s'_a R_{A'}^- A'_a$. Moreover, $(A'_a, A_a) \in H^{\forall\exists}$ since for every $s'_{a1} \in A'_a$, at least one of its superstates s_{a1} is in A_a (otherwise s'_{a1} would not be included in A'_a), and as such $(s'_{a1}, s_{a1}) \in H$. Thus $\forall s'_{a1} \in A'_a\ \exists s_{a1} \in A_a : (s'_{a1}, s_{a1}) \in H$.

(3) Suppose $s_a R_A^+ A_a$. Then by the construction, $\forall s_c \in \gamma(s_a)\ \exists s_{a1} \in A_a\ \exists s'_c \in \gamma(s_{a1})$ s.t. $s_c R s'_c$, i.e. $\forall s_c \in \gamma(s_a)\ \exists s'_c \in \gamma(A_a)$ s.t. $s_c R s'_c$. Since s'_a is a substate of s_a, then $\gamma'(s'_a) \subseteq \gamma(s_a)$, thus in particular $\forall s_c \in \gamma'(s'_a)\ \exists s'_c \in \gamma(A_a)$ s.t. $s_c R s'_c$. Again, let $A'_a \subseteq S'_A$ be the set consisting of all the substates of states in A_a. As before $\gamma'(A'_a) = \gamma(A_a)$. Thus the following holds: $\forall s_c \in \gamma'(s'_a)\ \exists s'_c \in \gamma'(A'_a)$ s.t. $s_c R s'_c$. This implies that $s'_a R_{A'}^+ A'_a$. Moreover, as before $(A'_a, A_a) \in H^{\forall\exists}$.

□

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 [11].

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

6 Conclusion

We have investigated the precision and model checking complexity of 3-valued abstract models that preserve the full μ-calculus.

In order to evaluate precision of models, we have suggested a new definition of precision of 3-valued abstract models, which measures the precision of a model compared to the information retained in the abstract states and abstraction mapping. Namely, the abstract states define the "resolution" through which one can look at the concrete states at every point during the inductive evaluation of a property. An abstract model is precise if it enables to verify or falsify

every property that the resolution of the abstract states enables to verify or falsify, resp.

Examining previously suggested abstract models using our new definition revealed that may transitions do not enable to achieve maximal precision. We have therefore suggested a new class of models that use may hyper transitions to over approximate the concrete transitions. We proposed a construction of a precise abstract model of this class.

Hyper transitions make the size of the model exponential in the number of abstract states. To avoid this exponential blowup, which already existed in previously suggested models that use must hyper transitions, we have suggested a new abstract model checking algorithm for the alternation free μ-calculus, in which the hyper transitions are computed by need. As a result, the model checking complexity reduces to $O(|S_A|^2 \times |\varphi|)$, without compromising its 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. Finally, we have incorporated our abstract model checking into an abstraction-refinement algorithm, where the refinement is monotonic in terms of the precision of the models before and after refinement.

References

[1] S. Shoham, O. Grumberg, 3-valued abstraction: More precision at less cost., in: Twenty-First Annual IEEE Symposium on Logic In Computer Science (LICS), Seattle, Washington, 2006, pp. 399–410.

[2] E. Clarke, O. Grumberg, D. Peled, Model Checking, MIT press, 1999.

[3] G. Bruns, P. Godefroid, Model checking partial state spaces with 3-valued temporal logics, in: Computer Aided Verification, 1999, pp. 274–287.

[4] D. Kozen, Results on the propositional μ-calculus, TCS 27.

[5] K. G. Larsen, Modal specifications, in: J. Sifakis (Ed.), Proceedings of the 1989 International Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, France, Vol. 407 of Lecture Notes in Computer Science, Springer-Verlag, 1989.

[6] K. Larsen, B. Thomsen, A modal process logic, in: Proceedings of Third Annual Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, 1988, pp. 203–210.

[7] D. Dams, R. Gerth, O. Grumberg, Abstract interpretation of reactive systems, ACM Transactions on Programming Languages and Systems (TOPLAS) 19 (2).

[8] K. Namjoshi, Abstraction for branching time properties, in: Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03), Vol. 2725 of LNCS, Springer, Boulder, CO, USA, 2003, pp. 288–300.

[9] D. Dams, K. Namjoshi, The existence of finite abstractions for branching time model checking, in: 19th IEEE Symposium on Logic in Computer Science (LICS), IEEE Computer Society, 2004, pp. 335–344.

[10] L. de Alfaro, P. Godefroid, R. Jagadeesan, Three-valued abstractions of games: Uncertainty, but with precision, in: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS), 2004, pp. 170–179.

[11] S. Shoham, O. Grumberg, Monotonic abstraction-refinement for CTL, in: 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Vol. 2988 of LNCS, Barcelona, Spain, 2004, pp. 546–560.

[12] K. Larsen, L. Xinxin, Equation solving using modal transition systems, in: J. Mitchell (Ed.), Proceedings of the Fifth Annual IEEE Symp. on Logic in Computer Science (LICS), IEEE Computer Society Press, 1990, pp. 108–117.

[13] P. Cousot, R. Cousot, Abstract interpretation frameworks, J. Logic and Computation 2 (1992) 511–547.

[14] P. Cousot, R. Cousot, Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, in: Proceedings of the Conference on Programming Language Implementation and Logic Programming (PLILP'92), Springer-Verlag, 1992, pp. 269–295, lecture Notes in Computer Science 631.

[15] R. Cleaveland, P. Iyer, D. Yankelevich, Optimality in abstraction of model checking, in: Static Analysis Symposium (SAS), 1995, pp. 51–63.

[16] D. A. Schmidt, Closed and logical relations for over- and under-approximation of powersets, in: Static Analysis Symposium (SAS), 2004, pp. 22–37.

[17] A. Gurfinkel, O. Wei, M. Chechik, Systematic construction of abstractions for model-checking, in: Conference on Verification, Model Checking and Abstract Interpretation (VMCAI), 2006.

[18] P. Cousot, R. Cousot, Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: popl4, Los Angeles, California, 1977, pp. 238–252.

[19] P. Godefroid, M. Huth, R. Jagadeesan, Abstraction-based model checking using modal transition systems, in: Proceedings of CONCUR'01, 2001.

[20] G. Bruns, P. Godefroid, Generalized model checking: Reasoning about partial state spaces, in: CONCUR'00, Vol. 1877, 2000, pp. 168–182.

[21] P. Godefroid, R. Jagadeesan, Automatic abstraction using generalized model checking, in: Proc. of Conference on Computer-Aided Verification (CAV), Vol. 2404 of LNCS, Springer-Verlag, Copenhagen, Denmark, 2002, pp. 137–150.

[22] C. S. Pasareanu, R. Pelánek, W. Visser, Concrete model checking with abstract matching and refinement, in: Computer Aided Verification (CAV), 2005, pp. 52–66.

[23] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, Pacific J. Math 5 (1955) 285–309.

[24] M. Huth, R. Jagadeesan, D. Schmidt, Modal transition systems: A foundation for three-valued program analysis, in: European Symposium on Programming (ESOP'01), Vol. 2028, 2001, pp. 155–169.

[25] P. Godefroid, R. Jagadeesan, On the expressiveness of 3-valued models, in: Proceedings of VMCAI'2003 (4th Conference on Verification, Model Checking and Abstract Interpretation), Vol. 2575 of LNCS, Springer-Verlag, New York, 2003, pp. 206–222.

[26] P. Cousot, R. Cousot, Systematic design of program analysis frameworks, in: Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages (POPL), ACM, New York, NY, USA, 1979, pp. 269–282.

[27] R. Giacobazzi, F. Ranzato, F. Scozzari, Making abstract interpretations complete, Journal of the ACM 47 (2) (2000) 361–416.

[28] R. Giacobazzi, E. Quintarelli, Incompleteness, counterexamples, and refinements in abstract model checking, in: Static Analysis Symposium (SAS), LNCS 2126, Springer Verlag, 2001, pp. 356–373.

[29] D. Schmidt, Comparing completeness properties of static analyses and their logics, in: Asian Symp. Prog. Lang. Systems (APLAS'06), LNCS 4279, Springer Verlag, 2006, pp. 183–199.

[30] F. Ranzato, F. Tapparo, Strong preservation as completeness in abstract interpretation, in: Proc. European Symp. Programming, LNCS 2986, Springer Verlag, 2004, pp. 18–32.

[31] F. Ranzato, F. Tapparo, Strong preservation of temporal fixpoint-based operators by abstract interpretation, in: Conference on Verification, Model Checking and Abstract Interpretation (VMCAI), LNCS 3855, Springer Verlag, 2006, pp. 332–347.

[32] S. Shoham, O. Grumberg, A game-based framework for CTL counterexamples and 3-valued abstraction-refinement, in: Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03), Vol. 2725 of LNCS, Springer, Boulder, CO, USA, 2003, pp. 275–287, to appear in TOCL.

A Handling Multiple Initial States

So far we considered the verification problem of a formula in a specific concrete state. We now extend the discussion to the case where the concrete Kripke structure M_C has a *set* of initial states, denoted S_{0C}, with the usual meaning, that M_C satisfies φ, denoted $M_C \models \varphi$, if $\forall s_0 \in S_{0C} : [\![\varphi]\!]^{M_C}(s_0) = \text{tt}$. Otherwise, M_C falsifies φ, denoted $M_C \not\models \varphi$.

Typically, when the concrete model has a set of initial states, so does the abstract model. For example, in a GTS [11] the set of abstract initial states S_{0A} has to be some set such that

$\forall\exists\mathbf{(1)}$: $\forall s_{0c} \in S_{0C}\ \exists s_{0a} \in S_{0A}$ s.t. $s_{0c} \in \gamma(s_{0a})$, and
$\forall\exists\mathbf{(2)}$: $\forall s_{0a} \in S_{0A}\ \exists s_{0c} \in S_{0C}$ s.t. $s_{0c} \in \gamma(s_{0a})$.

$\forall\exists(1)$ is needed to preserve truth, as it ensures that the initial states of the abstract model represent *all* the concrete initial states. On the other hand, $\forall\exists(2)$ is needed to preserve falsity, as it ensures that each abstract initial state represents at least one concrete initial state. For example, in [11], S_{0A} is built such that $s_{0a} \in S_{0A}$ iff $\exists s_{0c} \in S_{0C}$ s.t. $s_{0c} \in \gamma(s_{0a})$.

This construction is precise if the abstract states represent disjoint sets of concrete states. Yet, similarly to the imprecision introduced by the may transitions when the abstract states are not necessarily disjoint, the same problem occurs with respect to the initial states of an abstract model.

In particular, suppose that some concrete initial state s_{0c} is represented by two abstract states: s_a in which φ_1 is true, but φ_2 is indefinite, and s'_a in which φ_2 is true, but φ_1 is indefinite. Then considering s_a as the only initial state will enable verification of φ_1 but not φ_2, and vice versa for s'_a. Yet, no choice of a set of initial states will enable verification of both formulas, even if s_{0c} is the only initial state: including s_a in S_{0A} will prevent verifying φ_2 and including s'_a will prevent verifying φ_2.

This example demonstrates that sometimes different sets of initial abstract states need to be considered for different properties. Therefore, to get a precise abstract model, one needs to allow multiple sets of initial states, with the meaning that any one of them suffices to verify or falsify a property.

Thus, rather than a set of initial states, the class of HTSs is extended by a set of sets of initial states $\mathcal{S}_0 \subseteq 2^{S_A}$, with the meaning that each of the sets in $\mathcal{S}_0$ is a "legal" set of initial states, i.e., it satisfies the $[\forall\exists(1)]$ and $[\forall\exists(2)]$ conditions. In the exact HTS M_A^E, $\mathcal{S}_0$ will consist of *all* the sets that satisfy these conditions.

An extended HTS satisfies φ, denoted $M_A \models^3 \varphi$, if there exists $S_{0A} \in \mathcal{S}_0$ where all the states satisfy φ. It falsifies φ, denoted $M_A \not\models^3 \varphi$, if there exists $S_{0A} \in \mathcal{S}_0$ where at least one state falsifies φ. Otherwise the value of φ in M_A is indefinite, denoted $M_A \models^{?3} \varphi$. Provided that the sets in $\mathcal{S}_0$ fulfill conditions $\forall\exists(1)$ and $\forall\exists(2)$, this ensures preservation of both truth and falsity.

Here again, instead of checking for each possible set of abstract states if it should be included in $\mathcal{S}_0$ (which requires two $\forall\exists$ checks), and then checking if it enables verification or falsification of φ, one may use a similar technique as was used for the hyper transitions and choose the candidates more carefully.

The idea is to apply the previous model checking algorithm by setting S_d to $\{s_a \mid \exists s_{0c} \in S_{0C} \text{ s.t. } s_{0c} \in \gamma(s_{0a})\}$. This is the maximal set that fulfills condition $\forall\exists(2)$. Thus, the sets in $\mathcal{S}_0$ in the exact HTS are exactly all the subsets of S_d that fulfill $\forall\exists(1)$ (including S_d itself). When the coloring is over, do the following.

(1) If at least one initial node n is colored F, then $M_C \not\models \varphi$. This is because $n = s_a \vdash \varphi$ for some $s_a \in S_d$. Since $S_d \in \mathcal{S}_0$ this implies that $M_A^E \not\models^3 \varphi$.
(2) Otherwise, let ${S_0}^T = \{s_a \in S_d \mid n = s_a \vdash \varphi \text{ is colored } T\}$ be the set of underlying states of the initial nodes that are colored T. If ${S_0}^T$ fulfills the $\forall\exists(1)$ condition, then it is a "legal" set of initial states, in which all of the states satisfy φ, meaning that $M_A^E \models^3 \varphi$, and thus $M_C \models \varphi$.
(3) If none of the above holds, then $M_A^E \models^{?3} \varphi$, which means that the abstraction is not precise enough. The correctness of this conclusion results from the fact that if there existed a possible set of initial states in $\mathcal{S}_0$ that falsifies φ, then it would have included a state from S_d that falsifies φ, in which case the first item would have applied. Similarly, if there existed a possible set of initial states that enables verification of φ, then it would have clearly been a subset of ${S_0}^T$, thus the second item would have applied.