Model Checking with Formula-Dependent Abstract Models

Alexander Asteroth, Christel Baier, Ulrich Aßmann

Universität Bonn, Römerstr. 164, 53117 Bonn, Germany
{aster,assmann,baier}@cs.uni-bonn.de

Abstract. We present a model checking algorithm for $\forall CTL$ (and full CTL) which uses an iterative abstraction refinement strategy.
It terminates at least for all transition systems $\mathcal{M}$ that have a finite simulation or bisimulation quotient. In contrast to other abstraction refinement algorithms, we always work with abstract models whose sizes depend only on the length of the formula Φ (but not on the size of the system, which might be infinite).

1 Introduction

The state explosion problem is still the major problem for applying model checking to systems of industrial size. Several techniques have been suggested to overcome this limitation of model checking; including symbolic methods with BDDs [6,33] or SAT-solvers [4], partial order reduction [35,22,40], compositional reasoning [29,21] and abstraction [11,26,13,27,29,16,19]. See [14] for an overview.
In this paper, we concentrate on abstraction in a temporal logical setting. Let $\mathcal{M}$ be the concrete model that we want to verify against a temporal logical formula Φ. The rough idea of the (exact) abstraction approach is to replace $\mathcal{M}$ by a much smaller abstract model $\mathcal{A}_\alpha$ with the *strong preservation* property stating that $\mathcal{A}_\alpha \models \Phi$ iff $\mathcal{M} \models \Phi$. The subscript α stands for an abstraction function that describes the relation between concrete and abstract states. In the simplest case, α is just a function from the concrete state space S to the abstract state space. (the state space of the abstract model $\mathcal{A}_\alpha$). For instance, dealing with the abstraction function α that assigns to each concrete state s its (bi-)simulation equivalence class, we get the (bi-)simulation quotient system $\mathcal{M}_{bis}$ or $\mathcal{M}_{sim}$, for which strong preservation holds if Φ is a CTL^* resp. $\forall CTL^*$ formula [5,13].

Algorithm 1 Schema of the Abstraction Refinement Approach.

construct an initial abstract model $\mathcal{A}_0$; $i := 0$;
REPEAT
  Model_Check($\mathcal{A}_i, \Phi$);
  IF $\mathcal{A}_i \not\models \Phi$ **THEN** $\mathcal{A}_{i+1}$:= Refinement($\mathcal{A}_i$,Φ) **FI**;
  $i := i + 1$;
UNTIL $\mathcal{A}_{i-1} \models \Phi$ or $\mathcal{A}_i = \mathcal{A}_{i-1}$;
IF $\mathcal{A}_{i-1} \models \Phi$ **THEN** return "yes" **ELSE** return "no" **FI**.

G. Berry, H. Comon, and A. Finkel (Eds.): CAV 2001, LNCS 2102, pp. 155–168, 2001.

If Φ is fixed these are unnecessarily large. In general, *conservative* abstractions that rely on the *weak preservation* property, stating that $\mathcal{A}_\alpha \models \Phi$ implies $\mathcal{M} \models \Phi$, yield much smaller abstract models. Such models can be used in the abstraction refinement schema shown in Algorithm 1 (e.g. [10,18,26,23,12]). Here, Model_Check(...) denotes any standard model checking algorithm and Refinement(...) an operator that adds further information about the original system $\mathcal{M}$ to $\mathcal{A}_i$ to obtain an abstract slightly more "concrete" model. A necessary property that ensures partial correctness of the above abstraction refinement technique is the strong preservation property for the final abstract model which might be obtained when no further refinement steps are possible.

The major difficulty is the design of a refinement procedure which on one hand should add enough information to the abstract model such that the "chances" to prove or disprove the property Φ in the next iteration increase in a reasonable measure while on the other hand the resulting new abstract model $\mathcal{A}_{i+1}$ should be reasonable small. The first goal can be achieved by specification-dependent refinement steps such as counterexample guided strategies [10,26,12] where the current abstract model $\mathcal{A}_i$ is refined according to an error trace that the model checker has returned for $\mathcal{A}_i$ or by strategies, that work with under- and/or overapproximations for the satisfaction relation $\models_{\mathcal{M}}$ of the concrete model, e.g. [18,28,31,36]. To keep the abstract models reasonable small two general approaches can be distinguished. One approach focusses on small symbolic BDD representations of the abstract models (e.g. [10,25,31,36,15]), while other approaches attempt to minimize the number of abstract states (e.g. [11,13,27,19]). While most of the fully automatic methods are designed for very large but finite concrete systems, most abstraction refinement techniques for infinite systems are semi-automatic and use a theorem prover to perform the refinement step or to provide the initial model $\mathcal{A}_0$ [17,23,9,1,38]. An entirely automatic abstraction technique that can treat infinite systems is presented in [34].

Our Contribution: In this paper, we present an abstraction refinement algorithm that works with abstract models with a *fixed* state space that just depends on the specification (temporal logical formula) but not on the concrete system. In our approach, the concrete system $\mathcal{M}$ to be verified is an ordinary (very large or infinite) transition system. We use the general abstraction framework suggested in [19] and deal with abstract models $\mathcal{A}_i$ with two transition relations. Although our ideas work for full *CTL*, we provide the explanations for the sublogic $\forall$*CTL* for which the formalisms are simpler.

The rough idea of our algorithm is the use of abstract models $\mathcal{A}_i$ that are approximations of $\mathcal{A}_\Phi$, the abstract model that results from the original model $\mathcal{M}$ when we collapse all states that satisfy the same subformulas of Φ. (Here, Φ is the formula we want to check for $\mathcal{M}$.) Of course, the computation of the abstract model $\mathcal{A}_\Phi$ would be at least as hard as model checking the original system $\mathcal{M}$. Anyway, we can use the state space of $\mathcal{A}_\Phi$ (which consists of sets of subformulas of Φ or their negations) for the abstract models $\mathcal{A}_i$. Thus, the size of any of the abstract models $\mathcal{A}_i$ is at most exponential in the length $|\Phi|$ of the formula; independent on the size of the concrete system which might be infinite. Any abstract model $\mathcal{A}_i$ is equipped with an abstraction function α_i which stands for *partial knowledge* about the satisfaction relation $\models_{\mathcal{M}}$ in the concrete system $\mathcal{M}$. The abstraction function α_i maps any concrete state s to the abstract state $\sigma = \alpha_i(s)$ in $\mathcal{A}_i$ consisting of those subformulas Ψ of Φ where we already know that $s \models_{\mathcal{M}} \Psi$ for all

$\Psi \in \sigma$ and all those negated subformulas $\neg\Psi$ where $s \not\models_{\mathcal{M}} \Psi$ is already shown. Refining $\mathcal{A}_i$ means adding more information about the concrete satisfaction relation $\models_{\mathcal{M}}$; resulting in an abstract model $\mathcal{A}_{i+1}$ where $\alpha_{i+1}(s)$ is a superset of $\alpha_i(s)$. Partial correctness of our algorithm is guaranteed for (concrete) transition systems of arbitrary size. Our algorithm terminates at least if the concrete system has a finite simulation or bisimulation quotient. The only theoretical requirement for an entirely automatic implementation is the effectiveness of the dual predecessor predicate in the concrete system.

Related Work: Our methodology borrows ideas from many other abstraction refinement algorithms. We work with *under-* and *overapproximations* for the concrete satisfaction relation $\models_{\mathcal{M}}$ that we derive from the abstraction function α_i. Although such "sandwich" techniques are used by several other authors, e.g. [3,28,31], we are not aware any other method that is designed for general (possibly infinite) transition systems and works with abstract models of a fixed size. Our methodology is also close to the framework of [18] where an abstraction refinement algorithm for $\forall CTL$ and finite concrete transition systems is presented. [18] only needs underapproximations for the concrete satisfaction relation. The major difference to our algorithm is the treatment of formulas with a least or greatest fixed point semantics (such as $\forall\Diamond\Psi$ and $\forall\Box\Psi$) in the refinement step.[1] Abstraction techniques with under- and/or overapproximations that focus on abstract models with small BDD representations are presented in [31,36,15]. We also use ideas of stable partitioning algorithms for computing the quotient space with respect to simulation or bisimulation like equivalences [37,7,32,24,8]. However, instead of splitting blocks (sets of concrete states that are identified in the current abstract model) into new subblocks (and thus, creating new abstract states), our approach refines the abstract model by *moving subblocks* from one abstract state to another abstract state (which presents more knowledge about the satisfaction relation $\models_{\mathcal{M}}$).
The method is also loosely related to tableau based methods as presented in [30,39].

Outline: In Section 2, we explain our notation concerning transition systems, *CTL* and briefly recall the basic results on abstract interpretations which our algorithm relies on. The type of abstract models used in our algorithm is introduced in Section 3. Section 4 presents our abstraction refinement algorithm for $\forall CTL$ and sketches the ideas to handle full *CTL*. Section 5 concludes the paper.

2 Preliminaries

We expect some background knowledge on transition systems, temporal logic, model checking, abstraction and only explain the notations used throughout this paper. For further details see, e.g. [14].

Transition Systems : A transition system is a tuple $\mathcal{M} = (S, \rightarrow, I, AP, L)$ where S is a set of states, $I \subseteq S$ the set of initial states, AP a finite set of atomic propositions and $L : S \rightarrow 2^{AP}$ a labeling function which assigns to any state $s \in S$ the set $L(s)$ of atomic

[1] Our refinement operator works with a "one-step-lookahead" while [18] treats paths that might have length > 1. In fact, this explains why underapproximations are sufficient in the framework of [18] while we need both under- and overapproximations to mimic the standard least or greatest fixed point computation. The fact that we just refine according to single transitions (paths of length 1) makes it possible to treat infinite systems.

propositions that hold in s. $\rightarrow \subseteq S \times S$ denotes the transition relation. Let $Post(s) = \{s' \in S : s \rightarrow s'\}$, $\widetilde{Pre}(B) = \{s \in S : Post(s) \subseteq B\}$. A path in a transition system is a maximal sequence $\pi = s_0 \rightarrow s_1 \rightarrow \ldots$ of states such that $s_i \in Post(s_{i-1})$, $i = 1, 2, \ldots$. Here, maximality means that either π is infinite or ends in a terminal state (i.e., a state without successors).

Computation Tree Logic (CTL) : *CTL* (state) formulas in positive normal form are built from the following grammar.

$$\Phi ::= true \;\Big|\; a \;\Big|\; \neg a \;\Big|\; \Phi_1 \wedge \Phi_2 \;\Big|\; \Phi_1 \vee \Phi_2 \;\Big|\; \forall\varphi \mid \exists\varphi \qquad \varphi ::= \; \mathsf{X}\Phi \;\Big|\; \Phi_1 \mathsf{U} \Phi_2 \;\Big|\; \Phi_1 \tilde{\mathsf{U}} \Phi_2$$

with $a \in AP$. Here, X and U are the standard temporal modalities "Next step" and "Until" while $\tilde{\mathsf{U}}$ denotes "weak until" (also often called "unless").[2] Operators for modelling "eventually" or "always" are derived as usual, e.g. $\forall\Diamond\Phi = \forall true \mathsf{U} \Phi$ and $\forall\Box\Phi = \forall\Phi\tilde{\mathsf{U}} false$. The universal fragment of *CTL* (where the application of "$\exists$" is not allowed) is denoted by $\forall CTL$. Similarly, $\exists CTL$ denotes the existential fragment of *CTL*. The satisfaction relation $\models_{\mathcal{M}}$ for *CTL* formulas and transition systems $\mathcal{M}$ is defined in the standard way. The satisfaction set for Φ in $\mathcal{M}$ is given by $Sat_{\mathcal{M}}(\Phi) = \{s \in S : s \models_{\mathcal{M}} \Phi\}$. We write $\mathcal{M} \models \Phi$ iff Φ holds for any initial state, i.e., iff $I \subseteq Sat_{\mathcal{M}}(\Phi)$. Although negation is only allowed on the level of atomic propositions, we shall use expressions of the type $\neg\Psi$ (with the intended meaning $s \models_{\mathcal{M}} \neg\Psi$ iff $s \not\models_{\mathcal{M}} \Psi$).

Abstract Interpretations : Let $\mathcal{M} = (S, \rightarrow, I, AP, L)$ be a transition system that models the "concrete system" (that we want to verify). Let S_A be an arbitrary set of "abstract states". In what follows, we use the Latin letter s for concrete states (i.e., states $s \in S$) and the greek letter σ for abstract states (i.e., states $\sigma \in S_A$). An *abstraction function* for $\mathcal{M}$ (with range S_A) is a function $\alpha : S \rightarrow S_A$ such that $\alpha(s) = \alpha(s')$ implies $L(s) = L(s')$. The induced *concretization function* $\gamma : S_A \rightarrow 2^S$ is just the inverse image function $\gamma = \alpha^{-1}$ (that is, $\gamma(\sigma) = \{s \in S : \alpha(s) = \sigma\}$). We use the results of [19] and associate with α two transition relations $\rightarrow_\alpha$ (which we shall use to get underapproximations for the satisfaction sets $Sat_{\mathcal{M}}(\cdot)$) and $\rightsquigarrow_\alpha$ (yielding overapproximations). They are given by

$$\sigma \rightarrow_\alpha \sigma' \text{ iff } \exists\, s \in \gamma(\sigma)\ \exists\, s' \in \gamma(\sigma') \text{ s.t. } s \rightarrow s'$$
$$\sigma \rightsquigarrow_\alpha \sigma' \text{ iff } \forall s \in \gamma(\sigma)\ \exists\, s' \in \gamma(\sigma') \text{ s.t. } s \rightarrow s'.$$

For any (abstract) path $\sigma_0 \rightsquigarrow_\alpha \sigma_1 \rightsquigarrow_\alpha \ldots$ and concrete state $s_0 \in \gamma(\sigma_0)$, there is a (concrete) path $s_0 \rightarrow s_1 \rightarrow \ldots$ in $\mathcal{M}$ such that $\alpha(s_i) = \sigma_i$, $i = 0, 1, \ldots$ while the corresponding statement for $\rightarrow_\alpha$ may be wrong. Vice versa, any (concrete) path $s_0 \rightarrow s_1 \rightarrow \ldots$ in $\mathcal{M}$ can be lifted to a path $\sigma_0 \rightarrow_\alpha \sigma_1 \rightarrow_\alpha \ldots$ where $\sigma_i = \alpha(s_i)$.

Let $\mathcal{U} = (S_A, \rightarrow_\alpha, I_\alpha, AP, L_\alpha)$ and $O = (S_A, \rightsquigarrow_\alpha, I_\alpha, AP, L_\alpha)$ be the transition system with state space S_A where the set of abstract initial states is $I_\alpha = \alpha(I) = \{\alpha(s) : s \in I\}$. The abstract labeling function $L_\alpha : A \rightarrow 2^{AP}$ is given by $L_\alpha(\sigma) = \alpha(s)$ for some/all concrete states $s \in \gamma(\sigma)$. Then, we have weak preservation of the following type.

[2] Any ordinary *CTL* formula (where also negation is allowed in the state formulas) can be transformed into positive normal form. Note that the dual to the until operator (often called the "release operator") can be obtained by $\neg(\neg\Phi_1 \mathsf{U} \neg\Phi_2) = (\neg\Phi_1 \wedge \Phi_2)\tilde{\mathsf{U}}(\Phi_1 \wedge \Phi_2)$.

Lemma 1. *(cf.* [13,27,19]*) Let s be a concrete state.*

(1) If Ψ *is a* $\forall$*CTL formula and* $\alpha(s) \models_{\mathcal{U}} \Psi$ *then* $s \models_{\mathcal{M}} \Psi$.
(2) If Ψ *is a* $\exists$*CTL formula and* $\alpha(s) \models_{O} \Psi$ *then* $s \models_{\mathcal{M}} \Psi$.

3 Abstract Φ-Models

Throughout this paper, we assume a fixed concrete transition system $\mathcal{M} = (S, \rightarrow, I, AP, L)$ without terminal states and a $\forall$*CTL* formula Φ. When we refer to a subformula then we mean a formula which is not a constant *true* or *false*. $sub(\Phi)$ denotes the set of all subformulas of Φ. We may assume that, $AP \subseteq sub(\Phi)$. We refer to any subformula of Φ of the form $\Psi = \forall\varphi$ as a $\forall$subformula of Φ.

The Abstract State Space S_Φ : Let $cl(\Phi)$ denote the set of all subformulas Ψ of Φ and their negation $\neg\Psi$ (where we identify $\neg\neg a$ and a). I.e., $cl(\Phi) = sub(\Phi) \cup \{\neg\Psi : \Psi \in sub(\Phi)\}$. We define the set $S_\Phi \subseteq 2^{cl(\Phi)}$ as follows. S_Φ denotes the set of $\sigma \subseteq cl(\Phi)$ such that the following conditions (i) and (ii) hold. (i) for any atomic proposition $a \in AP$ and $\sigma \in S_\Phi$, either $a \in \sigma$ or $\neg a \in \sigma$. (ii) asserts the consistency of σ with respect to propositional logic and local consistency with respect to "until" and "weak until". We just mention the axioms for "until".[3]

1. If $\Psi_2 \in \sigma$ and $\forall\Psi_1 \mathsf{U} \Psi_2 \in sub(\Phi)$ then $\forall\Psi_1 \mathsf{U} \Psi_2 \in \sigma$.
2. If $\Psi_2 \notin \sigma$ and $\forall\Psi_1 \mathsf{U} \Psi_2 \in \sigma$ then $\Psi_1 \in \sigma$ (provided that $\Psi_1 \notin \{true, false\}$).
3. If $\neg\Psi_1, \neg\Psi_2 \in \sigma$ and $\forall\Psi_1 \mathsf{U} \Psi_2 \in sub(\Phi)$ then $\neg\forall\Psi_1 \mathsf{U} \Psi_2 \in \sigma$.
4. If $\neg\forall\Psi_1 \mathsf{U} \Psi_2 \in \sigma$ then $\neg\Psi_2 \in \sigma$.

The abstract models $\mathcal{U}_\Phi$ **and** O_Φ yield precise abstractions. Let $\alpha_\Phi : S \rightarrow S_\Phi$ be given by $\alpha_\Phi(s) = \{\Psi \in sub(\Phi) : s \models_{\mathcal{M}} \Psi\} \cup \{\neg\Psi : \Psi \in sub(\Phi),\ s \not\models_{\mathcal{M}} \Psi\}$. It is well-known [20] that for the abstract model that we get with the abstraction function α_Φ we just can establish the weak preservation property but do not have strong preservation. However, when we add a new atomic proposition a_Ψ for any $\forall$subformula Ψ of Φ then we get an abstract model for which a slight variant of the strong preservation property holds. Let

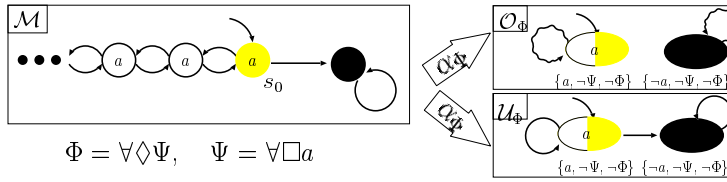
$$AP_\Phi = AP \cup \{a_\Psi : \Psi \text{ is a } \forall\text{subformula of } \Phi\}.$$

We put $a_\Psi = a$ if $\Psi = a$ is an atomic proposition. Let $L_\mathcal{U}, L_O : S_\Phi \rightarrow AP_\Phi$ be given by

$$L_\mathcal{U}(\sigma) = \{a_\Psi \in AP_\Phi : \Psi \in \sigma\}, \quad L_O(\sigma) = \{a_\Psi \in AP_\Phi : \neg\Psi \notin \sigma\}.$$

When dealing with underapproximations, we use the labeling function $L_\mathcal{U}$ while L_O will serve for the overapproximations. We define $\mathcal{U}_\Phi = (S_\Phi, \rightarrow_{\alpha_\Phi}, I_{\alpha_\Phi}, AP_\Phi, L_\mathcal{U})$ and $O_\Phi = (S_\Phi, \rightsquigarrow_{\alpha_\Phi}, I_{\alpha_\Phi}, AP_\Phi, L_O)$.

[3] For "weak until" we have essentially the same axioms as for "until". The propositional logical axioms are obvious; e.g. we require that "$\Psi \in \sigma$ implies $\neg\Psi \notin \sigma$" and the symmetric axiom "$\neg\Psi \in \sigma$ implies $\Psi \notin \sigma$". One of the axioms for conjunction is "$\Psi_1 \wedge \Psi_2 \in \sigma$ iff $\Psi_1 \in \sigma$ and $\Psi_2 \in \sigma$." Note that we do not require maximality; i.e., $\Psi, \neg\Psi \notin \sigma$ is possible if $\Psi \notin AP$.



Intuitively, the labelings $L_{\mathcal{U}}$ and L_O with the auxiliary atomic propositions a_Ψ shall encode the information about the satisfaction set $Sat_{\mathcal{M}}(\Psi)$ that might got lost with the abstract transition relations $\rightarrow_\alpha$ and $\rightsquigarrow_\alpha$.

Example: For the concrete system $\mathcal{M}$ shown in the picture above and the formula $\Phi = \forall\Diamond\forall\Box a$, $\mathcal{M} \not\models \Phi$ while $O_\Phi \models \Phi$. In our examples we depict concrete states by circles, abstract states by ellipses. Their names are written below while the corresponding labels are written inside the states.□

The Formulas $\overline{\Psi}$ and $\widetilde{\Psi}$: For each subformula Ψ of Φ we define new $\forall CTL$ formulas $\overline{\Psi}$ and $\widetilde{\Psi}$ by structural induction. If Ψ is *true*, *false* or a literal then $\overline{\Psi} = \widetilde{\Psi} = \Psi$. If $\Psi = \Psi_1 \vee \Psi_2$ then $\overline{\Psi} = \overline{\Psi_1} \vee \overline{\Psi_2}$ and $\widetilde{\Psi} = \widetilde{\Psi_1} \vee \widetilde{\Psi_2}$. Conjunction is treated in a similar way. The transformations for "next step", "until" and "weak until" make use of the new atomic propositions. For $\Psi = \forall X\Psi_0$ we put $\overline{\Psi} = (\forall X\overline{\Psi_0}) \vee a_\Psi$ and $\widetilde{\Psi} = (\forall X\widetilde{\Psi_0}) \wedge a_\Psi$. If $\Psi = \forall\Psi_1 U \Psi_2$ then we put $\overline{\Psi} = \forall\overline{\Psi_1} U (\overline{\Psi_2} \vee a_\Psi)$ and $\widetilde{\Psi} = (\forall\widetilde{\Psi_1} U \widetilde{\Psi_2}) \wedge a_\Psi$. Similarly, we treat weak until. It is easy to see that for any concrete state s and $\Psi \in cl(\Phi)$:

$$\alpha_\Phi(s) \models_{\mathcal{U}_\Phi} \overline{\Psi} \quad \text{iff} \quad \alpha_\Phi(s) \models_{O_\Phi} \widetilde{\Psi} \quad \text{iff} \quad s \models_{\mathcal{M}} \Psi.$$

In the example above, we get $\widetilde{\Phi} = (\forall\Diamond\widetilde{\Psi}) \wedge a_\Phi$ where $\widetilde{\Psi} = (\forall\Box a) \wedge a_\Psi$ and the desired property $O_\Phi \not\models \widetilde{\Phi}$.

Abstract Φ-Models : $\mathcal{U}_\Phi$ and O_Φ contain all information that we need to model check the original system $\mathcal{M}$ against the formula Φ. In our abstraction refinement algorithm we make use of abstract models which can be viewed as approximations of $\mathcal{U}_\Phi$ and O_Φ.

Definition 1. *An* abstract Φ-model *for $\mathcal{M}$ is a tuple $\mathcal{A} = (\alpha, \gamma, \mathcal{U}, O)$ consisting of an abstraction function $\alpha : S \rightarrow S_\Phi$ with $\alpha(s) \subseteq \alpha_\Phi(s)$ for any concrete state $s \in S$, the concretization function $\gamma = \alpha^{-1} : S_\Phi \rightarrow S$ and the two transition systems $\mathcal{U} = (S_\Phi, \rightarrow_\alpha, I_\alpha, AP_\Phi, L_{\mathcal{U}})$ and $O = (S_\Phi, \rightsquigarrow_\alpha, I_\alpha, AP_\Phi, L_O)$ where I_α, $\rightarrow_\alpha$, $\rightsquigarrow_\alpha$ are as in Section 2.* □

Intuitively, the sets $\alpha(s)$ consist of all subformulas Ψ of Φ where $s \models_{\mathcal{M}} \Psi$ has already been verified and all formulas $\neg\Psi$ where $s \not\models_{\mathcal{M}} \Psi$ has already been shown. However, there might be formulas $\Psi \in sub(\Phi)$ such that neither $\Psi \in \alpha(s)$ nor $\neg\Psi \in \alpha(s)$. For such formulas Ψ, we do not yet know whether $s \models_{\mathcal{M}} \Psi$.

Let $\mathcal{A} = (\alpha, \gamma, \mathcal{U}, O)$ be an abstract Φ-model. We associate with $\mathcal{A}$ two satisfaction relations. $\models_{\mathcal{U}}$ denotes the standard satisfaction relation for *CTL* and the transition system $\mathcal{U}$. As we assume that the concrete transition system $\mathcal{M}$ has no terminal states, all paths in $\mathcal{M}$ and $\mathcal{U}$ are infinite. However, the abstract transition system O might have terminal states. For O, we slightly depart from the standard semantics of *CTL*. For the finite paths in O, the satisfaction relation $\models_O$ treats weak until and until in the

same way. Let $\pi = \sigma_0 \rightsquigarrow_\alpha \sigma_1 \rightsquigarrow_\alpha \ldots \rightsquigarrow_\alpha \sigma_n$ be a finite path. Then, $\pi \models_O \Psi_1 \mathsf{U} \Psi_2$ iff $\pi \models_O \Psi_1 \widetilde{\mathsf{U}} \Psi_2$ iff either $\sigma_0, \sigma_1, \ldots, \sigma_n \models_O \Psi_1$ or there is some $k \in \{0, 1, \ldots, n\}$ with $\sigma_0, \sigma_1, \ldots, \sigma_{k-1} \models_O \Psi_1$ and $\sigma_k \models_O \Psi_2$.[4] The reason why we need this modification is that we "reverse" the result established by [19] stating that $\alpha(s) \models_O \Psi$ implies $s \models_{\mathcal{M}} \Psi$ for any $\exists$*CTL* formula Ψ (compare Lemma 1, part (2), and Lemma 2, part (b)) For infinite paths and any type of path formulas, we deal with the usual *CTL* semantics in O. Also for the next step and weak until operator and finite paths in O, we work with the usual semantics. (Thus, $\sigma \models_O \forall \mathsf{X} \Psi$ holds for all terminal states σ in O.)

Lemma 2. *For any concrete state $s \in S$ and $\Psi \in sub(\Phi)$:*

(a) If $\alpha(s) \models_{\mathcal{U}} \overline{\Psi}$ then $s \models_{\mathcal{M}} \Psi$.
(b) If $\alpha(s) \not\models_O \widetilde{\Psi}$ then $s \not\models_{\mathcal{M}} \Psi$.
(c) If $\Psi \in \alpha(s)$ then $\alpha(s) \models_{\mathcal{U}} \overline{\Psi}$.
(d) If $\neg\Psi \in \alpha(s)$ then $\alpha(s) \not\models_O \widetilde{\Psi}$.

Any abstract Φ-model $\mathcal{A} = (\alpha, \gamma, \mathcal{U}, O)$ induces under- and overapproximations for the sets $Sat_{\mathcal{M}}(\Psi) = \{s \in S : s \models_{\mathcal{M}} \Psi\}$, $\Psi \in sub(\Phi)$.

Definition 2. *Let $Sat^+_{\mathcal{A}}(\Psi) = \{s \in S : \neg\Psi \notin \alpha(s)\}$, $Sat^-_{\mathcal{A}}(\Psi) = \{s \in S : \Psi \in \alpha(s)\}$.* □

Lemma 3. *$Sat^-_{\mathcal{A}}(\Psi) \subseteq Sat_{\mathcal{M}}(\Psi) \subseteq Sat^+_{\mathcal{A}}(\Psi)$ for any $\Psi \in sub(\Phi)$.* □

Lemma 3 follows by $\alpha(s) \subseteq \alpha_\Phi(s)$. Clearly, given α or γ, the abstract Φ-model $\mathcal{A}$ is uniquely determined. Vice versa, given over- and underapproximations $Sat^+(\Psi)$ and $Sat^-(\Psi)$ for $Sat_{\mathcal{M}}(\Psi)$ there exists a unique abstract Φ-model $\mathcal{A}$ with $Sat^+(\Psi) = Sat^+_{\mathcal{A}}(\Psi)$ and $Sat^-(\Psi) = Sat^-_{\mathcal{A}}(\Psi)$.[5]

Definition 3. *$\mathcal{A} \models \Phi$ iff $\Phi \in \sigma$ for all abstract initial states σ and $\mathcal{A} \models \neg\Phi$ iff there is an abstract initial state σ with $\neg\Phi \in \sigma$.*[6] □

Clearly, $\mathcal{A} \models \Phi$ iff $I \subseteq Sat^-_{\mathcal{A}}(\Phi)$ iff $\Phi \in \alpha(s)$ for any concrete initial state s. Similarly, $\mathcal{A} \not\models \Phi$ iff there is a concrete initial state s such that $\neg\Phi \in \alpha(s)$. By Lemma 2(c,d):

Lemma 4. *If $\mathcal{A} \models \Phi$ then $\mathcal{M} \models \Phi$. If $\mathcal{A} \models \neg\Phi$ then $\mathcal{M} \not\models \Phi$.* □

Blocks and the Partition $\Pi_{\mathcal{A}}$: We refer to the sets $B = \gamma(\sigma)$, $\sigma \in S_\Phi$, as *blocks* in $\mathcal{M}$ with respect to $\mathcal{A}$. Clearly, the collection $\Pi_{\mathcal{A}}$ of all blocks in $\mathcal{M}_{\mathcal{A}}$ is a partition of the concrete state space S. It should be noticed that for any block $B \in \Pi_{\mathcal{A}}$ either $B \subseteq Sat^-_{\mathcal{A}}(\Psi)$ or $B \cap Sat^-_{\mathcal{A}}(\Psi) = \emptyset$. The same holds for $Sat^+_{\mathcal{A}}(\Psi)$.

4 An Abstraction Refinement Model Checking Algorithm

Our algorithm (sketched in Algorithm 2) uses the abstraction refinement schema of Algorithm 1. We start with an abstract Φ-model $\mathcal{A}_0$ and will successively refine the model $\mathcal{A}_i$ until $\mathcal{A}_i \models \Phi$ or $\mathcal{A}_i \models \neg\Phi$. The output of our algorithm (sketched in Algorithm 2) is clear from Lemma 4.

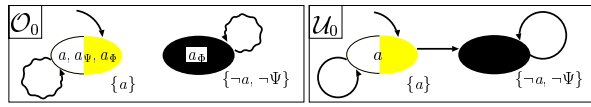
[4] Alternatively, when we interpret a path formula $\Phi = \forall\varphi$ over O then we may use the standard semantics for *CTL* but switch from $\forall\Psi_1 \mathsf{U} \Psi_2$ to the formula $\forall\Psi_1 \mathsf{U}(\Psi_2 \vee (\Psi_1 \wedge \forall \mathsf{X} \mathit{false}))$.

[5] Consider the model $\mathcal{A}$ induced by the abstraction function $\alpha(s) = \{\Psi : s \in Sat^-(\Psi)\} \cup \{\neg\Psi : s \notin Sat^+(\Psi)\}$.

[6] The reader should notice that $\mathcal{A} \not\models \Phi$ is *not* the same as $\mathcal{A} \models \neg\Phi$. $\mathcal{A} \not\models \Phi$ and $\mathcal{A} \not\models \neg\Phi$ is possible.

The initial abstract Φ-model is the abstract Φ-model $\mathcal{A}_0 = \mathcal{A}_{AP}$ that we get with the abstraction functions $\alpha_0 = \alpha_{AP} : S \to S_\Phi$ where $\alpha_{AP}(s) = \lceil L(s) \cup \{\neg a : a \in AP \setminus L(s)\} \rceil$. Here and in the following, $\lceil \sigma \rceil$ denotes the smallest element of S_Φ containing σ.[7]

The use of α_{AP} reflects the knowledge that all concrete states labeled with an atomic proposition a satisfy a while $\neg a$ holds for s if a is an atomic proposition not in $L(s)$. The status of more complex subformulas in Φ (whose truth value cannot be derived from the axioms for S_Φ) is still open. For the concrete system $\mathcal{M}$ and formula Φ depicted in the previous figure (Section 3), the initial abstract model $\mathcal{A}_0$ is as shown on below.



Algorithm 2 Main Procedure of the Abstraction Refinement Algorithm.

$\mathcal{A}_0 := \mathcal{A}_{AP}$; $i := 0$;
REPEAT
  $\mathcal{A}$:= Model_Check($\mathcal{A}_i, \Phi$);
  IF $\mathcal{A}_i \not\models \Phi$ and $\mathcal{A}_i \not\models \neg\Phi$ **THEN**
    FOR ALL ∀subformulas Ψ of Φ **DO**
      IF $Sat^+_{\mathcal{A}}(\Psi) \neq Sat^-_{\mathcal{A}}(\Psi)$ **THEN**
        $\mathcal{A} := \text{Refine}(\mathcal{A}, \Psi)$;
      ELSE
        replace Ψ by the atomic proposition a_Ψ
      FI
    OD
  FI
  $i := i+1$; $\mathcal{A}_i := \mathcal{A}$;
UNTIL $\mathcal{A}_i \models \Phi$ or $\mathcal{A}_i \models \neg\Phi$;
IF $\mathcal{A}_i \models \Phi$ **THEN** return "yes" **ELSE** return "no" **FI**.

Model Checking the Abstract Φ-Model: Let $\mathcal{A}_i = (\alpha, \gamma, \mathcal{U}, O)$ be the current abstract Φ-model. In any iteration, we apply a standard model checker that successively treats any ∀subformulas Ψ of Φ for both transition systems $\mathcal{U}$ and O.

Let Ψ be a ∀subformula of Φ. First, we apply a standard model checking routine for $\mathcal{U}$ and the formula $\overline{\Psi}$ to calculate the satisfaction set $Sat_{\mathcal{U}}(\overline{\Psi}) = \{\sigma \in S_\Phi : \sigma \models_{\mathcal{U}} \overline{\Psi}\}$. We derive the set $NewSat(\Psi) = \{\sigma \in S_\Phi : \Psi \notin \sigma,\ \sigma \models_{\mathcal{U}} \overline{\Psi}\}$ of all abstract states σ where $\overline{\Psi}$ now holds while $\overline{\Psi}$ did not hold in the previous iteration. By Lemma 2, part (a), we know that Ψ holds for all concrete states $s \in \bigcup\{\gamma(\sigma) : \sigma \in NewSat(\Psi)\}$. Thus, we can improve the underapproximation $Sat^-_{\mathcal{A}_i}(\Psi)$ of $Sat_{\mathcal{M}}(\Psi)$ by adding all blocks $\gamma(\sigma)$ where $\sigma \in NewSat(\Psi)$ to $Sat^-_{\mathcal{A}_i}(\Psi)$.

[7] If $\sigma \subseteq 2^{cl(\Phi)}$ meets all axioms concerning propositional consistencies then σ can be extended (according to the axioms that we require for S_Φ) to a least superset $\lceil\sigma\rceil \in S_\Phi$ that contains σ. E.g. for $\Phi = \forall a \mathsf{U} b$, $\lceil\{b\}\rceil = \{b, \Phi\}$.

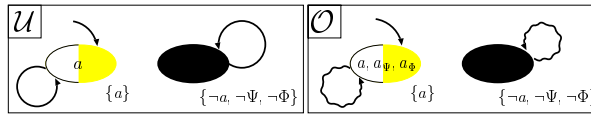
Second, we call a standard model checker for O and $\widetilde{\Psi}$ to obtain the set $NewSat(\neg\Psi) = \{\sigma \in S_\Phi : \neg\Psi \notin \sigma,\ \sigma \not\models_O \widetilde{\Psi}\}$ of all abstract states σ where $\widetilde{\Psi}$ is not satisfied while $\widetilde{\Psi}$ did hold for σ in the previous iteration. Lemma 2, part (b), yields that none of the concrete states $s \in \bigcup\{\gamma(\sigma) : \sigma \in NewSat(\neg\Psi)\}$ satisfies Ψ. Hence, we may remove the blocks $\gamma(\sigma)$ where $\sigma \in NewSat(\neg\Psi)$ from $Sat^+_{\mathcal{A}_i}(\Psi)$ (i.e., we improve the overapproximation).

Algorithm 3 The Model-Checking-Routine Model_Check($\mathcal{A}, \Phi$).

Let γ be the concretization function of $\mathcal{A}$.
FOR ALL $\forall$subformulas Ψ of Φ **DO**
  calculate the set $NewSat(\Psi) = \{\sigma \in S_\Phi \ \sigma \models_{\mathcal{U}} \overline{\Psi}$ and $\Psi \notin \sigma\}$;
  FOR ALL $\sigma \in NewSat(\Psi)$ **DO** $\gamma(\lceil \sigma \cup \{\Psi\} \rceil) := \gamma(\sigma) \cup \gamma(\lceil \sigma \cup \{\Psi\} \rceil)$;
  $\gamma(\sigma) := \emptyset$ **OD**;
  calculate the set $NewSat(\neg\Psi) = \{\sigma \in S_\Phi \ \sigma \not\models_O \widetilde{\Psi}$ and $\neg\Psi \notin \sigma\}$;
  FOR ALL $\sigma \in NewSat(\neg\Psi)$ **DO** $\gamma(\lceil \sigma \cup \{\neg\Psi\} \rceil) := \gamma(\sigma) \cup \gamma(\lceil \sigma \cup \{\neg\Psi\} \rceil)$;
  $\gamma(\sigma) := \emptyset$ **OD**;
OD
return the abstract Φ-model induced by γ.

Algorithm 3 combines the two model checking fragments and returns a new abstract Φ-model $\mathcal{A}' = \text{Model\_Check}(\mathcal{A}_i, \Phi)$ with the abstraction function α' where $\alpha'(s)$ arises from $\alpha(s)$ by adding Ψ if $\alpha(s) \in NewSat(\Psi)$ and adding $\neg\Psi$ if $\alpha(s) \in NewSat(\neg\Psi)$.[8]

Example : For the initial model $\mathcal{A}_0$ in the running example, $NewSat(\Psi) = NewSat(\Phi) = NewSat(\neg\Psi) = \emptyset$ while $NewSat(\neg\Phi)$ consists of the black abstract state $\sigma = \{\neg a, \neg\Psi\}$. Therefore, we move $\gamma(\sigma)$ to $\sigma' = \{\neg a, \neg\Psi, \neg\Phi\}$ and obtain a model $\mathcal{A}$ with the following components $\mathcal{U}$ and O.



The refinement operator takes as input the abstract Φ-model $\mathcal{A}$ that the model checker returns and replaces $\mathcal{A}$ by another abstract Φ-model $\mathcal{A}_{i+1}$ where again the under- and overapproximations are improved. $\mathcal{A}_{i+1}$ is obtained by a sequence of refinement steps that successively treat any of the $\forall$subformulas of Φ. As usual, the subformulas should be considered in an order consistent with the subformula relation. Let us assume that $\mathcal{A}$ is the current abstract Φ-model to be refined according to a $\forall$subformula Ψ of Φ. If the over- and underapproximations for Ψ agree in $\mathcal{A}$, i.e., if $Sat^+_{\mathcal{A}}(\Psi) = Sat^-_{\mathcal{A}}(\Psi)$, then

[8] Any movement of blocks might change (improve) the current abstract Φ-model $\mathcal{A}$. Thus, any FOR-loop of Model_Check($\mathcal{A}$,Φ) is started with a model that might be even better than the original model $\mathcal{A}$.

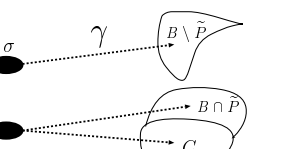
we may conclude that $Sat^+_{\mathcal{A}}(\Psi) = Sat_{\mathcal{M}}(\Psi) = Sat^-_{\mathcal{A}}(\Psi)$. As the precise satisfaction set for Ψ is known there is no need for further treatment of Ψ. From this point on, Ψ (and its subformulas) can be ignored. Thus, we just replace Ψ by the atomic proposition a_Ψ. E.g. if $\Phi = \forall\mathsf{X}(\forall\Diamond a \wedge b)$ and $\Psi = \forall\Diamond a$ then we replace Φ by $\forall\mathsf{X}(a_\Psi \wedge b)$. Otherwise, i.e., if $Sat^-_{\mathcal{A}}(\Psi)$ is a proper subset of $Sat^+_{\mathcal{A}}(\Psi)$, we calculate $\mathcal{A}' = \text{Refine}(\mathcal{A}, \Psi)$ by:

CASE Ψ **IS** $\forall\mathsf{X}\Psi_0$ **THEN** return Refine_Forall_Next($\mathcal{A}$,Ψ);
$\forall\Psi_1\mathsf{U}\Psi_2$ **THEN** return Refine_Forall_Until($\mathcal{A}$,Ψ);
$\forall\Psi_1\widetilde{\mathsf{U}}\Psi_2$ **THEN** return Refine_Forall_WeakUntil($\mathcal{A}$,Ψ);
ENDCASE

First, we briefly sketch the next step operator. Let $\Psi = \forall X\Psi_0$. Clearly, all concrete states s where $Post(s) \subseteq Sat^-_{\mathcal{A}'}(\Psi_0)$ satisfy Ψ. Similarly, only those concrete states s where $Post(s) \subseteq Sat^+_{\mathcal{A}}(\Psi_0)$ are candidates to fulfill Ψ. Thus, we may replace $\mathcal{A}$ by the abstract Φ-model $\mathcal{A}'$ with

$$Sat^+_{\mathcal{A}'}(\Psi) = \widetilde{Pre}\left(Sat^+_{\mathcal{A}}(\Psi_0)\right), \quad Sat^-_{\mathcal{A}'}(\Psi) = \widetilde{Pre}\left(Sat^-_{\mathcal{A}}(\Psi_0)\right)$$

while the over- and underapproximations for $Sat_{\mathcal{M}}(\Psi')$ (where $\Psi' \neq \Psi$) do not change. This change of $\mathcal{A}$ corresponds to a *splitting* of the blocks $B \in \Pi_{\mathcal{A}}$ into the subblocks $B \cap \tilde{P}$ and $B \setminus \tilde{P}$ where $\tilde{P} = \widetilde{Pre}(\ldots)$. The splitting is performed twice: first for $\tilde{P} = \widetilde{Pre}(Sat^-_{\mathcal{A}}(\Psi_0))$ which yields an "intermediate" abstract Φ-model $\mathcal{A}''$; second we split the blocks in $\mathcal{A}''$ with the set $\tilde{P} = \widetilde{Pre}(Sat^+_{\mathcal{A}}(\Psi_0))$ In our algorithm the splitting operation does not create new abstract states. Let $B = \gamma(\sigma)$ where Ψ, $\neg\Psi \notin \sigma$ and $\tilde{P} = \widetilde{Pre}(Sat^-_{\mathcal{A}}(\Psi))$. We realize the splitting of B by *moving* the subblock $B \cap \tilde{P}$ from the abstract state σ to the abstract state $\lceil\sigma \cup \{\Psi\}\rceil$. Similarly, we treat the splitting according to the overapproximations.



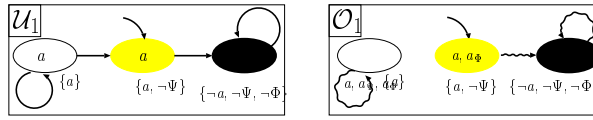
The procedure for the handling of until and weak until is based on similar ideas. For $\Psi = \forall\Psi_1\mathsf{U}\Psi_2$ we switch from $\mathcal{A}$ to the abstract Φ-model $\mathcal{A}'$ where

$$Sat^-_{\mathcal{A}'}(\Psi) = Sat^-_{\mathcal{A}}(\Psi_2) \cup \left(Sat^-_{\mathcal{A}}(\Psi_1) \cap \widetilde{Pre}\left(Sat^-_{\mathcal{A}}(\Psi)\right)\right).$$

Then, we check whether the least fixed point computation of $Sat_{\mathcal{M}}(\Psi)$ via the underapproximations is finished. For this, we just need the information whether $\mathcal{A}' = \mathcal{A}$, i.e., whether at least one of the blocks has been split into proper subblocks (i.e., γ changed). If so and if Ψ_1 and Ψ_2 are propositional formulas (for which the precise satisfaction sets are already computed) then we may conclude that $Sat^-_{\mathcal{A}}(\Psi)$ agrees with $Sat_{\mathcal{M}}(\Psi)$. In this case, we switch from $\mathcal{A}$ to $\mathcal{A}''$ where $Sat^+_{\mathcal{A}''}(\Psi) = Sat^-_{\mathcal{A}}(\Psi)$ and replace Ψ by the atomic proposition a_Ψ. If the computation of $Sat_{\mathcal{M}}(\Psi)$ is not yet finished then

we improve the upper bound. These ideas are presented in Algorithm 4. The treatment of weak until in the refinement step is almost the same as for until; the only difference being – as we have to calculate a greatest fixed point via overapproximations – that the roles of under- and overapproximations have to be exchanged.

Example : Let us revisit the running example. Let $\mathcal{A} = (\alpha, \gamma, \mathcal{U}, O)$ be the current abstract Φ-model the model checker has returned in the first iteration (see the picture above). Refinement starts with $\Psi = \forall\Box a$. We get $\widetilde{Pre}(Sat^+_{\mathcal{A}}(\Psi)) = \widetilde{Pre}(\gamma(\{a\})) = \gamma(\{a\}) \setminus \{s_0\}$. Thus, the grey concrete initial state s_0 is moved to $\{a, \neg\Psi\}$. All other refinement steps leave the model unchanged. Refine$(\mathcal{A}, \Phi)$ returns the model with components $\mathcal{U}_1, O_1$ as shown below.



In the following model checking phase, $NewSat(\Psi) = NewSat(\Phi) = NewSat(\neg\Psi) = \emptyset$. $NewSat(\neg\Phi)$ consists of the grey abstract state $\sigma = \{a, \neg\Psi\}$. Therefore, we move $\gamma(\sigma) = \{s_0\}$ to the abstract state $\sigma' = \{a, \neg\Psi, \neg\Phi\}$. We obtain an abstract Φ-model $\mathcal{A}_2$ where the abstract interpretation of the concrete initial state s_0 is $\alpha_2(s_0) = \sigma'$. As σ' contains $\neg\Phi$, the condition $\mathcal{A}_2 \models \neg\Phi$ in the repeat-loop of Algorithm 2 holds (see Def. 3). Hence, Algorithm 2 terminates with the correct answer "no". □

Remark : There is no need for an explicit treatment of the *boolean connectives* $\vee$ and $\wedge$ in the model checking or refinement step. For instance, if $\Psi = \Psi_1 \vee \Psi_2$ is a subformula of Φ then improving the approximations for the sets $Sat_{\mathcal{M}}(\Psi_1)$ automatically yields an improvement for the underapproximation for $Sat_{\mathcal{M}}(\Psi)$. "Moving" a block B from an abstract state σ to the abstract state $\sigma' = \lceil \sigma \cup \{\Psi_1\} \rceil$ has the side effect that B is added to both $Sat^-_{\mathcal{A}}(\Psi_1)$ and $Sat^-_{\mathcal{A}}(\Psi)$. This is due to the axioms, we require for the elements in S_Φ. The corresponding observation holds for the overapproximations $Sat^+_{\mathcal{A}}(\cdot)$. □
Remark: The *atomic propositions* a_Ψ play a crucial role in both the model checking and the refinement procedure. The labelings $L_{\mathcal{U}}$ and L_O cover the information that might got lost due to the transition relations $\rightarrow_\alpha$ and $\rightsquigarrow_\alpha$. In the refinement phase, they are necessary to detect when the computation of a least or greatest fixed point is finished. □

Theorem 1. [Partial Correctness] *If Algorithm 2 terminates with the answer "yes" then* $\mathcal{M} \models \Phi$. *If Algorithm 2 terminates with the answer "no" then* $\mathcal{M} \not\models \Phi$. □

Because of the similarities with stable partitioning algorithms for calculating the (bi-) simulation equivalence classes [37,7,32,24] it is not surprising that our algorithm terminates provided that the (bi-)simulation quotient space of $\mathcal{M}$ is finite.

Theorem 2. [Termination] *If the concrete model* $\mathcal{M}$ *has a finite simulation or bisimulation quotient then Algorithm 2 terminates.*

Algorithm 4 Refine_Forall_Until($\mathcal{A}, \Psi$) where $\Psi = \forall \Psi_1 \mathrm{U} \Psi_2$.

Let γ be the concretization function of $\mathcal{A}$.
$\widetilde{P} := \widetilde{Pre}(Sat^-_{\mathcal{A}}(\Psi))$; changed:= false; (* improve the underapproximation for $Sat_{\mathcal{M}}(\Psi)$ *)
FOR ALL $\sigma \in S_\Phi$ where $\Psi \notin \sigma$, $\Psi_1 \in \sigma$ and $\gamma(\sigma) \cap \widetilde{P} \neq \emptyset$ **DO**
  $\gamma(\lceil\, \sigma \cup \{\Psi\}\, \rceil) := (\gamma(\sigma) \cap \widetilde{P}) \cup \gamma(\lceil\, \sigma \cup \{\Psi\}\, \rceil)$; $\gamma(\sigma) := \gamma(\sigma) \setminus \widetilde{P}$; changed:= true;
OD;
IF not changed and Ψ_1, Ψ_2 are propositional formulas **THEN**
  (* the least fixed point computation is finished; put $Sat^+_{\mathcal{A}}(\Psi) := Sat^-_{\mathcal{A}}(\Psi)$ *)
  replace Ψ by the atomic proposition a_Ψ;
  FOR ALL $\sigma \in S_\Phi$ with $\Psi \notin \sigma$ and $\neg\Psi \notin \sigma$ **DO**
    $\gamma(\lceil\, \sigma \cup \{\neg\Psi\}\, \rceil) := \gamma(\lceil\, \sigma \cup \{\neg\Psi\}\, \rceil) \cup \gamma(\sigma)$; $\gamma(\sigma) := \emptyset$;
  OD
ELSE
  $\widetilde{P} := \widetilde{Pre}(Sat^+_{\mathcal{A}}(\Psi))$; (* improve the overapproximation for $Sat_{\mathcal{M}}(\Psi)$ *)
  FOR ALL $\sigma \in S_\Phi$ where $\neg\Psi \notin \sigma$, $\neg\Psi_1 \notin \sigma$, $\neg\Psi_2 \in \sigma$ and $\gamma(\sigma) \setminus \widetilde{P} \neq \emptyset$ **DO**
    $\gamma(\lceil\, \sigma \cup \{\neg\Psi\}\, \rceil) := \gamma(\lceil\, \sigma \cup \{\neg\Psi\}\, \rceil) \cup (\gamma(\sigma) \setminus \widetilde{P})$; $\gamma(\sigma) := \gamma(\sigma) \cap \widetilde{P}$
  OD
FI
Return the abstract Φ-model with concretization function γ.

Full *CTL*: Our algorithm can be extended to treat full *CTL*. The major difference is the handling of existential quantification which requires the use of the transition relation $\rightsquigarrow_\alpha$ when calculating the underapproximations while for the overapproximations we use the transition relation $\rightarrow_\alpha$. Given an abstract Φ-model $\mathcal{A} = (\alpha, \gamma, \mathcal{U}, O)$, we work (as before) with two satisfaction relations $\models_{\mathcal{U}}$ and $\models_O$. E.g. $\sigma \models_{\mathcal{U}} \exists\varphi$ iff there exists a path π in O (i.e., a path built from transitions w.r.t. $\rightsquigarrow_\alpha$) that starts in σ and $\pi \models_{\mathcal{U}} \varphi$. In the refinement phase, we use the predecessor predicate $Pre(\cdot)$ rather than $\widetilde{Pre}(\cdot)$. For instance, to improve the underapproximation for a subformula $\Psi = \exists\Diamond\Psi_0$ we split any block $B = \gamma(\sigma)$ (where $\Psi \notin \sigma$) into $B \cap Pre(Sat^-_{\mathcal{A}}(\Psi))$ and $B \setminus Pre(Sat^-_{\mathcal{A}}(\Psi))$. Again, the partial correctness relies on the results of [19]. Termination can be guaranteed for any concrete system with a finite bisimulation quotient.

5 Concluding Remarks

We have presented a general abstraction refinement algorithm for model checking large or infinite transition systems against $\forall$*CTL* (or *CTL*) formulas. Partial correctness can be established for any concrete transition system $\mathcal{M}$ which (if it is finite) could be represented by a BDD or might be a program with variables of an infinite type. Termination can be guaranteed for all concrete systems with a finite bisimulation quotient. For $\forall$*CTL*, our algorithm terminates also if only the simulation quotient is finite.

Clearly, the feasability of our algorithm crucially depends on the representation of the concrete system for which we have to extract the $\widetilde{Pre}$-information. In principle, our methodology can be combined with several fully or semi-automatic techniques that provide an abstract model. For large but finite concrete systems, we suggest a symbolic representation of the transition relation in $\mathcal{M}$ and the blocks in $\Pi_{\mathcal{A}}$ with BDDs. We

just started to implement our method with a BDD representation for the concrete model $\mathcal{M}$ but, unfortunately, cannot yet report on experimental results. It might be interesting to see whether (and how) the abstraction techniques for BDDs (e.g. [15,31]) can be combined with our algorithm. To reason about infinite systems, the fully automatic approach of [34] seems to fit nicely in our framework as it works with a $\widetilde{Pre}$-operator similar to the one that we use.

One of the further directions we intend to investigate is the study of real time systems or other types of transition systems that are known to have finite (bi-)simulation quotients [2,24]. In principle, our technique should be applicable to establish qualitative properties of timed automata (expressed in *CTL*). It would be interesting to see whether our method can be modified to handle quantitative properties (e.g. specified in *TCTL*).

References

1. P. Abdulla, A. Annichini, S. Bensalem, A. Boujjani, P. Habermehl, Y. Lakhnech. Verification of infinite state systems by combining abstraction and reachability analysis. In Proc. CAV'99, LNCS 1633, 1999.
2. R. Alur, D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. A. Aziz, T. R. Shiple, V. Singhal, A. L. Sangiovanni-Vincentelli. Formula-dependent equivalence for compositional CTL model checking. Proc. CAV'94, LNCS 818, pp. 324–337, 1994.
4. A. Biere, A. Cimatti, E. Clarke, M. Fujita, Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Design Automation Conference*, pp.317–320, 1999.
5. M. Browne, E. Clarke, O. Grumberg. Characterizing finite Kripke structures in Propositional Temporal Logic. *Theoretical Computer Science*, 59(1-2):115–131, July 1988.
6. J. Burch, E. Clarke, K. McMillan, D. Dill, L. Hwang. Symbolic model checking 10^{20} states and beyond. *Information and Computation*, 1992.
7. A. Bouajjani, Jean-Claude Fernandez, N. Halbwachs. Minimal model generation. Proc. CAV'90, LNCS 531, pp. 197–203, 1990.
8. D. Bustan, O. Grumberg. Simulation based minimization. Computing Science Reports CS-2000-04, Computer Science Department, Technion, Haifa 32000, Israel, 2000.
9. S. Bensalem, Y. Lakhnech, S. Owre. Computing abstractions of infinite state systems compositionally and automatically. LNCS 1427, Proc. CAV'98, pp. 319–331, 1998.
10. F. Balarin, A. Sangiovanni-Vincentelli. An iterative approach to language containment. Proc. CAV'93, LNCS 697, pp. 29-40, 1993.
11. P. Cousot, R. Cousot. Abstract interpretation a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Proc. POPL'77, pp. 238-252, 1977.
12. E. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith. Counterexample–guided abstraction refinement. LNCS 1855, Proc. CAV'00, pp. 154-169, 2000.
13. E. Clarke, O. Grumberg, D. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, September 1994.
14. E. Clarke, O. Grumberg, D. Peled. *Model Checking*. MIT Press, 2000.
15. E. Clarke, S. Jha, Y. Lu, D. Wang. Abstract BDDs: a technique for using abstraction in model checking. In Proc. Correct Hardware Design and Verification Methods, LNCS 1703, pp. 172–186, 1999.
16. D. Dams. *Abstract Interpretation and Partition Refinement for Model Checking*. PhD thesis, Technische Universiteit Einhoven, 1996.

17. J. Dingel, T. Filkorn. Model checking for infinite state systems using data abstraction, assumption commitment style reasoning and theorem proving. In Proc. CAV'95, LNCS 939, pp. 54–69, 1995.
18. D. Dams, R. Gerth, O. Grumberg. Generation of reduced models for checking fragments of CTL. In Proc. CAV'93, LNCS 697, pp. 479–490, 1993.
19. D. Dams, R. Gerth, O. Grumberg. Abstract interpretation of reactive systems. *ACM Transactions on Programming Languages and Systems*, 19(2):253–291, March 1997.
20. E. A. Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pp. 995–1072. Elsevier Science Publishers, Amsterdam, The Netherlands, 1990.
21. O. Grumberg, D. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, 1994.
22. P. Godefroid. Partial order methods for the verification of concurrent systems: An approach to the state explosion problem (Ph.D.Thesis, University of Liege) LNCS 1032, 1996.
23. S. Graf, H. Saidi. Construction of abstract state graphs with PVS. In Proc. CAV'97, LNCS 1254, pp 72–83, 1997.
24. M. Henzinger, T. Henzinger, P. Kopke. Computing simulations on finite and infinite graphs. In Proc. FOCS'95, pp. 453–462, IEEE Computer Society Press. 1995.
25. P. Kelb, D. Dams, R. Gerth. Efficient symbolic model checking of the full μ-calculus using compositional abstractions. Computing Science Reports 95/31, Eindhoven University of Technology, 1995.
26. R. Kurshan. *Computer-aided Verification of Coordinating Processes: The Automata-Theoretic Approach*. Princeton University Press, 1994.
27. C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6(1):11–44, January 1995.
28. J. Lind-Nielsen, H. Andersen. Stepwise CTL model checking of State/Event systems. In Proc. CAV'99, LNCS 1633, pp. 316–327, 1999.
29. D. Long. *Model Checking, Abstraction and Compositional Verification*. PhD thesis, Carnegie Mellon University, 1993.
30. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proceedings of the Twelfth Annual ACM Symposium on Principles of Programming Languages*, pages 97–107, New York, January 1985. ACM.
31. W. Lee, A. Pardo, J.-Y. Jang, G. Hachtel, F. Somenzi. Tearing based automatic abstraction for ctl model checking. In *Proc. ICCAD'96*, pp. 76–81, 1996.
32. D. Lee, M. Yannakakis. Online minimization of transition systems. In Proc. STOC'92, pp. 264–274, 1992. ACM Press.
33. K. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.
34. K. Namjoshi, R. Kurshan. Syntactic program transformation for automatic abstraction. In Proc. CAV'2000, LNCS 1855, pp. 435–449, 2000.
35. D. Peled. All from one, one from all: on model checking using representatives. In Proc. CAV'93, LNCS 697, pp. 409-423, 1993.
36. A. Pardo, G. Hachtel. Automatic abstraction techniques for propositional μ-calculus model checking. In Proc. CAV'97, LNCS 1254, pp. 12-23, 1997.
37. R. Paige, R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.
38. H. Saidi, N. Shankar. Abstract and model check while you prove. In Proc. CAV'99, LNCS 1633, pp 443-454, 1999.
39. H. B. Sipma, T. E. Uribe, Z Manna. Deductive Model Checking. In Proc. CAV'96, LNCS 1102, pp. 208–219, 1996
40. A. Valmari. State of the art report: Stubborn sets. *Petri-Net Newsletters*, 46:6–14, 1994.