Lazy Approximation for Dense Real-Time Systems*

(Extended Abstract)

Draft of a paper to be presented at FORMATS/FTRTFT'04, Grenoble 2004

Maria Sorea**

Universität Ulm, Abteilung Künstliche Intelligenz, Germany
sorea@informatik.uni-ulm.de

Abstract. We propose an effective and complete method for verifying safety and liveness properties of timed systems, which is based on predicate abstraction for computing finite abstractions of timed automata and TCTL formulas, finite-state CTL model checking, and successive refinement of finite-state abstractions. Starting with some coarse abstraction of the given timed automaton and the TCTL formula we define a finite sequence of refined abstractions that converges to the region graph of the real-time system. In each step, new abstraction predicates are selected nondeterministically from a finite, predetermined basis. Symbolic counterexamples from failed model-checking attempts are used to heuristically choose a small set of new abstraction predicates for incrementally refining the current abstraction. Without sacrificing completeness, this algorithm usually does not require computing the complete region graph to decide model-checking problems. Abstraction refinement terminates quickly, as a multitude of spurious counterexamples is eliminated in every refinement step through the use of symbolic counterexamples for TCTL.

1 Introduction

Timed Automata [2] are state-transition graphs augmented with a finite set of real-valued clocks. The clocks proceed at a uniform rate and constrain the times at which transitions may occur. Given a timed automaton and a property expressed in a (timed) temporal logic, model checking answers the question whether the timed automaton satisfies the given formula. The fundamental graph-theoretic model checking algorithm by Alur, Courcoubetis and Dill [1] constructs a finite quotient, the so-called *region graph*, of the infinite state graph corresponding to the timed automaton. Algorithms directly based on the explicit construction of such a partition of states are inefficient since the number of equivalence classes of states of the region graph grows exponentially with the largest time constant and the number of clocks that are used to specify timing constraints.

In [18,17] we propose a novel method for verifying safety and liveness properties of timed systems based on predicate abstraction [13] for timed automata, finite-state model checking, and counterexample-guided abstraction refinement. We define a set of

* This work was supported by SRI International, by NSF grants CCR-ITR-0326540 and CCR-ITR-0325808.

** Currently at Stanford University. This research has been mainly conducted while the author was visiting SRI International.

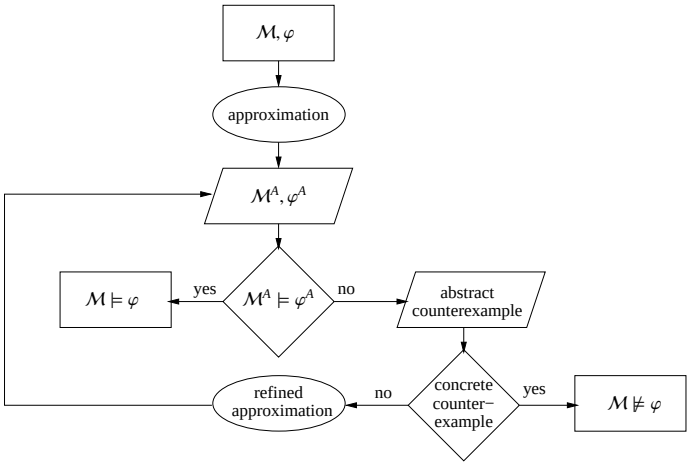

Fig. 1. Lazy approximation.

so-called *basis predicates*, which are expressive enough for distinguishing between any two clock regions. This set of predicates determines a strongly preserving abstraction in the sense that a timed automaton validates a μ-calculus formula iff the corresponding finite abstraction validates this formula. The control structure of the timed automaton is preserved in the abstract system. The abstracted systems no longer refer to the real-time nature of computations, and finite-state model checkers can be used to establish safety and liveness properties in the abstracted system.

In many cases it is not necessary to compute the exact abstraction using the entire basis of predicates, since a coarser approximation of this is sufficient for proving or refuting the desired property. Since we consider safety and liveness properties we maintain both under- and over-approximations of the given timed system. These approximations are computed via an iterative abstraction-refinement process that starts with some coarse approximations of the timed system and computes a sequence of approximations until the one necessary for proving or refuting the property is obtained. In each refinement step new abstraction predicates are selected from the finite set of basis predicates and new, more detailed approximations are computed. Hereby, the choice of predicates is guided by counterexamples from failing model-checking attempts. We call this method *lazy approximation*. This process of abstracting and refining approximations is illustrated in Figure 1. When using the entire basis of predicates for computing the approximations, the under- and over-approximation are identical, yielding therefore a strongly property preserving abstraction of the timed system. Since the sequence of approximations converges toward the region graph of the real-time systems, the method of lazy approximation is complete [18,17]. The main advantage of this approach is that finite time-abstractions are computed lazily. This results in substantial savings in computation whenever coarse abstractions are sufficient to prove the property at hand. Standard benchmark examples for timed automata such as train gate controller and a

version of Fischer's mutual exclusion protocol can be proved using only a few abstraction predicates.

In this paper we extend our previous results [18,17] in several directions. First, we consider TCTL [1] for expressing qualitative and quantitative properties of timed systems, instead of the untimed logic considered in [18,17]. We define an abstraction function for TCTL that maps a TCTL formula to a CTL formula, together with the inverse operation of concretization. The predicates necessary for the abstraction are extracted from the timed-bounded modalities of the TCTL formula. For extracting the predicates we introduce for every timed-bounded modality of a given TCTL formula φ a new clock variable z_i. Now, the set of abstraction predicates Ψ_φ with respect to φ consists of all the formulas $z_i \sim c$ with free variables z_i, where $\sim c$ denotes the timed bound of the modalities occurring in φ. For examples, the abstraction predicates corresponding to the TCTL formula $\varphi = \mathbf{EG}_{<2}\, p \ \wedge\ \mathbf{A}[q\, \mathbf{U}_{\leq 4}\, r]$, with p, q, r atomic propositions, are given as $\psi_1 \equiv z_1 < 2$, $\psi_2 \equiv z_2 \leq 4$. The resulting abstract CTL formula is now obtained using these predicates as $\varphi^A = \mathbf{EG}(p \wedge \psi_1) \wedge \mathbf{A}[q\, \mathbf{U}\, (r \wedge \psi_2)]$.

Second, in contrast to the previous version of our algorithm [18,17], where refined approximations are recomputed from scratch, we compute abstraction refinements in an incremental fashion, following the approach outlined by Das and Dill [8] for the untimed case.

Third, we introduce a symbolic form of counterexamples for the full TCTL logic, as sequences over sets of states. These symbolic structures are timed extensions of the symbolic counterexamples for (untimed) CTL [20]. We use symbolic counterexamples in the abstraction-refinement algorithm as a heuristic for selecting new abstraction predicates from the given set of basis predicates. Symbolic counterexamples make the refinement process converge more quickly compared to the use of linear counterexamples, as a multitude of spurious counterexamples are discarded in every refinement step. Moreover, since we define symbolic counterexamples for the full TCTL, the method of lazy approximation is applicable for full TCTL, and not only for a fragment of universal formulas as it is the case when using linear counterexamples.

The main contributions of our paper are

1. A definition of abstraction functions for timed automata and TCTL based on predicate abstraction.
2. A definition of symbolic counterexamples for full TCTL.
3. An incremental abstraction refinement algorithm for computing finite approximations of timed automata and TCTL formulas.
4. A proof for termination, soundness and completeness of the abstraction refinement algorithm.

Related Work. The abstract interpretation framework [7] has been used earlier in the context of real-time systems for formalizing approximations of safety properties [23,11,9]. In contrast, the techniques proposed in [18,17] and extended in this paper, also allow for verifying liveness. Whereas verification techniques for infinite-state systems based on predicate abstraction [13,5,19,14] are usually used in an incomplete way for proving safety properties, our verification method for timed systems is even complete for liveness properties.

Counterexample-guided refinement has been studied by many researchers, and recent work includes [6,8,15,14]. In contrast to these approaches, we use counterexamples only as a heuristic for selecting good pivot predicates from a fixed, predetermined pool of abstraction predicates to speed convergence of the approximation processes.

Dill and Wong-Toi [11] also use an iteration of both over- and under-approximations of the reachable state set of timed automata, but their techniques are limited to proving invariants. Daws and Tripakis [9] propose several abstractions that reduce the state space of a timed system, while preserving reachability properties. Tripakis and Yovine [22] show how to abstract dense real time to obtain time-abstracting, finite bisimulations. Behrmann, Bouyer, Larsen and Pelánek [4] propose zone based abstractions with respect to the minimal and maximal constants to which clocks are compared, obtaining a sound and complete verification method for reachability. Whenever it suffices to compute rather coarse abstractions, we expect to obtain much smaller transition systems by means of lazy approximation. Alur, Itai, Kurshan, and Yannakakis [3] present a technique based on over-approximations: the method consists in attempting to prove a property on an abstract system, where some clocks are ignored; if this attempt fails, clocks are reintroduced progressively until either the property is proved on the abstract system, or all the clocks have been reintroduced. The method still requires exact computation of the region graph for each abstracted system.

Henzinger, Jhala, Majumdar, and Sutre [14] present an abstraction-refinement algorithm for model checking safety properties that integrates the construction of the abstract model with the verification process. The abstract model is constructed on demand during verification, by refining only parts of the current abstract model. However, this method allows for checking only reachability properties, whereas our approach can be used to verify or refute any kind of TCTL properties.

All the above-described approaches use linear counterexamples during the refinement process. In contrast, symbolic counterexamples make the refinement process converge more quickly compared to the use of linear counterexamples, since several spurious counterexamples are discarded in one refinement step.

Organization. The remainder of this paper is organized as follows. Section 2 reviews the basic notions of timed automata and TCTL. Finite over- and under-approximations of timed automata are defined in the first part of Section 3, while the second part contains definitions of abstraction and concretization functions for TCTL. Symbolic counterexamples for TCTL as sequences over sets of states are introduced in Section 4. In Section 5, we define the iterative abstraction refinement algorithm and show termination and completeness thereof. Finally, Section 6 contains some concluding remarks. For lack of space we omit some of the proofs, but detailed proofs can be found in [18,21].

2 Preliminaries

Given a set of clocks C, the set of *timing (or clock) constraints Constr* comprises $\mathtt{tt}$, $x \bowtie d$, and $x - y \bowtie d$, where $x, y \in C$, $d \in \mathbb{N}$, $\bowtie \in \{\leq, <, =, >, \geq\}$. The set *Inv* is the subset of *Constr*, where $\bowtie$ is chosen from $\{\leq, <\}$. For a positive integer γ, *Constr*(γ) is the finite subset of all clock constraints $x \sim \gamma$, $x - y \sim \gamma$, where $x, y \in C$.

A *timed automaton* [2] is a tuple $\mathcal{S} = \langle L, P, C, E, L_0, I \rangle$, where

- L is a nonempty finite set of locations.
- $P : L \to \mathcal{P}(\mathbf{AP})$ maps each location to a set of propositional symbols $\mathbf{AP}$.
- C is a finite set of clocks.
- $E \subseteq L \times \mathcal{P}(Constr) \times \mathcal{P}(C) \times L$ is a transition relation; we write $l \xrightarrow{g,r} l'$ for $\langle l, g, r, l' \rangle \in E$.
- $L_0 \subseteq L$ is the set of initial locations.
- $I : L \to \mathcal{P}(Inv)$ assigns a set of downward closed clock constraints to each location l; the elements of $I(l)$ are the *invariants* for location l.

A function $\nu : C \to \mathbb{R}_{\geq 0}$ is a *clock valuation*, and the set of clock valuations is collected in $\mathcal{V}_C$. $\mathcal{V}_0$ denotes the set of initial clock valuations that assigns to every clock the value 0. The clock valuation $(\nu + \delta)$ is obtained by adding δ to the value of each clock in ν. For $X \subseteq C$, $\nu[X := 0]$ denotes the clock valuation that updates every clock $x \in X$ to zero, and leaves all the other clock values unchanged. The value $g\nu$ of a clock constraint g with respect to the clock valuation ν is obtained by substituting the clocks x in g with the corresponding value $\nu(x)$. If $g\nu$ simplifies to the true value, ν satisfies g and we write $\nu \models g$. A set $\mathcal{X} \subseteq \mathcal{V}_C$ of clock valuations satisfies $g \in Constr$, written as $\mathcal{X} \models g$, if and only if $\nu \models g$ for all $\nu \in \mathcal{V}_C$. A pair $(l, \nu) \in L \times \mathcal{V}_C$ is called a *timed configuration*, if it satisfies the invariants $I(l)$; formally, $\nu \models I(l)$ iff $\nu \models g$ for every invariant $g \in I(l)$. A *clock region* [2] is a set $\mathcal{X} \subseteq \mathcal{V}_C$ of clock valuations, such that for all timing constraints $g \in Constr(\gamma)$ and for any two $\nu_1, \nu_2 \in \mathcal{X}$ it is the case that $\nu_1 \models g$ if and only if $\nu_2 \models g$. In this case we write $\nu_1 \cong \nu_2$.

A *timed step* is either a *delay step*, where time advances by some positive real-valued δ, or an instantaneous *state transition step*. For $\delta > 0$, we say that the timed configuration $(l, \nu + \delta)$ is obtained from (l, ν) by a *delay step* $(l, \nu) \overset{\delta}{\Rightarrow} (l, \nu + \delta)$, if the invariant constraint $\nu + \delta \models I(l)$ holds. A *state transition step* $(l, \nu) \overset{g,r}{\Rightarrow} (l', \nu')$ occurs if there exists a $l \xrightarrow{g,r} l' \in E$, and $\nu \models g$, $\nu' := \nu[r := 0]$, and $\nu' \models I(l')$.

The lazy approximation method, we present here, allows for verifying not only safety properties, but also liveness properties. Liveness in dense real time is complicated by the possible sequences of infinitesimally decreasing delay steps; they constitute a degenerated behavior of a system, a behavior that has to be disallowed. As in [18,17], we eliminate this undesired behavior by restricting the model of timed automata to delay steps that force a clock to step beyond integer bounds when all fractional clock values are not zero. We have shown [18,17] that such a restriction does not change the possible observations of the model with respect to μ-calculus formulas. The proof can easily be adapted to TCTL formulas.

A *restricted delay step* [18,17] is a delay step $(l, \nu) \overset{\delta}{\Rightarrow} (l, \nu + \delta)$ for all positive, real-valued δ, such that

$$\exists x \in C.\, \exists k \in \{0, \ldots, \gamma\}.\, \nu(x) = k \vee (\nu(x) < k \wedge \nu(x) + \delta \geq k).$$

In this paper we consider timed systems with restricted delay steps. The union of restricted delay and state transition steps defines the timed transition relation $\Rightarrow$ of a timed system $\mathcal{S}$.

The semantics of a timed system $\mathcal{S} = \langle L, L_0, C, I, P, E \rangle$ is given by associated with it a transition system $\mathcal{M} = \langle \mathbf{S}, \mathbf{S}_0, \mathbf{P}, \mathbf{N} \rangle$, where $\mathbf{S} = L \times \mathcal{V}_C$, $\mathbf{S}_0 = L_0 \times \mathcal{V}_0 \subseteq \mathbf{S}$ are

the initial states, $\mathbf{P} = P$, and $\mathbf{N}$ is the timed transition relation $\Rightarrow$ introduced above. For $(s, s') \in \mathbf{N}$, we also write $s' \in \mathbf{N}(s)$, and if $S \subseteq \mathbf{S}$, then $\mathbf{N}(S)$ is $\cup_{s \in S} \mathbf{N}(s)$. The converse transition relation $\widetilde{\mathbf{N}}$ is defined by $\widetilde{\mathbf{N}}(s, s') \iff \mathbf{N}(s', s)$. We assume that the transition relation $\mathbf{N}$ is total, that is, every state has a successor. A *path* π is a finite or infinite sequence of configurations $\pi = (s_0, s_1, \ldots)$ such that $s_{i+1} \in \mathbf{N}(s_i)$ for all $i \geq 0$. We sometimes denote a path by $s_0 \Rightarrow s_1 \Rightarrow \ldots$.

Given a set $S \subseteq \mathbf{S}$ and the transition relation $\mathbf{N}$, we define three *predicate transformers* from $2^{\mathbf{S}}$ to $2^{\mathbf{S}}$; $post(\mathbf{N})(S) = \mathbf{N}(S)$, $pre(\mathbf{N})(S) = \widetilde{\mathbf{N}}(S)$, and $\widetilde{pre}(\mathbf{N})(S) = \{s \in \mathbf{S} \mid \mathbf{N}(s) \subseteq S\}$. The *postcondition* function $post(\mathbf{N})(S)$ computes for a given set S of states, the set of states that can be reached in one step from some state in S. The *preimage* function $pre(\mathbf{N})(S)$ returns the set of states that can reach S in a single step. The *precondition* function $\widetilde{pre}(\mathbf{N})(S)$ returns the set of those states that have no successors outside of S.

The logic TCTL [1] is a dense real-time extension of CTL with time bounded modalities and is defined by the grammar ($p \in \mathbf{AP}$)

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{E}[\varphi_1 \, \mathbf{U}_{\sim c} \, \varphi_2] \mid \mathbf{A}[\varphi_1 \, \mathbf{U}_{\sim c} \, \varphi_2] \, .$$

The semantics of TCTL formulas is given in the usual way, with respect to a transition system. The notions of s-path and TCTL-structure are as in [1], and are reviewed in Appendix A.

3 Abstraction Functions

3.1 Abstracting Timed Systems

Definition 1 (Abstraction Predicates [18,17]). Given a set of clocks C, an *abstraction predicate* with respect to C is any formula with the set of free variables in C. Similarly to timing constraints, the value of an abstraction predicate ψ with respect to a clock valuation ν, where both free and bound variables are interpreted in the domain C, is denoted by the juxtaposition $\psi\nu$. Whenever $\psi\nu$ evaluates to true, we write $\nu \models \psi$.

A set of abstraction predicates $\Psi = \{\psi_0, \cdots, \psi_{n-1}\}$ determines an *abstraction function* α, which maps clock valuations ν to a *bitvector* b of length n, such that the i-th component of b is set if and only if ψ_i holds for ν. Here, we assume that bitvectors of length n are elements of the set B_n, which are functions of domain $\{0, \cdots, n-1\}$ and codomain $\{0, 1\}$. The inverse image of α, that is, the *concretization function* γ, maps a bitvector to the set of clock valuations that satisfy all ψ_i whenever the i-th component of the bitvector is set. Thus, a set of concrete states (l, ν) is transformed by the abstraction function α into the abstract state $\alpha(l, \nu)$, and an abstract state (l, b) is mapped by γ to a set of concrete states $\gamma(l, b)$.

Definition 2 (Abstraction/Concretization [18,17]). Let C be a set of clocks and $\mathcal{V}_C$ the corresponding set of clock valuations. Given a finite set of predicates $\Psi = \{\psi_0, \cdots, \psi_{n-1}\}$, the *abstraction function* $\alpha : L \times \mathcal{V}_C \to L \times B_n$ is defined by

$$\alpha(l, \nu)(i) := (l, \psi_i\nu)$$

and the *concretization function* $\gamma : L \times B_n \to L \times \mathcal{P}(\mathcal{V}_C)$ is defined by

$$\gamma(l, b) := \{(l, \nu) \in L \times \mathcal{V}_C \mid I(l) \ \wedge \ \bigwedge_{i=0}^{n-1} \psi_i \nu \equiv b(i)\}.$$

We also use the notations $\alpha(S) := \{\alpha(l, \nu) \mid (l, \nu) \in S\}$ and $\gamma(S^a) := \{\gamma(l, b) \mid (l, b) \in S^a\}$. Now, the abstraction/concretization pair (α, γ) forms a Galois connection.

Definition 3 (Feasible State). An abstract state (l, b) is *feasible* if and only if its concretization is not empty, that is, $\gamma(l, b) \neq \emptyset$.

A set of predicates is feasible, if the conjunction of the predicates is satisfiable.

Definition 4 (Over-/Under-approximation [18,17]). Given a (concrete) transition system $\mathcal{M} = \langle \mathbf{S}^c, \mathbf{S}^c_0, \mathbf{P}, \Rightarrow \rangle$, where $\mathbf{S}^c = L \times \mathcal{V}_C$, $\mathbf{S}^c_0 = L_0 \times \mathcal{V}_0$, and a set Ψ of abstraction predicates, we construct two (abstract) transition systems $\mathcal{M}^+_\Psi = \langle \mathbf{S}^a, \mathbf{S}^a_0, \mathbf{P}, \Rightarrow^+ \rangle$, and $\mathcal{M}^-_\Psi = \langle \mathbf{S}^a, \mathbf{S}^a_0, \mathbf{P}, \Rightarrow^- \rangle$.

- $\mathbf{S}^a := L \times B_n$
- $(l, b) \Rightarrow^+ (l', b')$ iff
  $\exists \nu, \nu' \in \mathcal{V}_C$ s.t. $(l, \nu) \in \gamma(l, b) \ \wedge \ (l', \nu') \in \gamma(l', b')$. $(l, \nu) \Rightarrow (l', \nu')$
- $(l, b) \Rightarrow^- (l', b')$ iff (l, b) feasible, and
  $\forall \nu \in \mathcal{V}_C$ s.t. $(l, \nu) \in \gamma(l, b)$. $\exists \nu' \in \mathcal{V}_C$ s.t. $(l', \nu') \in \gamma(l', b')$. $(l, \nu) \Rightarrow (l', \nu')$
- $\mathbf{S}^a_0 := \{(l_0, b_0) \mid l_0 \in L_0, \text{ and } b_0(i) = 1 \text{ iff } \nu_0 \models \psi_i\}$.

$\mathcal{M}^+_\Psi$ is called an *over-approximation*, and $\mathcal{M}^-_\Psi$ an *under-approximation* of $\mathcal{M}$.

Obviously, we have that $\Rightarrow^- \subseteq \Rightarrow^+$. If the set Ψ of abstraction predicates is understood from the context, we omit it in the notation for under-, and over-approximation, and we write $\mathcal{M}^-$, respectively $\mathcal{M}^+$.

For the transition relations $\Rightarrow^-$, $\Rightarrow^+$, and $\Rightarrow$ we define $\gamma(\Rightarrow^-)$, $\gamma(\Rightarrow^+)$, respectively $\alpha(\Rightarrow)$ as follows:

$$\begin{aligned}
\gamma(\Rightarrow^-) &:= \{((l, \nu), (l', \nu')) \in \mathbf{S}^c \mid \exists b, b'.\ (l, b) \Rightarrow^- (l', b') \ \wedge \ (l, \nu) \in \gamma(l, b) \ \wedge \\
&\qquad\qquad (l', \nu') \in \gamma(l', b')\} \\
\gamma(\Rightarrow^+) &:= \{((l, \nu), (l', \nu')) \in \mathbf{S}^c \mid \exists b, b'.\ (l, b) \Rightarrow^+ (l', b') \ \wedge \ (l, \nu) \in \gamma(l, b) \ \wedge \\
&\qquad\qquad (l', \nu') \in \gamma(l', b')\} \\
\alpha(\Rightarrow) &:= \{(\alpha(l, \nu), \alpha(l', \nu')) \mid (l, \nu) \Rightarrow (l', \nu')\}
\end{aligned}$$

The next statement follows directly from Definition 4.

Lemma 1 ([18,17]). For a (concrete) transition system $\mathcal{M}$ with the transition relation $\Rightarrow$ and the corresponding over- and under-approximations $\mathcal{M}^+_\Psi$, $\mathcal{M}^-_\Psi$ with respective transition relations $\Rightarrow^+$, and $\Rightarrow^-$: (1) $\gamma(\Rightarrow^-) \subseteq \Rightarrow \subseteq \gamma(\Rightarrow^+)$ and (2) $\Rightarrow^- \subseteq \alpha(\Rightarrow) \subseteq \Rightarrow^+$.

Definition 4 does not allow the incremental computation of over- and under-approximations. When adding new predicates to Ψ, new approximations have to be constructed from scratch starting from the initial transition system. We modify Definition 4 such that successive approximations can be computed incrementally from previously obtained approximations by adding new predicates from the basis.

We introduce the following notations. A bitvector of length k is denoted by $b[0 : k-1]$, and corresponds to the set $\Psi_k = \{\psi_0, \ldots, \psi_{k-1}\}$ of abstraction predicates. The abstraction and concretization functions determined by Ψ_k are denoted by α_k, γ_k, respectively. The finite over-approximation of $\mathcal{M}$ with respect to Ψ_k is denoted by $\mathcal{M}^+_{\Psi_k}$ and is the tuple $\langle \mathbf{S}^a_k, \mathbf{S}^a_{0_k}, \mathbf{P}, \Rightarrow^+_k \rangle$. Similarly, the finite under-approximation of $\mathcal{M}$ with respect to Ψ_k is denoted by $\mathcal{M}^-_{\Psi_k} = \langle \mathbf{S}^a_k, \mathbf{S}^a_{0_k}, \mathbf{P}, \Rightarrow^-_k \rangle$. Note that the mapping function $\mathbf{P}$ does not depend on the abstraction predicates, merely on the finite control structure L.

Definition 5 (Incremental Over-/Under-approximation). For a timed system $\mathcal{S}$, with corresponding transition system $\mathcal{M}$, and a TCTL formula φ, let Ψ be the corresponding basis of abstraction predicates, $\mathcal{M}^+_{\Psi_k} = \langle \mathbf{S}^a_k, \mathbf{S}^a_{0_k}, \mathbf{P}, \Rightarrow^+_k \rangle$ and $\mathcal{M}^-_{\Psi_k} = \langle \mathbf{S}^a_k, \mathbf{S}^a_{0_k}, \mathbf{P}, \Rightarrow^-_k \rangle$ the over-approximation and under-approximation of $\mathcal{M}$ obtained in step i with respect to a set $\Psi_k \subset \Psi$ of abstraction predicates, respectively. Let $\Psi_{k'}$ be the set of predicates obtained from the failed model-checking attempt in step i. The over-approximation $\mathcal{M}^+_{\Psi_m} = \langle \mathbf{S}^a_m, \mathbf{S}^a_{0_m}, \mathbf{P}, \Rightarrow^+_m \rangle$ respectively under-approximation $\mathcal{M}^-_{\Psi_m} = \langle \mathbf{S}^a_m, \mathbf{S}^a_{0_m}, \mathbf{P}, \Rightarrow^-_m \rangle$ obtained in step $i+1$ with respect to the set of predicates $\Psi_m = \Psi_k \cup \Psi_{k'}$, is derived from $\mathcal{M}^+_{\Psi_k}$, respectively $\mathcal{M}^-_{\Psi_k}$, as follows:

- $\mathbf{S}^a_m = \{(l, b[0 : m-1]) \mid (l, b[0 : k-1]) \in \mathbf{S}^a_k$ and
  $\forall i = k, \ldots, k+k'-1.\ b[i] = 1$ if $\Psi_k \cap \psi_i \neq \emptyset$, else $b[i] = 0\}$
- $(l, b[0 : m-1]) \Rightarrow^+_m (l', b'[0 : m-1])$ iff
  - $(l, b[0 : k-1]) \Rightarrow^+_k (l', b'[0 : k-1])$ and
  - $\exists\, \nu_m, \nu'_m \in \mathcal{V}_C$ s.t. $(l, \nu_m) \in \gamma_m(l, b[0 : m-1])$ and $(l', \nu'_m) \in \gamma_m(l, b'[0 : m-1])$ with $\nu_m = \nu_k \cap \{\nu \in \mathcal{V}_C \mid \Psi_{k'} \nu \equiv 1\}$ and $\nu'_m = \nu'_k \cap \{\nu' \in \mathcal{V}_C \mid \Psi_{k'} \nu \equiv 1\}$ such that $(l, \nu_m) \Rightarrow (l', \nu'_m)$.
- $(l, b[0 : m-1]) \Rightarrow^-_m (l', b'[0 : m-1])$ iff
  - $(l, b[0 : k-1]) \Rightarrow^-_k (l', b'[0 : k-1])$ and
  - $\forall\, \nu_m, \nu'_m \in \mathcal{V}_C$ s.t. $(l, \nu_m) \in \gamma_m(l, b[0 : m-1])$ and $(l', \nu'_m) \in \gamma_m(l', b'[0 : m-1])$ with $\nu_m = \nu_k \cap \{\nu \in \mathcal{V}_C \mid \Psi_{k'} \nu \equiv 1\}$ and $\nu'_m = \nu'_k \cap \{\nu' \in \mathcal{V}_C \mid \Psi_{k'} \nu \equiv 1\}$ such that $(l, \nu_m) \Rightarrow (l', \nu'_m)$.

The set $\mathbf{S}^a_m$ can also be defined as in Definition 4 as the product of L and B_m, where B_m is the set of all bitvectors of length m. However, the above definition is more restrictive, in the sense that a smaller set of abstract states than $L \times B_m$ is obtained, since infeasible states are discarded.

3.2 TCTL Abstraction

We define abstractions and concretizations functions for TCTL formulas based on a set of abstraction predicates Ψ_φ. The predicates are extracted from the timed-bounded modalities of the formulas. For every timed-bounded modality of a given formula φ we

introduce a new clock variable z_i. Now, the set of abstraction predicates Ψ_φ with respect to φ consists of all the formulas $z_i \sim c$ with free variables z_i, where $\sim c$ denotes the timed bound of the modalities occurring in φ. For example, the abstraction predicates corresponding to the formula $\varphi = \mathbf{EG}_{<2}\, p \wedge \mathbf{A}[q\, \mathbf{U}_{\leq 4}\, r]$, with p, q, r atomic propositions, are given as $\Psi_\varphi = \{z_1 < 2, z_2 \leq 4\}$.

Definition 6 (TCTL Abstraction Predicates). Given a TCTL formula φ. A TCTL *abstraction predicate* is a formula $z_i \sim c_i$, with z_i a free variable and $\sim c_i$ the time bound of the i-th modality in φ. The TCTL abstraction predicates corresponding to a formula φ are collected in the set Ψ_φ. If φ does not contain any timed-bounded modalities, then Ψ_φ is empty.

Definition 7 (TCTL Abstraction/Concretization). Given a TCTL formula φ, and a set Ψ_φ of abstraction predicates. Furthermore, let $\psi \equiv z \sim c$ be a predicate in Ψ_φ, corresponding to the bounded modality $\mathbf{U}_{\sim c}$. The *abstraction function* $\alpha_\varphi : TCTL \rightarrow CTL$ is defined inductively over the structure of φ.

$$
\begin{aligned}
&\alpha_\varphi(\mathtt{tt}) := \mathtt{tt} \qquad \alpha_\varphi(p) := p \qquad \alpha_\varphi(\neg\varphi) := \neg\alpha_\varphi(\varphi)\\
&\alpha_\varphi(\varphi_1 \wedge \varphi_2) := \alpha_\varphi(\varphi_1) \wedge \alpha_\varphi(\varphi_2)\\
&\alpha_\varphi(\mathbf{E}[\varphi_1\, \mathbf{U}_{\sim c}\, \varphi_2] := \mathbf{E}[\alpha_\varphi(\varphi_1)\, \mathbf{U}\, (\alpha_\varphi(\varphi_2) \wedge \psi)]\\
&\alpha_\varphi(\mathbf{A}[\varphi_1\, \mathbf{U}_{\sim c}\, \varphi_2] := \mathbf{A}[\alpha_\varphi(\varphi_1)\, \mathbf{U}\, (\alpha_\varphi(\varphi_2) \wedge \psi)]
\end{aligned}
$$

The *concretization function* $\gamma_\varphi : CTL \rightarrow TCTL$ maps a CTL formula to a TCTL formula, and is the inverse operation to α_φ, that is $\gamma_\varphi(\varphi) = {\alpha_\varphi}^{-1}(\varphi)$.

Now, given a timed automaton $\mathcal{S}$ with a set of clocks C, and a TCTL formula φ, we add the clocks z_i corresponding to the bounded modalities of φ to C, and define the abstraction predicates with respect to the new set of clocks. The initial values of the clocks z_i equal zero, and the largest constants, which these clocks are even compared to are given by the constants c_i, appearing in the bounded modalities of φ. Let Ψ be the set of abstraction predicates corresponding to $\mathcal{S}$ and φ. If φ does not contain any timed-bounded modalities, then Ψ consists only of the predicates with respect to the automaton clocks, as in Definition 1.

Definition 8 (Predicate Abstracted Semantics of CTL). Let φ^c be a TCTL formula, $\mathcal{M} = \langle \mathbf{S}^c, \mathbf{S}_0^c, \mathbf{P}, \Rightarrow \rangle$ a transition system, and Ψ a set of abstraction predicates. Consider, as given in Definition 4, the over-approximation $\mathcal{M}_\Psi^+ = \langle \mathbf{S}^a, \mathbf{S}_0^a, \mathbf{P}, \Rightarrow^+ \rangle$, and the under-approximation $\mathcal{M}_\Psi^- = \langle \mathbf{S}^a, \mathbf{S}_0^a, \mathbf{P}, \Rightarrow^- \rangle$ of $\mathcal{M}$. Furthermore, let $\varphi = \alpha_\varphi(\varphi^c)$ be the CTL formula obtained by abstracting φ^c using the predicates corresponding to the bounded modalities in φ. Then, the *predicate abstracted* semantics $[\![\varphi]\!]^{\mathcal{M}_\Psi^\sigma}$, where σ is either $+$ or $-$, of the CTL formula φ with respect to the finite-state transition systems $\mathcal{M}_\Psi^\sigma$ is defined in a mutually inductive way. The notation $\bar{\sigma}$ is used to toggle the sign σ.

$$
\begin{aligned}
[\![\mathtt{tt}]\!]^{\mathcal{M}_\Psi^\sigma} &:= \mathbf{S}^a\\
[\![p]\!]^{\mathcal{M}_\Psi^\sigma} &:= \{(l, b) \in \mathbf{S}^a \mid p \in \mathbf{P}(l)\}\\
[\![\neg\varphi]\!]^{\mathcal{M}_\Psi^\sigma} &:= \mathbf{S}^a \setminus [\![\varphi]\!]^{\mathcal{M}^{\bar{\sigma}}}
\end{aligned}
$$

$$[\![\varphi_1 \vee \varphi_2]\!]^{\mathcal{M}^\sigma_\Psi} := [\![\varphi_1]\!]^{\mathcal{M}^\sigma_\Psi} \cup [\![\varphi_2]\!]^{\mathcal{M}^\sigma_\Psi}$$

$$[\![\mathbf{E}[\varphi_1 \,\mathbf{U}\, \varphi_2]]\!]^{\mathcal{M}^\sigma_\Psi} := \{s \in \mathbf{S}^a \mid \text{for some path } \pi = (s_0 \Rightarrow^\sigma s_1 \Rightarrow^\sigma \ldots) \text{ with } s_0 = s,$$
$$\text{for some } i \geq 0,\ s_i \in [\![\varphi_2]\!]^{\mathcal{M}^\sigma_\Psi} \text{ and } s_j \in [\![\varphi_1]\!]^{\mathcal{M}^\sigma_\Psi} \text{ for } 0 \leq j < i\}$$

$$[\![\mathbf{A}[\varphi_1 \,\mathbf{U}\, \varphi_2]]\!]^{\mathcal{M}^\sigma_\Psi} := \{s \in \mathbf{S}^a \mid \text{for every path } \pi = (s_0 \Rightarrow^\sigma s_1 \Rightarrow^\sigma \ldots) \text{ with } s_0 = s,$$
$$\text{for some } i \geq 0,\ s_i \in [\![\varphi_2]\!]^{\mathcal{M}^\sigma_\Psi} \text{ and } s_j \in [\![\varphi_1]\!]^{\mathcal{M}^\sigma_\Psi} \text{ for } 0 \leq j < i\}$$

We also write $\mathcal{M}^\sigma, (l, b) \models^a \varphi$, to denote that $(l, b) \in [\![\varphi]\!]^{\mathcal{M}^\sigma_\Psi}$.

Theorem 1 (Soundness of Abstraction). Let $\mathcal{M} = \langle \mathbf{S}^c, \mathbf{S}^c_0, \mathbf{P}, \Rightarrow \rangle$ be a transition system, Ψ a set of abstraction predicates, and $\mathcal{M}^+$, $\mathcal{M}^-$ the over-approximation and under-approximation of $\mathcal{M}$ with respect to Ψ. Then for any TCTL formula φ the following holds.

$$\gamma([\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^-_\Psi}) \subseteq [\![\varphi]\!]^{\mathcal{M}} \subseteq \gamma([\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^+_\Psi})$$

Here, α_φ is the abstraction function for TCTL formulas from Definition 7, and γ the concretization function from Definition 2.

We now give a criterion, based on the notion of regions, for a set of abstraction predicates that is sufficient for guaranteeing convergence of the over- and under-approximations in general.

3.3 Set of Basis Predicates

A *basis* is a set of abstraction predicates that is expressive enough to distinguish between two clock regions. If a basis is used for predicate abstraction, then the approximation is exact with respect to the TCTL logic, that is, the approximation is property-preserving.

Definition 9 (Basis [18,17]). Let $\mathcal{S}$ be a timed automaton with clock set C and let Ψ be a set of abstraction predicates. Then Ψ is a *basis* with respect to $\mathcal{S}$ iff for all clock valuations $\nu_1, \nu_2 \in \mathcal{V}_C$: $[\ (\forall \psi \in \Psi.\ \nu_1 \models \psi \Leftrightarrow \nu_2 \models \psi) \ \Rightarrow\ \nu_1 \cong \nu_2\ ]$.

For example, for a timed automaton $\mathcal{S}$ with clock set C and largest constant γ the (infinite) set of clock constraints *Constr*, the (infinite) set of invariant constraints *Inv*, the (finite) set of clock constraints *Constr*(γ), and the (finite) set of membership predicates for the quotient $\mathcal{V}_C$ modulo $\cong$ are all basis sets. Since the set of predicates *Constr*(γ) is finite, there is a finite basis for every timed automaton. Notice, however, that this basis is not necessarily minimal.

Theorem 2. Let $\mathcal{S}$ be a timed automaton, $\mathcal{M}$ the corresponding transition system, and φ a TCTL formula. Furthermore, let C be the set of clocks corresponding to $\mathcal{S}$ and φ, and γ the largest constant, which these clocks are compared to. Let Ψ be a basis with respect to C, and $\mathcal{M}^-_\Psi$, $\mathcal{M}^+_\Psi$ the under- and over-approximation of $\mathcal{S}$ with respect to Ψ. Then, for any TCTL formula φ,

$$[\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^-_\Psi} = [\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^+_\Psi}\ .$$

Corollary 1 (Basis Completeness). Let $\mathcal{S} = \langle L, L_0, C, I, P, E \rangle$, be a timed automaton, and $\mathcal{M}$ the corresponding transition system. Then for any TCTL formula φ, and initial state $l_0 \in L_0$ the following holds: (Ψ is a basis for $\mathcal{S}$ and φ)

$$(l_0, b_0) \in [\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^-_\Psi} \ \Leftrightarrow\ (l_0, \nu_0) \in [\![\varphi]\!]^{\mathcal{M}} \ \Leftrightarrow\ (l_0, b_0) \in [\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^+_\Psi}\ .$$

4 Symbolic Counterexamples for TCTL

Given a Kripke structure $\mathcal{M}$, with $\mathbf{S}_0$ initial states, and a TCTL formula φ, a *symbolic counterexample* carries a justification that $\mathcal{M}, \mathbf{S}_0 \not\models \varphi$. When we write $\vec{X}$, we mean a list of elements of the form $[X_0, \ldots]$, and $\vec{X}^m$ implies that the list $\vec{X}$ is of length $m+1$, that is, of the form $[X_0, \ldots, X_m]$. We write $c \vdash \mathcal{M}, C \not\models \varphi$ to denote that c is a counterexample, which demonstrates that for every state $s \in C$, $\mathcal{M}, s \not\models \varphi$.

For a transition system $\mathcal{M} = \langle \mathbf{S}, \mathbf{S}_0, \mathbf{P}, \mathbf{N} \rangle$, and a set $\mathbf{C} \subseteq \mathbf{S}$, let Ω_0 denote the set of all s-paths starting in any state in $\mathbf{C} \cap \mathbf{S}_0$, defined as $\Omega_0 = \{\rho \in f(s) \mid s \in \mathbf{C} \cap \mathbf{S}_0\}$[1] and $\Omega_0(t)$ be those states that can be reached from any state in $\mathbf{C} \cap \mathbf{S}_0$ at time point t, $\Omega_0(t) = \{s \in \mathbf{S} \mid s = \rho(t) \text{ and } \rho \in \Omega_0\}$. For a bounded modality with time bound $t \sim c$, let $\zeta \in \mathbf{AP}$ be an atomic proposition that holds in a state s if and only if $s \in \Omega_0(t)$.

Definition 10 (Symbolic Counterexamples). [2] Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{S}_0, \mathbf{P}, \mathbf{N} \rangle$ be a transition system, where $\mathbf{S} = L \times \mathcal{V}_C$, $\mathbf{S}_0 \subseteq \mathbf{S}$, and $\mathbf{N}$ is the transition relation. For a TCTL formula φ, and a set of states $\mathbf{C} \subseteq \mathbf{S}_0$, a symbolic counterexample c justifying $\mathcal{M}, \mathbf{C} \not\models \varphi$ has the form $\vec{X}^m$, and is defined as follow.

1. A counterexample $c \vdash \mathcal{M}, \mathbf{C} \not\models \mathbf{EG}_{\sim c}\, \varphi$ is a list $c = \vec{X}^m$, such that $\exists\, \mathbf{C}' \subseteq \mathbf{S}$ with
   (a) $\mathcal{M}, \mathbf{C}' \not\models \varphi \wedge \zeta$,
   (b) $X_0 \subseteq \mathbf{C}'$,
   (c) $X_{i+1} \subseteq X_i \cup \widetilde{pre}(\mathbf{N})(X_i)$, for $i < m$,
   (d) $\mathbf{C} = X_m$.
2. A counterexample $c \vdash \mathcal{M}, \mathbf{C} \not\models \mathbf{AG}_{\sim c}\, \varphi$ is a list $c = \vec{X}^m$, such that $\exists\, \mathbf{C}' \subseteq \mathbf{S}$ with
   (a) $\mathcal{M}, \mathbf{C}' \not\models \varphi \wedge \zeta$,
   (b) $X_0 \subseteq \mathbf{C}'$,
   (c) $X_{i+1} \subseteq X_i \cup pre(\mathbf{N})(X_i)$, for $i < m$,
   (d) $\mathbf{C} = X_m$.

5 Incremental Abstraction-Refinement Algorithm

Definition 11 (Set Inclusion w.r.t. Clock Valuations). For two sets of states $S = \{(l_1, \nu_1), \ldots, (l_m, \nu_m)\}$ and $S' = \{(l'_1, \nu'_1), \ldots, (l'_n, \nu'_n)\}$, the *set inclusion with respect to clock valuations* relation $S \subseteq_\nu S'$ is defined as:

$S \subseteq_\nu S'$ iff $[n = m,\ l_i = l'_i,$ and $\nu_i \subseteq \nu'_i,$ for all $0 \leq i \leq m]$.

The abstraction-refinement algorithm is displayed in Figure 2. The variables Ψ_n and Ψ_a store the currently unused (new) and used (actual) abstraction predicates, respectively. Initially, Ψ_a contains those predicates from the basis that correspond to the bounded modalities of φ, and possibly some predicates derived from the timing constraints of the automaton, and Ψ_n contains the remaining predicates (lines (1)-(2)). First, it is checked if $s_0 \in \gamma([\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}_{\Psi_a}})$ by calling a finite-state CTL model checker

[1] A s-path through $\mathbf{S}$ is a map ρ from $\mathbb{R}_{\geq 0}$ to $\mathbf{S}$, satisfying $\rho(0) = s$, for $s \in \mathbf{S}$. f is a map, which assigns to every $s \in \mathbf{S}$ a set of s-paths through $\mathbf{S}$ [1]. See also Appendix A.

[2] For lack of space we define here only counterexamples for $\mathbf{EG}_{\sim c}\, \varphi$ and $\mathbf{AG}_{\sim c}\, \varphi$, but the full definition can be found in Appendix B.

Algorithm: *abstract_and_refine*
Input: $\mathcal{M}$, $\mathbf{S}$, s_0, $\mathbf{N}$, φ, Ψ
Output: answer to model checking query "$\mathcal{M}, s_0 \models \varphi$?"

choose $\Psi' = \{\psi_1, \ldots, \psi_i\}$ from Ψ; (1)
$\Psi_n := \Psi \setminus \Psi'$; $\Psi_a := \Psi'$; (2)
loop (3)
  if $s_0 \in \gamma([\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^-_{\Psi_a}})$ **then return** true (4)
  else if $s_0 \notin \gamma([\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^+_{\Psi_a}})$ **then return** false (5)
  else let $[X_0, X_1, \ldots, X_n]$ be a counterexample in $\mathcal{M}^+_{\Psi_a}$ (6)
    if there exists $[X^c_0, X^c_1, \ldots, X^c_n]$ s.t. $X^c_i \subseteq_\nu \gamma(X_i)$ for all $0 \le i \le n$ (7)
      and $[X^c_0, X^c_1, \ldots, X^c_n]$ is counterexample in $\mathcal{M}$ (8)
    then return false (9)
    else let k s.t. $X^c_{k+1} \not\subseteq X^c_k \cup pre(\mathbf{N})(X^c_k)$; $S = pre(\mathbf{N})(X^c_{k-1}) \subseteq \mathbf{S}$ (10)
      choose feasible $\Psi' = \{\psi_1, \ldots, \psi_i\} \subseteq \Psi_n$ s.t. $\exists (l, \nu) \in S.\ \nu \models \psi_i$; (11)
      $\Psi_a := \Psi_a \cup \Psi'$; $\Psi_n := \Psi_n \setminus \Psi'$ (12)
    endif (13)
  endif (14)
endloop (15)

Fig. 2. Iterative abstraction-refinement algorithm.

that generates symbolic evidence, as for example the WMC model checker [10,20]. If indeed the under-approximation satisfies the abstracted formula $\alpha_\varphi(\varphi)$, then, by Corollary 1, $\mathcal{M}$ also satisfies φ and the algorithm returns **true** (line (4)). As next, we check if $s_0 \notin \gamma([\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^+_{\Psi_a}})$. If the over-approximation does not satisfy $\alpha_\varphi(\varphi)$, then, also by Corollary 1, $\mathcal{M}$ does not satisfy φ and the algorithm returns **false** (line (5)). Otherwise (line (6)), that is, if $s_0 \notin \gamma([\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^-_{\Psi_a}})$ and $s_0 \in \gamma([\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^+_{\Psi_a}})$, the CTL model checker returns a counterexample in the form of an abstract list of sets of states $[X_0, X_1, \ldots, X_n]$, where the initial state of $\mathcal{M}^+_{\Psi_a}$ is contained in X_n. If for the abstract list of sets of states there exists a corresponding list of concrete sets of states, which is indeed a counterexample for the concrete transition system and given (TCTL) formula, then we obtain a counterexample for the concrete model-checking problem (lines (7)-(9)). This requires checking the satisfiability of a Boolean formula with linear arithmetic constraints, which in turns requires quantifier elimination, and can be performed using, for example, DDDs [16], or the satisfiability checker ICS [12]. In case the abstract counterexample is spurious, there exists a smallest index k such that $X^c_{k+1} \not\subseteq X^c_k \cup pre(\mathbf{N})(X^c_k)$ (line (10)). k is the index of the list of states X^c_k that can reach in one step states in X^c_{k-1}, but which can no longer be reached from the states in X^c_{k+1}. Now, we have to choose those predicates from the basis that are satisfied by the valuations ν of some states $(l, \nu) \in S$ (lines (10)-(11)). We add the selected predicates to Ψ_a and compute a new abstraction. Notice that the concretization function γ actually depends on the current set Ψ_a of abstraction predicates. The iterative abstraction-refinement algorithm terminates

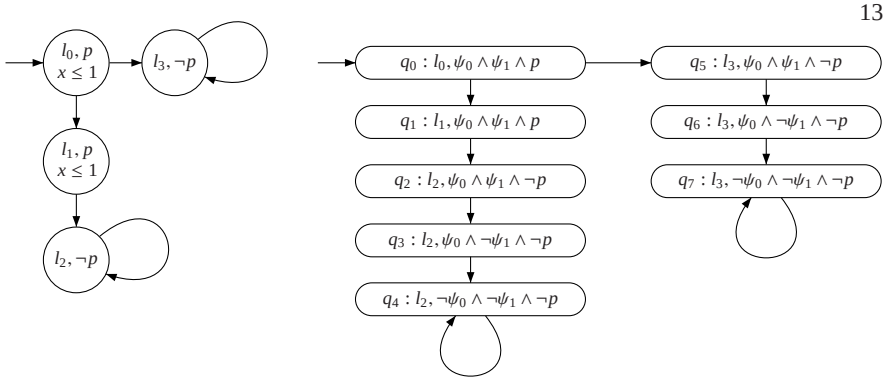

Fig. 3. Timed automaton (left) and over-approximation with $\psi_0 \equiv (z \leq 2)$, $\psi_1 \equiv (x \leq 1)$ (right) for Example 1.

after a finite number of refinements, yielding a sound and complete decision procedure for checking whether or not a timed automaton satisfies a given TCTL formula.

Theorem 3 (Termination, Soundness, and Completeness). Let $\mathcal{M}$ be a transition system with a corresponding finite basis Ψ, and φ a TCTL formula. Then the algorithm in Figure 2 always terminates. Moreover, if it terminates with **true**, then $\mathcal{M} \models \varphi$, and if the result is **false**, then $\mathcal{M} \not\models \varphi$.

Proof. Let n be the cardinality of the basis Ψ. Every execution of the loop (line (3)) adds at least one new predicate from the basis to the set Ψ_a (line (12)). After at most n iterations, according to Theorem 2, $[\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^-_\Psi} = [\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^+_\Psi}$. By Theorem 1, $\gamma([\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^-_\Psi}) = [\![\varphi]\!]^{\mathcal{M}} = \gamma([\![\alpha_\varphi(\varphi)]\!]^{\mathcal{M}^+_\Psi})$, and by Corollary 1, $\mathcal{M}^+_\Psi$ satisfies the formula $\alpha_\varphi(\varphi)$ if and only if $\mathcal{M}$ satisfies φ. Thus, the algorithm terminates, since either φ can be established or a concrete counterexample can be derived. □

Example 1. Consider the timed automaton from Figure 3, left side, for which we want to establish the property $\mathbf{EG}_{\leq 2}\, p$. A given basis for this system and this property is $\Psi = \{x = 0, z = 0, x = 1, z = 1, x = 2, z = 2, x \geq 1, x \leq 1, z \geq 1, z \leq 1, x \geq 2, x \leq 2, z \geq 2\}$. The transition system with the initial over-approximation using the abstraction predicates $\psi_0 \equiv (z \leq 2)$ and $\psi_1 \equiv (x \leq 1)$ is shown in the right side of Figure 3. Model checking the abstract formula $\varphi = \alpha_\varphi(\mathbf{EG}_{\leq 2}\, p) = \mathbf{EG}(p \wedge \psi_0)$ on the transition system $\mathcal{M}^+_{\{\psi_0,\psi_1\}}$ returns **false**, since $s_0 = (l_0, x = z = 0) \notin \gamma([\![\varphi]\!]^{\mathcal{M}^+_{\{\psi_0,\psi_1\}}})$. The finite-state model-checking algorithm WMC [10,20] returns the symbolic counterexample $[X_0, X_1, X_2]$, where $X_0 = \{q_2, q_5\}$, $X_1 = \{q_1, q_2, q_5\}$, and $X_2 = \{q_0, q_1, q_2, q_5\}$. Recall the meaning of this counterexample list: the set X_0 consists of those states that do not satisfy the subformula $p \wedge \psi_0$, X_1 are the states in X_0 plus those states that can reach only states in X_0 in one step, and so forth. The concretization of this list of set of states is $[X^c_0, X^c_1, X^c_2]$ where $X^c_i \subseteq_\nu \gamma(X_i)$ for all $i = 0, 1, 2$.[3]

$$\gamma(X_0) = (l_2,\ z \leq 2\ \wedge\ x \leq 1)\ \cup\ (l_3,\ z \leq 2\ \wedge\ x \leq 1)$$

[3] To simplify the notation we denote sets of concrete states such as $\{(l, \nu) \mid l = l_0 \wedge \nu(x) < 1 \wedge \nu(z) \leq 2\}$ by $(l_0, x < 1 \wedge z \leq 2)$.

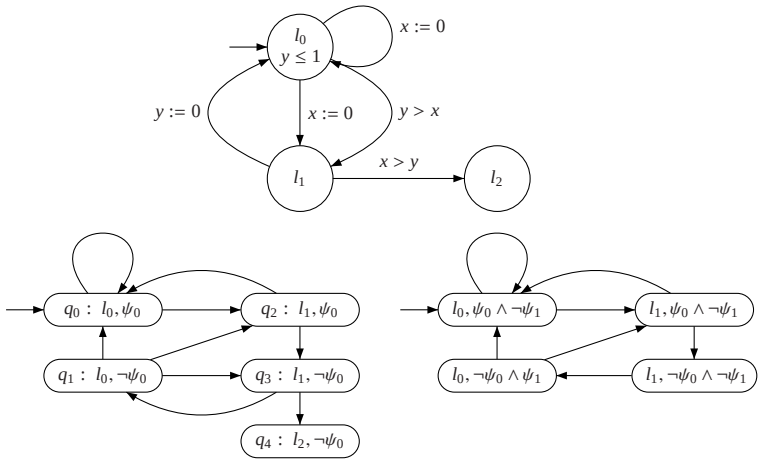

Fig. 4. Timed automaton and over-approximations with $\psi_0 \equiv (x = 0)$ (lower left part) and $\Psi = \{x = 0, x > y\}$ (lower right part) for Example 2.

$$\gamma(X_1) = \gamma(X_0) \cup (l_1,\ z \leq 2 \wedge x \leq 1)$$
$$\gamma(X_2) = \gamma(X_1) \cup (l_0,\ z \leq 2 \wedge x \leq 1)$$

Now we have to check if the list $[X_0^c, X_1^c, X_2^c]$ is a counterexample on the concrete transition system, by checking the satisfiability of the conditions from Definition 2 (1). All four conditions are satisfied in our example, thus, the abstraction-refinement algorithm terminates with the answer **false**, meaning that the timed system from Figure 3 does not satisfies the property $\mathbf{EG}_{\leq 2}\, p$.

We now illustrate a situation in which the concretization of the abstract counterexample yields a spurious concrete counterexample, which initiates a refinement step.

Example 2. Consider the timed automaton from the upper part in Figure 4. We want to prove that location l_2 is never reached, specified as $\varphi = \neg\mathbf{E}[\mathrm{tt}\,\mathbf{U}\,at\_l_2]$, where the atomic (boolean) proposition at_l_2 is true if the system is in location l_2. Note that φ is actually a CTL formula, and need therefore not to be abstracted. φ can be rewritten as $\mathbf{AG}\neg at\_l_2$. A given basis for this system is $\Psi = \{x = 0, y = 0, x \leq 1, x \geq 1, y \leq 1, y \geq 1, x > y, x < y\}$. The transition system of the initial over-approximation with the single abstraction predicate $\psi_0 \equiv (x = 0)$ is shown in the lower left part of Figure 4. Model checking $\varphi = \mathbf{AG}(\neg at\_l_2)$ on the transition system $\mathcal{M}^-_{\{x=0\}}$ returns **false**, since $s_0 = (l_0, x = y = 0) \notin \gamma([\![\varphi]\!]^-_{\{x=0\}})$. The list $[X_0, X_1, X_2, X_3]$, with $X_0 = \{q_4\}$, $X_1 = \{q_3, q_4\}$, $X_2 = \{q_1, q_2, q_3, q_4\}$, $X_3 = \{q_0, q_1, q_2, q_3, q_4\}$ is returned as a counterexample. The concretization of this counterexample yields

$$\gamma(X_0) = (l_2,\ x > 0 \wedge y \geq 0)$$
$$\gamma(X_1) = (l_1,\ x > 0 \wedge y \geq 0) \cup (l_2,\ x > 0 \wedge y \geq 0)$$
$$\gamma(X_2) = (l_0,\ x > 0 \wedge y \geq 0) \cup (l_1,\ x \geq 0 \wedge y \geq 0) \cup (l_2,\ x > 0 \wedge y \geq 0)$$
$$\gamma(X_3) = (l_0,\ x \geq 0 \wedge y \geq 0) \cup (l_1,\ x \geq 0 \wedge y \geq 0) \cup (l_2,\ x > 0 \wedge y \geq 0)$$

Now, we have to check if there is a corresponding symbolic counterexample on the concrete system, that is, there exists $[X_0^c, X_1^c, X_2^c, X_3^c]$, with $X_i^c \subseteq_\nu \gamma(X_i)$, for all $i = 0, \ldots, 3$. This is the case if the following formula is valid.

$$\phi = \exists\, X_0^c \subseteq_\nu \gamma(X_0), \ldots, X_3^c \subseteq_\nu \gamma(X_3), \exists\, \mathbf{C}' \subseteq \mathbf{S}.$$
$$(\mathcal{M} \not\models \mathbf{C}' \Rightarrow \neg at\_l_2) \wedge (X_0^c = \mathbf{C}') \wedge \wedge_{i=0}^{2} (X_{i+1}^c \Rightarrow (X_i^c \vee pre(\mathbf{N})(X_i^c)))$$

Here, ϕ is not satisfiable, since on the concrete transition system $X_2^c \not\subseteq (X_1^c \cup pre(\mathbf{N})(X_1^c))$ does not hold. $X_1^c \cup pre(\mathbf{N})(X_1^c) = (l_2, x > 0 \wedge y \geq 0) \cup (l_1, x > y \geq 0)$ does not contain a state with location l_0, but according to Definition 11 and the fact that $X_2^c \subseteq_\nu \gamma(X_2)$, X_2^c must contain a state with a l_0 location, and therefore $X_2^c \not\subseteq (X_1^c \cup pre(\mathbf{N})(X_1^c))$.

Now, $k = 1$ in our abstraction refinement algorithm, and $S = \{(l_1, \nu) \mid \nu(x) > \nu(y)\}$, and the new abstraction predicate is chosen to be $\psi_1 \equiv x > y$. Figure 4 (lower right part) shows the reachable fragment of the resulting approximation of $\mathcal{M}$ with $\Psi = \{\psi_0, \psi_1\}$. Model checking the formula $\varphi = \mathbf{AG}(\neg at\_l_2)$ on $\mathcal{M}^-_{\{\psi_0,\psi_1\}}$ succeeds, since $s_0 = (l_0, \psi_0 \wedge \neg\psi_1) \in \gamma([\![\varphi]\!]^{\mathcal{M}^-_{\{\psi_0,\psi_1\}}})$.

6 Conclusion

We have defined symbolic counterexamples for the full TCTL logic, and used them for developing a verification algorithm for timed automata based on predicate abstraction, untimed model checking, and decision procedures for the Boolean combination of linear arithmetic constraints. The main advantage of this approach is that finite state abstractions are computed lazily and incrementally. Abstraction refinement terminates quickly, as a multitude of spurious counterexamples is eliminated in every refinement step through the use of symbolic counterexamples for TCTL. Using symbolic counterexamples makes it possible to apply the abstraction refinement paradigm to the full TCTL logic.

Dual to the notion of symbolic counterexamples, we can also define symbolic witnesses for the full TCTL, as extensions of symbolic witnesses for CTL [20]. These witnesses and counterexamples can be seen as proofs for the judgment that the timed automaton does or does not satisfy the given TCTL formula, and can be independently verified using a satisfiability checker that can decide the theory of linear arithmetic with reals. Moreover, explicit linear or tree-like witnesses and counterexamples can be extracted from these symbolic evidence, following the approach in [20] for the untimed case.

The method of lazy approximation is also applicable for other real-time logics. Moreover, this technique can readily be extended to also apply to richer models than timed automata, such as parameterized timed automata, timed automata with other infinite data types, or even to hybrid systems. The price to pay is that such extensions are necessarily incomplete.

Work in progress investigates the combination of lazy approximation with the approach to controller synthesis for finite-state systems presented in [20], for synthesizing real-time controllers.

References

1. R. Alur, C. Courcoubetis, and D. Dill. Model checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.
2. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 25 April 1994.
3. R. Alur, A. Itai, R.P. Kurshan, and M. Yannakakis. Timing verification by successive approximation. *Information and Computation*, 118(1):142–157, 1995.
4. G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelánek. Lower and upper bounds in zone based abstractions of timed automata. *LNCS*, 2988:312–326, 2004.
5. S. Bensalem, Y. Lakhnech, and S. Owre. Computing abstractions of infinite state systems compositionally and automatically. *LNCS*, 1427:319–331, 1998.
6. E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. *LNCS*, 1855, 2000.
7. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis. *4th ACM Symposium on Principles of Programming Languages*, January 1977.
8. S. Das and D. Dill. Successive approximation of abstract transition relations. In *Proc. of Logic in Computer Science (LICS2001)*, 2001.
9. C. Daws and S. Tripakis. Model checking of real-time reachability properties using abstractions. *LNCS*, 1384:313–329, 1998.
10. L. de Moura, S. Owre, H. Rueß, J. Rushby, N. Shankar, M. Sorea, and A. Tiwari. SAL 2. In *16th Conference on Computer Aided Verification*, LNCS. Springer-Verlag, 2004. Tool description.
11. D. Dill and H. Wong-Toi. Verification of real-time systems by successive over and under approximation. *LNCS*, 939:409–422, 1995.
12. J.-C. Filliâtre, S. Owre, H. Rueß, and N. Shankar. ICS: Integrated canonizer and solver. In G. Berry, H. Comon, and A. Finkel, editors, *Proceedings of CAV'2001*, volume 2102 of *LNCS*, pages 246–249. Springer-Verlag, 2001.
13. S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. *LNCS*, 1254:72–83, 1997.
14. T.A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Lazy abstraction. In *Symposium on Principles of Programming Languages*, pages 58–70, 2002.
15. Y. Lachnech, S. Bensalem, S. Berezin, and S. Owre. Incremental verification by abstraction. *LNCS*, 2031:98–112, 2001.
16. J. Møller, J. Lichtenberg, H. R. Andersen, and H. Hulgaard. Difference decision diagrams. In *Computer Science Logic*, The IT University of Copenhagen, Denmark, September 1999.
17. M. Oliver Möller, Harald Rueß, and Maria Sorea. Predicate abstraction for dense real-time systems. *ENTCS*, 65(6), 2002. http://www.elsevier.com/locate/entcs/volume65.html.
18. M.O. Möller, H. Rueß, and M. Sorea. Predicate abstraction for dense real-time systems. Technical Report BRICS-RS-01-44, Dept. of Computer Science, University of Aarhus, 2001.
19. H. Saïdi and N. Shankar. Abstract and model check while you prove. *LNCS*, 1633:443–454, 1999.
20. N. Shankar and M. Sorea. Counterexample-driven model checking. Technical Report SRI-CSL-03-04, SRI International, 2003. `http://www.csl.sri.com/users/sorea/reports/wmc.ps.gz`.
21. Maria Sorea. *Verification of real-time systems through lazy approximations*. PhD thesis, University of Ulm, Germany, 2004. Forthcoming.
22. S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001. Kluwer Academic Publishers.
23. H. Wong-Toi. *Symbolic Approximations for Verifying Real-Time Systems*. PhD thesis, Stanford University, November 1994.

Note to the reviewer: the Appendices are not part of the official submission.

A TCTL Syntax and Semantics

Definition 12 (TCTL Syntax). Let **AP** be a set of atomic propositions, and $p \in \mathbf{AP}$. The formulas of TCTL are inductively defined as follows:

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{E}[\varphi_1 \, \mathbf{U}_{\sim c} \, \varphi_2] \mid \mathbf{A}[\varphi_1 \, \mathbf{U}_{\sim c} \, \varphi_2]$$

Definition 13 (s-Path). For a set **S** of states and $s \in \mathbf{S}$, an s-path through **S** is a map ρ from $\mathbb{R}_{\geq 0}$ to **S**, satisfying $\rho(0) = s$.

Definition 14 (TCTL-Structure). A structure for a TCTL formula is a tuple $\mathcal{M} = \langle \mathbf{S}, \mathbf{S}_0, \mu, f \rangle$, where

- **S** is a set of states.
- $\mathbf{S}_0 \in \mathbf{S}$ is an initial state.
- $\mu : \mathbf{S} \to \mathcal{P}(\mathbf{AP})$ is a labeling function, which assigns to each state a set of atomic propositions true in that state.
- f is a map, which assigns to every $s \in \mathbf{S}$ a set of s-paths through **S**, and is closed under suffix and fusion.[4]

Definition 15 (Semantics of TCTL). Given a TCTL-structure $\mathcal{M} = \langle \mathbf{S}, \mathbf{S}_0, \mu, f \rangle$, over a set $\mathbf{S} = L \times \mathcal{V}_C$, of timed configurations, and a TCTL-formula φ, the set of states $[\![\varphi]\!]^{\mathcal{M}}$, which validates the formula φ is defined inductively over the structure of φ.

$$\begin{aligned}
[\![p]\!]^{\mathcal{M}} &:= \{s \in \mathbf{S} \mid p \in \mu(s)\} \\
[\![\neg\varphi]\!]^{\mathcal{M}} &:= \mathbf{S} \setminus [\![\varphi]\!] \\
[\![\varphi_1 \wedge \varphi_2]\!]^{\mathcal{M}} &:= [\![\varphi_1]\!]^{\mathcal{M}} \cap [\![\varphi_2]\!]^{\mathcal{M}} \\
[\![\mathbf{E}[\varphi_1 \, \mathbf{U}_{\sim c} \, \varphi_2]]\!]^{\mathcal{M}} &:= \{s \in \mathbf{S} \mid \text{for some } \rho \in f(s), \text{ for some } t \sim c, \rho(t) \in [\![\varphi_2]\!]^{\mathcal{M}} \text{ and} \\
&\qquad \rho(t') \in [\![\varphi_1]\!]^{\mathcal{M}}, \text{ for all } 0 \leq t' < t\} \\
[\![\mathbf{A}[\varphi_1 \, \mathbf{U}_{\sim c} \, \varphi_2]]\!]^{\mathcal{M}} &:= \{s \in \mathbf{S} \mid \text{for all } \rho \in f(s), \text{ for some } t \sim c, \rho(t) \in [\![\varphi_2]\!]^{\mathcal{M}} \text{ and} \\
&\qquad \rho(t') \in [\![\varphi_1]\!]^{\mathcal{M}}, \text{ for all } 0 \leq t' < t\}
\end{aligned}$$

Definition 16 (Other Temporal Operators). The following temporal operators can be defined as usual.

$$\begin{aligned}
\mathbf{EF}_{\sim c} \, \varphi &:= \mathbf{E}[\mathsf{tt} \, \mathbf{U}_{\sim c} \, \varphi] \\
\mathbf{AF}_{\sim c} \, \varphi &:= \mathbf{A}[\mathsf{tt} \, \mathbf{U}_{\sim c} \, \varphi] \\
\mathbf{EG}_{\sim c} \, \varphi &:= \neg\mathbf{AF}_{\sim c} \, \neg\varphi \\
\mathbf{AG}_{\sim c} \, \varphi &:= \neg\mathbf{EF}_{\sim c} \, \neg\varphi \\
\mathbf{E}[\varphi_1 \, \mathbf{R}_{\sim c} \, \varphi_2] &:= \neg\mathbf{A}[\neg\varphi_1 \, \mathbf{U}_{\sim c} \, \neg\varphi_2] \\
\mathbf{A}[\varphi_1 \, \mathbf{R}_{\sim c} \, \varphi_2] &:= \neg\mathbf{E}[\neg\varphi_1 \, \mathbf{U}_{\sim c} \, \neg\varphi_2]
\end{aligned}$$

[4] Details on suffix and fusion closure can be found in [1].

B Symbolic Counterexamples for TCTL

Definition 17 (TCTL Symbolic Counterexamples). Let $\mathcal{M} = \langle \mathbf{S}, \mathbf{S}_0, \mathbf{P}, \mathbf{N} \rangle$ be a transition system, where $\mathbf{S} = L \times \mathcal{V}_C$, $\mathbf{S}_0 \subseteq \mathbf{S}$, and $\mathbf{N}$ is the transition relation. For a TCTL formula φ, and a set of states $C \subseteq \mathbf{S}_0$, a symbolic counterexample c justifying $\mathcal{M}, C \models \varphi$ has the form $\vec{X}^m$, where $\vec{X}$ is a sequence of states of length $m+1$, which is defined inductively over the structure of φ. Note that $\langle c_1, \ldots, c_k \rangle \vdash \mathcal{M}, C_1 \cup \ldots \cup C_k \not\models \varphi$ iff $c_i \vdash \mathcal{M}, C_i \not\models \varphi$ for $1 \leq i \leq k$.

1. A counterexample $c \vdash \mathcal{M}, C \not\models \mathbf{EG}_{\sim c}\, \varphi$ is a list $c = \vec{X}^m$, such that $\exists\, C' \subseteq \mathbf{S}$ with (a) $\mathcal{M}, C' \not\models \varphi \wedge \zeta$, (b) $X_0 \subseteq C'$, (c) $X_{i+1} \subseteq X_i \cup \widetilde{pre}(\mathbf{N})(X_i)$, for $i < m$, (d) $C = X_m$.
2. A counterexample $c \vdash \mathcal{M}, C \not\models \mathbf{AG}_{\sim c}\, \varphi$ is a list $c = \vec{X}^m$, such that $\exists\, C' \subseteq \mathbf{S}$ with (a) $\mathcal{M}, C' \not\models \varphi \wedge \zeta$, (b) $X_0 \subseteq C'$, (c) $X_{i+1} \subseteq X_i \cup pre(\mathbf{N})(X_i)$, for $i < m$, (d) $C = X_m$.
3. A counterexample $c \vdash \mathcal{M}, C \not\models \mathbf{EF}_{\sim c}\, \varphi$ is a list $c = \vec{X}^m$, such that $\exists\, C' \subseteq \mathbf{S}$ with (a) $\mathcal{M}, C' \not\models \varphi \wedge \zeta$, (b) $X_0 \subseteq C'$, (c) $X_{i+1} \subseteq X_i$, for $i < m$, (d) $C = X_m \subseteq \widetilde{pre}(\mathbf{N})(X_m)$.
4. A counterexample $c \vdash \mathcal{M}, C \not\models \mathbf{AF}_{\sim c}\, \varphi$ is a list $c = \vec{X}^m$, such that $\exists\, C' \subseteq \mathbf{S}$ with (a) $\mathcal{M}, C' \not\models \varphi \wedge \zeta$, (b) $X_0 \subseteq C'$, (c) $X_{i+1} \subseteq X_i$, for $i < m$, (d) $C = X_m \subseteq pre(\mathbf{N})(X_m)$.
5. A counterexample $c \vdash \mathcal{M}, C \not\models \mathbf{E}[\varphi_1\, \mathbf{U}_{\sim c}\, \varphi_2]$ is a list $c = \vec{X}^m$, such that $\exists\, C', C'' \subseteq \mathbf{S}$ with (a) $\mathcal{M}, C'' \not\models \varphi_2 \wedge \zeta$, and $\mathcal{M}, C' \not\models \varphi_1 \vee (\varphi_2 \wedge \zeta)$, (b) $C' \subseteq X_0 \subseteq C''$, (c) $X_{i+1} \subseteq (\widetilde{pre}(\mathbf{N})(X_i) \cap X_i) \cup C'$, (d) $C = X_m \subseteq \widetilde{pre}(\mathbf{N})(X_m)$.
6. A counterexample $c \vdash \mathcal{M}, C \not\models \mathbf{A}[\varphi_1\, \mathbf{U}_{\sim c}\, \varphi_2]$ is a list $c = \vec{X}^m$, such that $\exists\, C', C'' \subseteq \mathbf{S}$ with (a) $\mathcal{M}, C'' \not\models \varphi_2 \wedge \zeta$, and $\mathcal{M}, C' \not\models \varphi_1 \vee (\varphi_2 \wedge \zeta)$, (b) $C' \subseteq X_0 \subseteq C''$, (c) $X_{i+1} \subseteq (pre(\mathbf{N})(X_i) \cap X_i) \cup C'$, (d) $C = X_m \subseteq pre(\mathbf{N})(X_m)$.
7. A counterexample $c \vdash \mathcal{M}, C \not\models \mathbf{E}[\varphi_1\, \mathbf{R}_{\sim c}\, \varphi_2]$ is a list $c = \vec{X}^m$, such that $\exists\, C', C'' \subseteq \mathbf{S}$ with (a) $\mathcal{M}, C'' \not\models \varphi_2 \vee \neg\zeta$, and $\mathcal{M}, C' \not\models \varphi_1 \vee (\neg\varphi_2 \wedge \zeta)$, (b) $C' \subseteq X_0 \subseteq C''$, (c) $X_{i+1} \subseteq (\widetilde{pre}(\mathbf{N})(X_i) \cap C') \cup X_i$, (d) $C = X_m$.
8. A counterexample $c \vdash \mathcal{M}, C \not\models \mathbf{A}[\varphi_1\, \mathbf{R}_{\sim c}\, \varphi_2]$ is a list $c = \vec{X}^m$, such that $\exists\, C', C'' \subseteq \mathbf{S}$ with (a) $\mathcal{M}, C'' \not\models \varphi_2 \vee \neg\zeta$, and $\mathcal{M}, C' \not\models \varphi_1 \vee (\neg\varphi_2 \wedge \zeta)$, (b) $C' \subseteq X_0 \subseteq C''$, (c) $X_{i+1} \subseteq (pre(\mathbf{N})(X_i) \cap C') \cup X_i$, (d) $C = X_m$.
9. A counterexample $c \vdash \mathcal{M}, c \not\models \varphi_1 \vee \varphi_2$ is a list $c = \vec{X}^0$, such that $\exists\, C', C'' \subseteq \mathbf{S}$ with (a) $\mathcal{M}, C' \not\models \varphi_1$, and $\mathcal{M}, C'' \not\models \varphi_2$, (b) $C = X_0 = C' \cap C''$.
10. A counterexample $c \vdash \mathcal{M}, C \not\models \varphi_1 \wedge \varphi_2$ is a list $c = \vec{X}^0$, such that $\exists\, C', C'' \subseteq \mathbf{S}$ with (a) $\mathcal{M}, C' \not\models \varphi_1$, and $\mathcal{M}, C'' \not\models \varphi_2$, (b) $C = X_0 = C' \cup C''$.