Electronic Notes in Theoretical Computer Science 65 No. 6 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume65.html 20 pages

Predicate Abstraction for Dense Real-Time Systems ⋆

M. Oliver Möller [a], Harald Rueß [b], and Maria Sorea [b,1]

[a] *BRICS, University of Aarhus, Department of Computer Science, Ny Munkegade, Building 540, 8000 Århus C, Denmark* [2]

[b] *SRI International, Computer Science Laboratory, 333 Ravenswood Avenue, Menlo Park, CA 94025, USA*

Abstract

We propose predicate abstraction as a means for verifying a rich class of safety and liveness properties for dense real-time systems. First, we define a restricted semantics of timed systems which is observationally equivalent to the standard semantics in that it validates the same set of μ-calculus formulas without a next-step operator. Then, we recast the model checking problem $\mathcal{S} \models \varphi$ for a timed automaton $\mathcal{S}$ and a μ-calculus formula φ in terms of predicate abstraction. Whenever a set of abstraction predicates forms a so-called *basis*, the resulting abstraction is strongly preserving in the sense that $\mathcal{S}$ validates φ iff the corresponding finite abstraction validates this formula φ. Now, the abstracted system can be checked using familiar μ-calculus model checking. Like the region graph construction for timed automata, the predicate abstraction algorithm for timed automata usually is prohibitively expensive. In many cases it suffices to compute an approximation of a finite bisimulation by using only a subset of the basis of abstraction predicates. Starting with some coarse abstraction, we define a finite sequence of refined abstractions that converges to a strongly preserving abstraction. In each step, new abstraction predicates are selected nondeterministically from a finite basis. Counterexamples from failed μ-calculus model checking attempts can be used to heuristically choose a small set of new abstraction predicates for refining the abstraction.

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

⋆ This research was supported by the National Science Foundation under grants CCR-00-82560 and CCR-00-86096. Most of this research has been conducted while the first author was visiting SRI International, July/August 2001.

[1] Also affiliated with Universität Ulm, Germany.

[2] Basic Research in Computer Science, funded by the Danish National Research Foundation.

©2002 Published by Elsevier Science B. V.

the times at which transitions may occur. Given a timed automaton and a property expressed in a timed logic such as TCTL [1] or T_μ[11], model checking answers the question whether or not 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. Algorithms directly based on the explicit construction of such a partition are however unlikely to perform efficiently in practice, 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. A recent overview of data structures for representing regions in a symbolic way together with algorithms and tools for verifying real-time systems is given, for example, by Yovine [20].

We propose a novel algorithm for verifying a rich class of safety and liveness properties of timed automata based on computing finite abstractions of timed automata, model checking, and successive refinement of abstractions. Without sacrificing completeness, this algorithm does usually not require to compute the complete region graph in order to decide model checking problems. In the worst case, it terminates with a strongly preserving abstraction of the given model checking problem.

The computation of finite approximations of timed systems is based on the concepts of *abstract interpretation* [5], and, in particular, of *predicate abstraction* [10]. Given a transition system and a finite set of predicates, this method determines a finite abstraction, where each state of the abstract state space is a truth assignment to the abstraction predicates. The abstraction is conservative in the sense that a propositional μ-calculus formula holds for the concrete system if it holds for the predicate-abstracted system [17]. Since the reverse statement does not hold in general, predicate abstraction has so far mainly been used to only prove safety but not liveness properties.

The main problem with applying predicate abstraction in general is to come up with an appropriate set of abstraction predicates. In the case of timed automata, we show that a set of abstraction predicates expressive enough to distinguish between any two clock regions determines a strongly preserving abstraction, in the sense that the timed system satisfies the property under consideration if and only if the predicate-abstracted system satisfies this property. The main technical problem in the definition of the abstraction is to guarantee fairness in the abstract model; that is, to prevent delay steps to be abstracted into self-loops on the abstract system. Uribe [19] distinguishes between three different approaches in the literature for building fairness into the abstraction: first, by adding new fairness constraints to the abstract system, second, by incorporating fairness into the logic, and third, by modifying the finite-state model checker. We present a forth approach that addresses this problem by introducing a certain restriction on delay steps, and we show that the corresponding restricted semantics of timed automata is equivalent to a time-progressing semantics in the sense that these different interpretations

validate the same set of propositional μ-calculus formulas (without next-step operator). Altogether, our predicate abstraction algorithm determines a decision procedure for checking whether or not a timed automata satisfies some given μ-calculus formula.

The set of abstraction predicates required to compute a strongly preserving abstraction, a so-called *basis*, can still be excessively large. Starting with a trivial over-approximation, we successively select abstraction predicates from the finite basis. Counterexamples from failed model checking attempts are used in guiding the selection. The idea of counterexample-guided refinement has been used before by many researchers, and recent work includes [4,7,14]. In contrast to these approaches, we use the counterexample only as a heuristic for selecting good pivot predicates from a fixed, predetermined pool of abstraction predicates in order to speed-up convergence of the approximation processes. Also, verification techniques for infinite-state systems based on predicate abstraction [10,3,17] are usually incomplete. Our abstraction method for timed systems, however, is complete, since, in the limit, we construct a finite system that satisfies the same set of formulas under consideration as the original timed system.

Dill and Wong-Toi [8] 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. Based on techniques of predicate abstraction, Namjoshi and Kurshan's algorithm [16] computes a finite bisimulation whenever it exists. Thus, in principle, their algorithm could be applied to compute finite bisimulations of timed automata. Currently it is unclear, however, if their approach is applicable in practice. Tripakis and Yovine [18] show how to abstract dense real-time in order to obtain time-abstracting, finite bisimulations. Whenever it suffices to compute rather coarse abstractions, we expect to obtain much smaller transition systems by means of predicate abstraction and refinement of predicate abstractions.

The paper is structured as follows. In Section 2 we review the basic notions of timed automata including a natural semantics based on a nonconvergence assumption of time. We also define the notion of *restricted delay steps* and show that this restricted semantics of a timed automata is observationally equivalent to the natural semantics. The restricted semantics is used to define finite over- and under-approximations of timed systems in Section 3. In Section 4, we introduce the concept of a *basis* as a set of abstraction predicates expressive enough to distinguish between any two different clock regions, and we show that for predicate abstraction with a basis as abstraction predicates, the approximation is exact with respect to the next-free μ-calculus. Then, in Section 5, we define a terminating algorithm for iteratively refining abstractions until the given property is either proved or refuted. Finally, Section 6 contains some concluding remarks.

For lack of space we usually omit proofs, but detailed proofs can be found in an extended version of this paper [15].

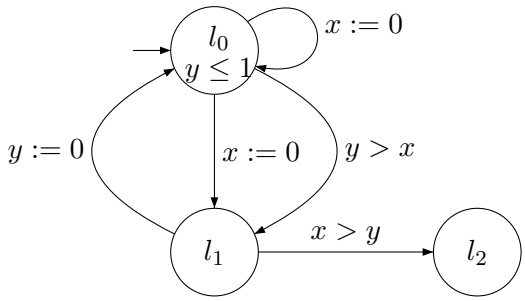


Fig. 1. Example of a Timed System.

2 Timed Systems

We review some basic notions of transition systems and timed systems. Furthermore, we introduce the notion of time-progressing systems, and show that delay steps in these systems are not observable in a version of the propositional μ-calculus without a next-step operator. These results set the stage for proving completeness of our abstraction techniques in Section 5.

The model of timed system as defined below is motivated by the timed automata model as introduced by Alur, Courcoubetis, and Dill [1].[3] Clocks for measuring time are encoded as variables, which are interpreted over the nonnegative reals $\mathbb{R}_0^+$. Transitions of timed systems are usually constrained by timing constraints.

Definition 2.1 [Timing Constraints] Given a set of clocks C, the set of *timing (or clock) constraints Constr* comprises **true**, $x \bowtie d$, 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 c, $Constr(c)$ is the finite subset of all timing constraints $x \bowtie d$, $x - y \bowtie d$, where $x, y \in C$, $\bowtie \in \{<, \leq, =, \geq, >\}$ and $d \in \{0, \ldots, c\}$.

Definition 2.2 [Timed Systems] Given a finite set of propositional symbols A, a *timed system* $\mathcal{S}$ is a tuple $\langle L, P, C, T, l_0, I\rangle$, where

- L is a nonempty finite set of locations,
- $P : L \to \wp(A)$ maps each location to a set of propositional symbols,
- C is a finite set of clocks,
- $T \subseteq L \times \wp(Constr) \times \wp(C) \times L$ is a transition relation,
- $l_0 \subseteq L$ is the initial location,
- and $I : L \to \wp(Inv)$ assigns a set of downward closed clock constraints to each location l; the elements of $I(l)$ are the *invariants* for location l.

We write $l \xrightarrow{g,r} l'$ for $\langle l, g, r, l'\rangle \in T$. Firing a transition does not only change

[3] For simplicity, we do not consider (synchronized) networks of timed automata. The results of this paper, however, can be extended for such networks.

the current location but also resets the clocks in r to 0. A transition may only be fired if the timing constraint (guard of the transition) g holds with respect to the current value of the clocks, and if the invariant of the target location is satisfied with respect to the modified value of the clocks.

Example 2.3 A timed system with three locations l_0, l_1, l_2 and two clocks x, y is displayed in Figure 1. The initial location is l_0, transitions are decorated with both timing constraints and clock resets such as $x := 0$. The invariant for location l_0 is $y \leq 1$. Timing constraints that are **true** are omitted.

A function $\nu : C \to I\!R_0^+$ is a *clock evaluation*, and the set of clock evaluations is collected in $\mathcal{V}_C$. The clock evaluation $(\nu + \delta)$ is obtained by adding δ to the value of each clock in ν. For $X \subseteq C$, $\nu[X{:=}0]$ denotes the clock evaluation 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 evaluation 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 $\nu \models\!\approx g$. A set $\mathcal{X} \subseteq \mathcal{V}_C$ of clock evaluations satisfies $g \in \mathit{Constr}$, written as $\mathcal{X} \models\!\approx g$, if and only if $\nu \models\!\approx g$ for all $\nu \in \mathcal{X}$. A pair $(l, \nu) \in L \times \mathcal{V}_C$ is called a *timed configuration*, if it satisfies the invariants $I(l)$; formally, $\nu \models\!\approx I(l)$ iff $\nu \models\!\approx g$ for every invariant $g \in I(l)$.

Alur, Courcoubetis, and Dill [1] introduce the fundamental notion of clock regions, which partition the space of possible clock evaluation for a timed automaton into finitely many regions.

Definition 2.4 [Clock Regions] Let $\mathcal{S}$ be a timed system with clocks C and largest constant c occurring in any timing constraint of $\mathcal{S}$. A *clock region* is a set $\mathcal{X} \subseteq \mathcal{V}_C$ of clock evaluations, such that for all timing constraints $g \in \mathit{Constr}(c)$ and for any two ν_1, $\nu_2 \in \mathcal{X}$ it is the case that $\nu_1 \models\!\approx g$ if and only if $\nu_2 \models\!\approx g$. In this case we write $\nu_1 \equiv_{\mathcal{S}} \nu_2$.

A *timed step* is either a *delay step*, where time advances by some positive real-valued δ, or an instantaneous *state transition step*.

Definition 2.5 [Timed Steps] Let $\mathcal{S}$ be a timed system with clock set C and transition relation T. For $\delta > 0$, we say that the timed configuration $(l, \nu{+}\delta)$ is obtained from (l, ν) by a *delay step* $(l, \nu) \xrightarrow{\delta} (l, \nu{+}\delta)$, if the invariant constraint $\nu + \delta \models\!\approx I(l)$ holds. A *state transition step* $(l, \nu) \xrightarrow{g,r} (l', \nu')$ occurs if there exists a $l \xrightarrow{g,r} l' \in T$, and $\nu \models\!\approx g$, $\nu' = \nu[r{:=}0]$, and $\nu' \models\!\approx I(l')$. The union of delay and state transition steps defines the timed transition relation $\Rightarrow$ of a timed system $\mathcal{S}$. Now, a *path* (or trace) is an infinite sequence of configurations $s_0 \Rightarrow s_1 \Rightarrow \ldots$.

Timed systems, as defined above, allow for infinite sequences of delay steps without ever exceeding some given bound. The sequence

$$(l, x = 0) \overset{1/2}{\Longrightarrow} (l, x = 1/2) \overset{1/4}{\Longrightarrow} (l, x = 3/4) \overset{1/8}{\Longrightarrow} (l, x = 7/8) \qquad \cdots \qquad (*)$$

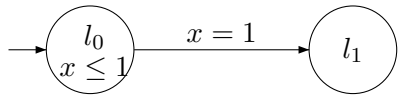


Fig. 2. Timed System for Example 2.6.

for example, never reaches time point 1. Systems with traces such that an infinite number of steps may happen in a bounded time frame are said to be *zeno*. This kind of behavior is usually ruled out by restricting possible behaviors to nonzeno only. In order to preserve faulty behavior that is caused by an infinite sequence of state transition steps, we use a slightly weaker assumption than nonzenoness. We only consider paths which satisfy the following assumption.

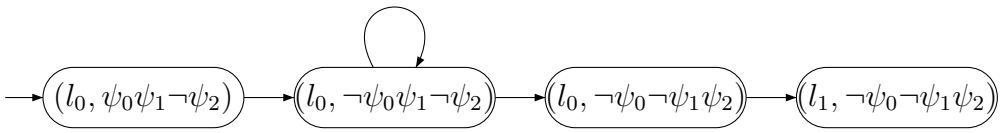
Assumption 1 (Nonconvergence of Time) In every infinite sequence of delay steps, the evaluation of every clock eventually exceeds every bound.

In the sequel we build time-abstractions which do not distinguish between state transition steps and delay steps. The main difficulty in defining such abstractions is to prevent delay steps to be abstracted into self-loops on the abstract system.

Example 2.6 Consider the timed system in Figure 2. Under the nonconvergence assumption this system satisfies the property that location l_1 is always reached. For example, the following sequence is the prefix of a possible trace of this system.

$$(l_0, x = 0) \stackrel{1/2}{\Longrightarrow} (l_0, x = 1/2) \stackrel{1/4}{\Longrightarrow} (l_0, x = 3/4) \stackrel{1/4}{\Longrightarrow} (l_0, x = 1) \stackrel{\mathbf{true},\emptyset}{\Longrightarrow} (l_1, x = 1)$$

We abstract the timed system from Figure 2 using the three abstraction predicates $\psi_0 \equiv x = 0$, $\psi_1 \equiv x < 1$, and $\psi_2 \equiv x = 1$. On the abstract system the single state transition step of the timed system is split according to whether or not these predicates hold. For example, in the initial abstract configuration only ψ_0 and ψ_1 hold, since the value of the clock in the initial concrete state is zero. Now, corresponding to delay steps with delay less than one, there is an abstract transition to a state where only ψ_1 holds. Using small enough delay steps one remains in this state or one reaches a state in which only ψ_2 holds, that is, the clock value is exactly one. A fragment of the resulting abstract transition system is given below.



Notice the self-loop at configuration $(l_0, \neg\psi_0\psi_1\neg\psi_2)$, which has not been present in the concrete system. For the presence of this loop, it no longer holds for the abstracted system that on every possible path a configuration with location l_1 is reached eventually.

In order to avoid such extraneous self-loops, the nonconvergence assump-

tion must somehow be incorporated into the abstract system. Such a restriction, however, can not be defined by means of time delays in the abstract system for the simple reason that there is no notion of time or time delay on this level. In our approach, we enforce the nonconvergence assumption explicitly by restricting the model of timed system to delay steps that force a clock to step beyond integer bounds when all fractional clock values are not zero. In this way, the second and third delay step of the trace $(*)$ above, for example, are explicitly ruled out.

Definition 2.7 [Restricted Delay Step] For a timed system $\mathcal{S}$ with clock set C and largest constant c, a *restricted delay step* is a delay step $(l, \nu) \xrightarrow{\delta} (l, \nu + \delta)$ for all positive, real-valued δ, such that

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

The union of state transition steps and restricted delay steps gives rise to a relation $\Rightarrow_R \subseteq (L, \mathcal{V}_C) \times (L, \mathcal{V}_C)$. Now, a *restricted path* is an infinite sequence of configurations $s_0 \Rightarrow_R s_1 \Rightarrow_R \ldots$.

Obviously, it is the case that $\Rightarrow_R$ is a subrelation of $\Rightarrow$. However, the restriction of delay steps above does not necessarily enforce time to progress, as demonstrated by the following restricted path for the system in Example 2.3.

$$(l_0, x = y = 0) \overset{\mathbf{true},\varnothing}{\Longrightarrow} (l_1, x = y = 0) \overset{\mathbf{true},\varnothing}{\Longrightarrow} (l_0, x = y = 0) \overset{\mathbf{true},\varnothing}{\Longrightarrow} (l_1, x = y = 0) \cdots$$

Note that a loop of state transition steps is required in order to prevent the clocks x and y from exceeding the clock value 0.

Corresponding to the nonconvergence assumption on timed traces and the restricted delay steps we associate two semantics for timed systems in terms of transition systems. The natural semantics $\mathcal{M}$ includes arbitrary delay steps under the nonconvergence of time assumption, while the restricted semantics $\mathcal{M}_R$ includes only restricted delay steps as in Definition 2.7.

Definition 2.8 [Semantics of Timed Systems] Let $\mathcal{S} = \langle L, P, C, T, l_0, I \rangle$ be a timed system. We associate with $\mathcal{S}$ two transition systems

$$\mathcal{M} := \langle L \times \mathcal{V}_C, P, (\Rightarrow), (l_0,\ \nu_0) \rangle$$
$$\mathcal{M}_R := \langle L \times \mathcal{V}_C, P, (\Rightarrow_R), (l_0,\ \nu_0) \rangle$$

The symbol ν_0 denotes the clock evaluation, that maps every clock to 0. $\mathcal{M}$ is called the *natural semantics*, and $\mathcal{M}_R$ is referred to as the *restricted semantics* of $\mathcal{S}$.

We demonstrate that the restriction of delay steps does not change the possible observations of the model with respect to μ-calculus formulas without next-step operators.

2.1 Definition of Next-Free μ-Calculus

The μ-calculus [13] is a branching-time temporal logic, where formulas are built from atomic propositions, boolean connectives, the least-fixpoint opera-

tor, and the next-step operator $\bigcirc\varphi$, which expresses the fact that there is a successor satisfying φ. Our main interest in removing the next-step operator stems from the fact that we do not want to distinguish between one delay step of duration, say, 1 and two subsequent delay steps of durations 2/5 and 3/5, since these traces are considered to be observationally equivalent. Logics without explicit next-step operator have also been considered, for example, by Dams [6] and Tripakis and Yovine [18].

Definition 2.9 [Next-Free μ-Calculus] Let A be a set of atomic predicates, and *Var* be a set of variables; then, for $p \in A$ and $Z \in \mathit{Var}$, the set $\mathcal{L}_\mu$ of next-free μ-calculus formulas is described by the grammar

$$\varphi ::= \mathit{tt} \mid p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \exists\,(\varphi\, U \varphi) \mid \forall\,(\varphi\, U \varphi) \mid Z \mid \mu Z.\varphi$$

Formulas are assumed to be syntacticaly monotonic, that is, every variable is assumed to appear under an even number of negations, in order to guarantee the existence of the fixpoints under consideration. A *sentence* is a formula without free variables.

Intuitively, an *existential until* formula $\exists\,(\varphi_1\, U \varphi_2)$ holds in some configuration s iff φ_1 holds until φ_2 holds on some path starting from s. Similarly, a *universal until* formula $\forall\,(\varphi_1\, U \varphi_2)$ holds in s if this conditions holds for all paths from s.

Given a transition system $\mathcal{M} = \langle S, P, \Rightarrow_R, s_0\rangle$, the semantics of a next-free μ-calculus sentence is given by the set of timed configurations $[\![\varphi]\!]^{\mathcal{M}}_{\vartheta}$ for which the formula holds. Subformulas containing free variables $Z \in \mathit{Var}$ are dealt with using *valuation functions* $\vartheta : \mathit{Var} \to \wp\,(S)$. The updating notation $\vartheta[Z := s]$ denotes the valuation ϑ' that agrees with ϑ on all variables except Z, where $\vartheta'(Z) = s \subseteq S$.

Definition 2.10 [Semantics of the Next-Free μ-Calculus] Given a transition system $\mathcal{M} = \langle S, P, \Rightarrow_R, s_0\rangle$ over the set $S = L \times \mathcal{V}_C$ of timed configurations and an assignment $\vartheta : \mathit{Var} \to \wp\,(S)$, the set of configurations $[\![\varphi]\!]^{\mathcal{M}}_{\vartheta}$ validating a formula $\varphi \in \mathcal{L}_\mu$ with respect to ϑ is defined inductively on the structure of φ.

$$\begin{aligned}
[\![\mathit{tt}]\!]^{\mathcal{M}}_{\vartheta} &:= S \\
[\![p]\!]^{\mathcal{M}}_{\vartheta} &:= \{(l,\nu) \in S \mid p \in P(l)\} \\
[\![\varphi_1 \wedge \varphi_2]\!]^{\mathcal{M}}_{\vartheta} &:= [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta} \;\cap\; [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta} \\
[\![\neg\varphi]\!]^{\mathcal{M}}_{\vartheta} &:= S \setminus [\![\varphi]\!]^{\mathcal{M}}_{\vartheta} \\
[\![\exists\,(\varphi_1\, U \varphi_2)]\!]^{\mathcal{M}}_{\vartheta} &:= \{s \in S \mid \text{for some path } \tau = (s_0 \Rightarrow s_1 \Rightarrow \ldots) \\
&\qquad \text{with } s_0 = s, \text{ for some } i \geq 0, s_i \in [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta} \\
&\qquad \text{and } s_j \in [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta} \text{ for } 0 \leq j < i\} \\
[\![\forall\,(\varphi_1\, U \varphi_2)]\!]^{\mathcal{M}}_{\vartheta} &:= \{s \in S \mid \text{for every path } \tau = (s_0 \Rightarrow s_1 \Rightarrow \ldots) \\
&\qquad \text{with } s_0 = s, \text{ for some } i \geq 0, s_i \in [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta} \\
&\qquad \text{and } s_j \in [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta} \text{ for } 0 \leq j < i\}
\end{aligned}$$

$$[\![Z]\!]^{\mathcal{M}}_{\vartheta} := \vartheta(Z)$$
$$[\![\mu Z.\varphi]\!]^{\mathcal{M}}_{\vartheta} := \cap\{\mathcal{E} \subseteq S \mid [\![\varphi]\!]^{\mathcal{M}}_{\vartheta[Z:=\mathcal{E}]} \subseteq \mathcal{E}\}$$

We write $\mathcal{M}, s, \vartheta \models \varphi$ to denote that $s \in [\![\varphi]\!]^{\mathcal{M}}_{\vartheta}$. The subscript ϑ is omitted whenever φ is a sentence.

Two configurations s and s' are said to be indistinguishable if they satisfy the same set of $\mathcal{L}_\mu$ sentences.

Definition 2.11 [μ-Equivalence] For a transition system $\mathcal{M}$, two configurations s, s' are *μ-equivalent* denoted by $s \equiv_{\mathcal{M}} s'$, if for every sentence $\varphi \in \mathcal{L}_\mu$: $s \in [\![\varphi]\!]^{\mathcal{M}}$ if and only if $s' \in [\![\varphi]\!]^{\mathcal{M}}$.

The binary relation $\equiv_{\mathcal{M}}$ is indeed an equivalence relation on clock evaluations. Moreover, μ-equivalence characterizes clock regions in the sense that two clock evaluations are in the same clock region if and only if they are μ-equivalent. Consequently, μ-equivalence is of finite index.

Lemma 2.12 Let $\mathcal{S}$ be a timed system with clock set C and largest constant c, and let $\mathcal{M}$ be the corresponding natural transition system. Then for all $l \in L$ and clock evaluations $\nu, \nu' \in \mathcal{V}_C$ with $\nu \equiv_{\mathcal{S}} \nu'$ the time configurations (l, ν) and (l, ν') are μ-equivalent, that is $(l, \nu) \equiv_{\mathcal{M}} (l, \nu')$.

The proof works by a straightforward structural induction on φ. We now show that the natural semantics and the restricted semantics of a timed system as introduced in Definition 2.8 are indistinguishable in the next-free μ-calculus. Intuitively, sentences in $\mathcal{L}_\mu$ can not distinguish quantitative values of clocks, and therefore all configurations with identical control locations and μ-equivalent clock evaluations satisfy the same set of $\mathcal{L}_\mu$ sentences.

Theorem 2.13 Let $\mathcal{S}$ be a timed system with clocks C, largest constant c, natural semantics $\mathcal{M}$, and restricted semantics $\mathcal{M}_R$. Under the nonconvergence assumption for $\mathcal{M}$, for every sentence $\varphi \in \mathcal{L}_\mu$:

$$[\![\varphi]\!]^{\mathcal{M}} \quad = \quad [\![\varphi]\!]^{\mathcal{M}_R}$$

Proof. Again, the proof is by induction on φ. The only interesting cases are the ones for the until formulas. Thus, consider φ to be of the form $\exists\,(\varphi_1\, U\, \varphi_2)$. According to Definition 2.10, $s \in [\![\exists\,(\varphi_1\, U\, \varphi_2)]\!]^{\mathcal{M}}_{\vartheta}$ iff there exists an path starting at s such that

$$s_i \in [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta} \text{ for some } i \geq 0, \text{and for all } 0 \leq j < i,\ s_j \in [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta} \tag{2}$$

Since every path in the restricted semantics is also a path in the natural semantics, it suffices to show that for every path in the natural semantics which validates (2), there exists a path in the restricted semantics which also validates (2). First, we show that a delay step in $\Rightarrow \setminus \Rightarrow_R$ does not step across the border of any region. Let $(l, \nu) \xrightarrow{\delta} (l, \nu + \delta)$ be a delay step in $\mathcal{M}$ but not

in $\mathcal{M}_R$. Then by Definition 2.7:

$$\begin{aligned}
&\neg\,(\exists x \in C.\ \exists k \in \{0,\ldots,c\}.\ \nu(x) = k \ \vee\ (\nu(x) < k \wedge \nu(x) + \delta \geq k)) \\
\Leftrightarrow\ &\forall x \in C.\ \forall k \in \{0,\ldots,c\}.\ (\nu(x) \neq k \ \wedge\ (\nu(x) < k \Rightarrow \nu(x) + \delta < k)) \\
\Leftrightarrow\ &\forall x \in C.\ \lfloor \nu(x) \rfloor \ <\ \nu(x),\ \nu(x) + \delta \ <\ \lfloor \nu(x) + 1 \rfloor
\end{aligned}$$

Consequently, it is the case that $\nu \equiv_{\mathcal{S}} (\nu + \delta)$. Using Lemma 2.12, for $(s, s') \in \Rightarrow \setminus \Rightarrow_R$ it holds that $s \equiv_{\mathcal{M}} s'$. Now, consider a finite path $\tau = (s_1 \Rightarrow \ldots \Rightarrow s_i)$ with $s_i \in [\![\varphi_2]\!]_\vartheta^{\mathcal{M}}$ and $\forall\, 1 \leq j < i.\ s_j \in [\![\varphi_1]\!]_\vartheta^{\mathcal{M}}$. We transform this path τ to a restricted path τ_R by removing the steps not contained in $\Rightarrow_R$ and by merging adjacent delays. Using Lemma 2.12, all $s_{e+f} = (l_{e+f}, \nu_{e+f})$ with $l_{e+f} = l_e$ and $\nu_{e+f} \equiv_{\mathcal{S}} \nu_e$, are μ-equivalent, that is, $(l_{e+f}, \nu_{e+f}) \equiv_{\mathcal{M}} (l_e, \nu_e)$. Removing all s_{e+f} with $f \geq 1$ from τ yields the subpath

$$\tau_R = (s_1 = s_{k_1} \Rightarrow_R s_{k_2} \cdots \Rightarrow_R s_{k_m} = s_i),\ k_h \in \{1,\ldots,i\},\ k_h < k_{h+1}$$

such that $s_{k_m} \in [\![\varphi_2]\!]_\vartheta^{\mathcal{M}}$ and for all $h < m$, $s_{k_h} \in [\![\varphi_1]\!]_\vartheta^{\mathcal{M}}$. By induction hypothesis, $s_{k_m} \in [\![\varphi_2]\!]_\vartheta^{\mathcal{M}_R}$ and for all $h < m$, $s_{k_h} \in [\![\varphi_1]\!]_\vartheta^{\mathcal{M}_R}$. Since both guards and invariants are timing constraints in *Constr*, they have identical truth values for the clock evaluations of s_e and s_{e+f}. Thus every step $s_{k_1} \Rightarrow_R s_{k_2}$ is indeed possible according to the restricted semantics, and τ_R is a restricted path. Thus $[\![\exists\,(\varphi_1\, U\, \varphi_2)]\!]_\vartheta^{\mathcal{M}} = [\![\exists\,(\varphi_1\, U\, \varphi_2)]\!]_\vartheta^{\mathcal{M}_R}$. The proof for universal untils is similar. □

This result allows us to focus on the restricted semantics of timed systems only, since any result expressible in $\mathcal{L}_\mu$ for the restricted semantics $\mathcal{M}_R$ also holds for the natural semantics $\mathcal{M}$. In the sequel we omit the indices R; thus the system $\mathcal{M}$ and the transition relation $\Rightarrow$ denote a restricted system and a restricted transition relation, respectively.

3 Predicate Abstraction of Timed Systems

Predicate abstraction [10,3,17] is used to compute a finite approximation of a given infinite state transition system. The method is based on a set of abstraction predicates, which in our context are predicates over clock evaluations.

Definition 3.1 [Abstraction Predicates] 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 evaluation 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 tt, we write $\nu \models\!\!\approx \psi$.

A set of abstraction predicates $\Psi = \{\psi_0, \cdots, \psi_{n-1}\}$ determines an abstraction function α, which maps clock evaluations 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 which 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 3.2 [Abstraction/Concretization] Let C be a set of clocks and $\mathcal{V}_C$ the corresponding set of clock evaluations. 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* $\gamma : L \times B_n \to L \times \wp(\mathcal{V}_C)$ 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 3.3 [Over-/Under-approximation] Given a (concrete) transition system $\mathcal{M} = \langle S^C, P, \Rightarrow, s_0^C \rangle$, where $S^C = L \times \mathcal{V}_C$ and $s_0^C = (l_0, \nu_0)$, and a set Ψ of abstraction predicates, we construct two (abstract) transition systems $\mathcal{M}_\Psi^+ = \langle S^A, P, \Rightarrow^+, s_0^A \rangle$, and $\mathcal{M}_\Psi^- = \langle S^A, P, \Rightarrow^-, s_0^A \rangle$.

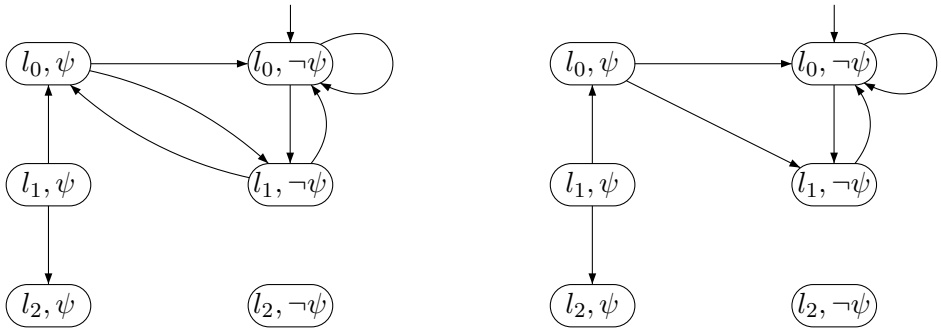
- $S^A := L \times B_n$
- $(l, b) \Rightarrow^+ (l', b')$ iff
  $\exists \nu, \nu' \in \mathcal{V}_C$ s.t. $(l, \nu) \in \gamma(l, b) \ \wedge \ (l', \nu') \in \gamma(l', b')$. $(l, \nu) \Rightarrow (l', \nu')$
- $(l, b) \Rightarrow^- (l', b')$ iff
  $\forall \nu \in \mathcal{V}_C$ s.t. $(l, \nu) \in \gamma(l, b)$. $\exists \nu' \in \mathcal{V}_C$ s.t. $(l', \nu') \in \gamma(l', b')$. $(l, \nu) \Rightarrow (l', \nu')$.
- $s_0^A := (l_0, b_0)$, where $b_0(i) = 1$ 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 $\Rightarrow^- \subseteq \Rightarrow^+$.

Example 3.4 Figure 3 shows the over- and under-approximation of the (concrete) system from Figure 1 with respect to the predicate set $\Psi = \{x > y\}$. The initial state is described by $(l_0, \neg\psi)$, since in the initial location the value of the clocks x and y is zero, and therefore $x \leq y$ holds. The transitions are built according to definition 3.2. For example, the transition from $(l_1, \neg\psi)$ to (l_0, ψ) in the over-approximation is not present in the under-approximation, since for the evaluation ν with $\nu(x) = 0$ and $\nu(y) = 0$ and $(l_1, \nu) \in \gamma(l_1, \neg\psi)$, there exists no evaluation ν' with $(l_0, \nu') \in \gamma(l_0, \psi)$ such that $(l_1, \nu) \Rightarrow (l_0, \nu')$.

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



a: Over-approximation b: Under-approximation

Fig. 3. Approximations of the timed system from Figure 1 with $\psi \equiv x > y$.

$\gamma(\Rightarrow^+)$ as follows:

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

The next statement follows directly from Definition 3.3.

Lemma 3.5 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 $\Rightarrow^+$, $\Rightarrow^-$ it is the case that

(i) $\gamma(\Rightarrow^-) \subseteq \Rightarrow \subseteq \gamma(\Rightarrow^+)$, and

(ii) $\Rightarrow^- \subseteq \alpha(\Rightarrow) \subseteq \Rightarrow^+$.

Definition 3.6 [Predicate Abstraction] Let $\mathcal{M} = \langle S^C, P, \Rightarrow, s_0^C\rangle$ be a transition system, and Ψ be a set of abstraction predicates. Consider, as given in Definition 3.3, the over-approximation $\mathcal{M}^+_\Psi = \langle S^A, P, \Rightarrow^+, s_0^A\rangle$, and the under-approximation $\mathcal{M}^-_\Psi = \langle S^A, P, \Rightarrow^-, s_0^A\rangle$ of $\mathcal{M}$. Then, the *predicate abstracted* semantics $[\![\varphi]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi}$, where σ is either $+$ or $-$, of a formula $\varphi \in \mathcal{L}_\mu$ with respect to a valuation function ϑ and the finite transition systems $\mathcal{M}^\sigma_\Psi$ is defined in a mutually inductive way. The notation $\bar\sigma$ is used to toggle the sign σ.

$$\begin{aligned}[\![tt]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi} &:= S^A\\ [\![p]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi} &:= \big\{(l,b) \in S^A \mid p \in P(l)\big\}\\ [\![\varphi_1 \wedge \varphi_2]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi} &:= [\![\varphi_1]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi} \cap [\![\varphi_2]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi}\\ [\![\neg\varphi]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi} &:= S^A \setminus [\![\varphi]\!]_\vartheta^{\mathcal{M}^{\bar\sigma}_\Psi}\\ [\![\exists\,(\varphi_1\, U\, \varphi_2)]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi} &:= \{s \in S^A \mid \text{for some path } \tau = (s_0 \Rightarrow^\sigma s_1 \Rightarrow^\sigma \ldots)\\ &\qquad \text{with } s_0 = s, \quad \text{for some } i \geq 0, s_i \in [\![\varphi_2]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi}\\ &\qquad \text{and } s_j \in [\![\varphi_1]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi} \text{ for } 0 \leq j < i\}\\ [\![\forall\,(\varphi_1\, U\, \varphi_2)]\!]_\vartheta^{\mathcal{M}^\sigma_\Psi} &:= \{s \in S^A \mid \text{for every path } \tau = (s_0 \Rightarrow^{\bar\sigma} s_1 \Rightarrow^{\bar\sigma} \ldots)\end{aligned}$$

$$\begin{aligned}
&\text{with } s_0 = s, \quad \text{for some } i \geq 0, s_i \in [\![\varphi_2]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} \\
&\text{and } s_j \in [\![\varphi_1]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} \text{ for } 0 \leq j < i\} \\
[\![Z]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} &:= \vartheta(Z) \\
[\![\mu Z.\varphi]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} &:= \cap\{\mathcal{E} \subseteq S^A \mid [\![\varphi]\!]_{\vartheta[Z:=\mathcal{E}]}^{\mathcal{M}_\Psi^\sigma} \subseteq \mathcal{E}\}
\end{aligned}$$

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

Theorem 3.7 (Soundness of Abstraction) Let $\mathcal{M} = \langle S^C, P, \Rightarrow, s_0^C \rangle$ be a transition system, Ψ a set of abstraction predicates, and $\mathcal{M}_\Psi^+$, $\mathcal{M}_\Psi^-$ the over-approximation and under-approximation of $\mathcal{M}$ with respect to Ψ. Then for any sentence $\varphi \in \mathcal{L}_\mu$ the following holds (γ denotes the concretization function with respect to Ψ):

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

Proof. The proof follows by induction on the structure of φ and makes use of Lemma 3.5. □

Example 3.8 Consider our running example in Figure 1, for which we want to verify that location l_2 is never reached. This property is expressed by the μ-calculus formula $\varphi := \neg\exists\,(tt\, U\, at_l_2)$, where $at_l_2 \in A$ is a (boolean) proposition that is true if the system is in location l_2. According to Definition 3.6, the set of abstract states of $\mathcal{M}_{\{\psi\}}^-$ which validate φ is given by

$$\begin{aligned}
[\![\neg\exists\,(tt\, U\, at_l_2)]\!]^{\mathcal{M}_{\{\psi\}}^-} &= S^A \setminus [\![\exists\,(tt\, U\, at_l_2)]\!]^{\mathcal{M}_{\{\psi\}}^+} = \\
&= \{(l_0, \psi), (l_0, \neg\psi), (l_1, \neg\psi)\}
\end{aligned}$$

The over-approximation of $\mathcal{M}$ with respect to the abstraction predicate $\psi \equiv (x > y)$ is shown in Figure 3. Since the initial state $(l_0, \neg\psi)$ of $\mathcal{M}_{\{\psi\}}^-$ is contained in the set $[\![\neg\exists\,(tt\, U\, at_l_2)]\!]^{\mathcal{M}_{\{\psi\}}^-}$, the formula φ holds on the abstract transition system. Thus, $\mathcal{M}_{\{\psi\}}^-, (l_0, b_0) \models^A \varphi$ holds. By Theorem 3.7, property φ also holds on the concrete transition system, $\mathcal{M}, (l_0, \nu_0) \models \varphi$.

An interesting aspect of this example is that $[\![\varphi]\!]^{\mathcal{M}_{\{\psi\}}^-} = [\![\varphi]\!]^{\mathcal{M}_{\{\psi\}}^+}$. We now give a criterion, based on the notion of regions, for a set of abstraction predicates, which is sufficient for guaranteeing convergence of the over- and under-approximations in general.

4 Basis

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 next-free μ-calculus.

Definition 4.1 [Basis] Let $\mathcal{S}$ be a timed system 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 evaluations ν_1, $\nu_2 \in \mathcal{V}_C$

$$(\forall \psi \in \Psi.\ \nu_1 \models\!\approx \psi \Leftrightarrow \nu_2 \models\!\approx \psi) \ \text{ implies } \ \nu_1 \equiv_{\mathcal{S}} \nu_2 \ .$$

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

Example 4.2 The set Ψ:=$\{x = 0,\ y = 0,\ x \leq 1,\ x \geq 1,\ y \leq 1,\ y \geq 1,\ x > y,\ x < y\}$ is a basis for the timed system in Figure 1.

Theorem 4.3 Let $\mathcal{S}$ be a timed system with clock set C and largest constant c, and $\mathcal{M}$ the corresponding transition system. Let Ψ be a basis with respect to $\mathcal{S}$, and $\mathcal{M}^-_\Psi$, $\mathcal{M}^+_\Psi$ the under- and over-approximation of $\mathcal{M}$ with respect to Ψ. Then, for any sentence $\varphi \in \mathcal{L}_\mu$,

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

Proof. Since it suffices to show that $\Rightarrow^- \supseteq \Rightarrow^+$, we assume two configurations (l, b) and (l', b') such that $(l, b) \Rightarrow^+ (l', b')$. According to Definition 3.3, there exist $\nu, \nu' \in \mathcal{V}_C$ such that $(l, \nu) \in \gamma(l, b)$ and $(l', \nu') \in \gamma(l', b')$ such that $(l, \nu) \Rightarrow (l', \nu')$.

First, in case $(l, \nu) \Rightarrow (l', \nu')$ holds due to a state transition step, the guards g of some transition $l \xrightarrow{g,r} l'$ are satisfied by the current clock evaluation ν, that is, $\nu \models\!\approx g$. Since Ψ is a basis, for all clock evaluations $\tilde{\nu} \in \mathcal{V}_C$ with $(l, \tilde{\nu}) \in \gamma(l, b)$ it follows by Definition 4.1 that $\nu \equiv_{\mathcal{S}} \tilde{\nu}$, and therefore by Definition 2.4, $\tilde{\nu} \models\!\approx g$. Thus, for all clock evaluations $\tilde{\nu}$ that satisfy the above conditions, the state transition $l \xrightarrow{g,r} l'$ is possible. Therefore, for all $\tilde{\nu} \in \mathcal{V}_C$ with $(l, \tilde{\nu}) \in \gamma(l, b)$ exists $\nu' \in \mathcal{V}_C$ with $(l', \nu') \in \gamma(l', b')$ such that $(l, \tilde{\nu}) \Rightarrow (l', \nu')$. By Definition 3.3 it follows that $(l, b) \Rightarrow^- (l', b')$.

Second, if $(l, \nu) \Rightarrow (l', \nu')$ holds due to a delay step, then $l = l'$ and $\nu' \models\!\approx I(l)$, for $(l, \nu') \in \gamma(l, b')$. Since $I(l) \in Inv$, and *Inv* is a basis, by Definition 4.1 it follows that for all clock evaluations $\tilde{\nu} \in \mathcal{V}_C$ with $(l, \tilde{\nu}) \in \gamma(l, b)$, $\nu \equiv_{\mathcal{S}} \tilde{\nu}$, and thus by Definition 2.4, $\tilde{\nu} \models\!\approx I(l)$. Since $(l, \nu) \Rightarrow (l, \nu')$, according to the restriction of delay steps (Definition 2.7) ν and ν' are not in the same region. Therefore, at location l there exists for all $\tilde{\nu} \in \mathcal{V}_C$ with $(l, \tilde{\nu}) \in \gamma(l, b)$ a delay step to some $\nu' \in \mathcal{V}_C$ with $(l, \nu') \in \gamma(l, b')$. Again, by Definition 3.3 it follows that $(l, b) \Rightarrow^- (l', b')$. □

Corollary 4.4 (Basis Completeness) Let $\mathcal{S} = \langle L, P, C, T, l_0, I \rangle$ be a timed system, $\mathcal{M} = \langle L \times \mathcal{V}_C, P, \Rightarrow, (l_0, \nu_0) \rangle$ the corresponding transition system, let Ψ be a basis for $\mathcal{S}$, and let $\gamma(l_0, b_0) = (l_0, \nu_0)$. Then for any sentence $\varphi \in \mathcal{L}_\mu$:

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

This follows directly from Theorems 3.7 and 4.3.

5 Refinement of the Abstraction

Given a concrete model $\mathcal{M}$ of a timed system, with initial state s_0, a finite basis Ψ of abstraction predicates, and a formula φ, we present an algorithm for computing an over-approximation of $\mathcal{M}$ that is sufficient to prove or refute the model checking problem $\mathcal{M} \models \varphi$. This over-approximation is based on a subset of the basis predicates and is computed using stepwise refinement. The abstraction-refinement algorithm is displayed in Figure 4. The variables Ψ_{new} and Ψ_{act} store the currently unused and used abstraction predicates, respectively. Initially Ψ_{act} contains a subset Ψ' of predicates from the basis, and Ψ_{new} contains the remaining predicates (lines (2) and (3) in Figure 4). First it is checked if $s_0 \in \gamma([\![\varphi]\!]^{\mathcal{M}^-_{\Psi_{act}}})$ by calling a finite-state μ-calculus model checker. If indeed the under-approximation satisfies φ, then, by Corollary 4.4 $\mathcal{M}$ also satisfies φ and the algorithm returns **true** (line (5)). As next, we check if $s_0 \notin \gamma([\![\varphi]\!]^{\mathcal{M}^+_{\Psi_{act}}})$. If the over-approximation does not satisfy φ, then, also by Corollary 4.4, $\mathcal{M}$ does not satisfy φ and the algorithm returns **false** (line (6)). Otherwise (line (7)), that is, $s_0 \notin \gamma([\![\varphi]\!]^{\mathcal{M}^-_{\Psi_{act}}})$ and $s_0 \in \gamma([\![\varphi]\!]^{\mathcal{M}^+_{\Psi_{act}}})$, the μ-calculus model checker returns a counterexample in the form of an abstract path (see [12]) $q_0 \Rightarrow^+ q_1 \Rightarrow^+ \cdots \Rightarrow^+ q_n$, where q_0 is the initial state of $\mathcal{M}^+_\psi$. If for the abstract path, there exists a corresponding path in the concrete transition system, then we get a counterexample for the concrete model checking problem (lines (8)-(10)). In this case the algorithm returns **false**. This check requires an off-the-shelf satisfiability-checker for the boolean combination of linear arithmetic constraints such as ICS [9]. In case the abstract counterexample is spurious, there exists a smallest index k and a concrete path $y_0 \Rightarrow \cdots \Rightarrow y_k$, where y_0 is the initial location of $\mathcal{M}$, and for all $i \in \{0, \cdots, k\}$, $y_i \in \gamma(q_i)$, such that there is no (concrete) transition from y_k to y_{k+1}, where $y_{k+1} \in \gamma(q_{k+1})$ (lines (11)-(14)). We choose a minimal set of new abstraction predicates from Ψ_{new} such that the transition from q_k to q_{k+1} is eliminated (lines (15)-(18)). This new set of abstraction predicates is selected in such a way that the formula

$$\exists\, y_1, y_2 \in S^C.\, y_1 \in \gamma(q_k) \;\wedge\; y_2 \in \gamma(q_{k+1}) \;\wedge\; y_1 \not\Rightarrow y_2$$

holds. Notice that the concretization function γ actually depends on the current set Ψ_{act} of abstraction predicates.

Theorem 5.1 (Termination, Soundness, and Completeness) Let $\mathcal{M}$ be a transition system with a corresponding finite basis Ψ, and φ a sentence in $\mathcal{L}_\mu$. Then the algorithm in Figure 4 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. Follows directly from the finiteness of the basis and Theorems 4.3 and 3.7. □

Algorithm: *abstract_and_refine*

input: $\mathcal{M}$, initial state s_0, φ, basis Ψ

output: answer to model checking query $\mathcal{M} \stackrel{?}{\models} \varphi$

(1) **choose** $\Psi' = \{\psi_1, \ldots, \psi_i\}$ from Ψ;
(2) $\Psi_{new} := \Psi \setminus \Psi'$;
(3) $\Psi_{act} := \Psi'$;
(4) **loop**
(5) **if** $s_0 \in \gamma([\![\varphi]\!]^{\mathcal{M}^-_{\Psi_{act}}})$ **then return** true
(6) **else if** $s_0 \notin \gamma([\![\varphi]\!]^{\mathcal{M}^+_{\Psi_{act}}})$ **then return** false
(7) **else let** $(q_0 \Rightarrow^+ q_1 \cdots \Rightarrow^+ q_n)$ be a counterexample in $\mathcal{M}^+_{\Psi_{act}}$
(8) **if** there exists a path $\tau = (y_0 \Rightarrow y_1 \cdots \Rightarrow y_n)$
(9) such that $y_0 = s_0$ and $y_i \in \gamma(q_i)$ for all $0 \leq i \leq n$
(10) **then return** false
(11) **else let** k such that $\exists$ a path $\tau = (y_0 \Rightarrow y_1 \cdots \Rightarrow y_k)$
(12) where $y_0 = s_0$ and
(13) $y_i \in \gamma(q_i)$ for all $0 \leq i \leq k$ and
(14) $\forall\, y_{k+1} \in \gamma(q_{k+1}).\, y_k \not\Rightarrow y_{k+1}$;
(15) **choose** minimal $\Psi' = \{\psi_1, \ldots, \psi_i\}$ from Ψ_{new} such that
(16) $\forall\, y_1 \in \gamma(q_k), y_2 \in \gamma(q_{k+1}).\, y_1 \not\Rightarrow y_2$;
(17) $\Psi_{act} := \Psi_{act} \cup \{\Psi'\}$;
(18) $\Psi_{new} := \Psi_{new} \setminus \Psi'$
(19) **endif**
(20) **endif**
(21) **endloop**

Fig. 4. Iterative Abstraction-Refinement Algorithm.

Example 5.2 Consider again the timed system from Figure 1, and the formula $\varphi := \neg\exists\,(tt\, U\, at_l_2)$ which describes the property that location l_2 is never reached. A given basis for this system is Ψ:=$\{x = 0,\, y = 0,\, x \leq 1,\, x \geq 1,\, y \leq 1,\, y \geq 1,\, x > y,\, x < y\}$. The transition system of the initial over-approximation with the single abstraction predicate $\psi_0 \equiv x = 0$ is shown in Figure 5.

Model checking the formula φ on the transition system $\mathcal{M}^-_{\{x=0\}}$ returns **false**, since $s_0 = (l_0, x = y = 0) \notin \gamma([\![\varphi]\!]^{\mathcal{M}^-_{\{x=0\}}})$. The algorithm returns the counterexample $(l_0, \psi_0) \Rightarrow^+ (l_1, \psi_0) \Rightarrow^+ (l_1, \neg\psi_0) \Rightarrow^+ (l_2, \neg\psi_0)$, which is emphasized in Figure 5 using lines in bold face. The concretizations of the states on this abstract path are as follows. To simplify the notation we denote sets of configurations such as $\{(l, \nu) \mid l = l_1 \wedge \nu(x) = 0 \wedge \nu(y) \geq 0\}$ by $(l_1,\, x = 0 \wedge y \geq 0)$.

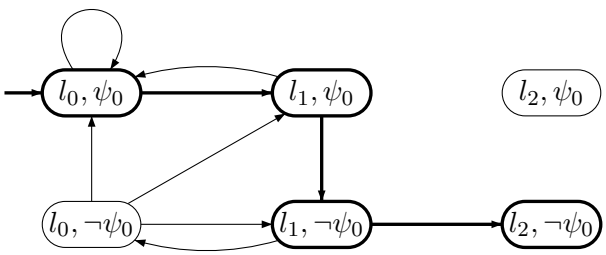


Fig. 5. Over-approximation of the timed system from Figure 1 with $\psi_0 \equiv x = 0$.

$$\begin{aligned}
\gamma(q_0) = \gamma(l_0, \psi_0) &= (l_0,\ x = 0 \ \wedge\ y \geq 0) \\
\gamma(q_1) = \gamma(l_1, \psi_0) &= (l_1,\ x = 0 \ \wedge\ y \geq 0) \\
\gamma(q_2) = \gamma(l_1, \neg\psi_0) &= (l_1,\ x > 0 \ \wedge\ y \geq 0) \\
\gamma(q_3) = \gamma(l_2, \neg\psi_0) &= (l_2,\ x > 0 \ \wedge\ y \geq 0)
\end{aligned}$$

Now we have to check if there is a corresponding counterexample on the concrete transition system, that is, if there exists a path $y_0 \Rightarrow y_1 \Rightarrow y_2 \Rightarrow y_3$, where y_0, y_1, y_2, $y_3 \in S^C$, such that $y_0 \in \gamma(q_0)$, $y_1 \in \gamma(q_1)$, $y_2 \in \gamma(q_2)$, $y_3 \in \gamma(q_3)$, and $y_0 = s_0$. This is the case if the formula

$$\begin{aligned}
F_1 := \exists\, y_0, y_1, y_2, y_3 \in S^C.\, y_0 \in \gamma(q_0) \ \wedge\ y_1 \in \gamma(q_1) \ \wedge\ y_2 \in \gamma(q_2) \ \wedge \\
y_3 \in \gamma(q_3) \ \wedge\ y_0 \Rightarrow y_1 \ \wedge\ y_1 \Rightarrow y_2 \ \wedge\ y_2 \Rightarrow y_3 \ \wedge\ y_0 = s_0
\end{aligned}$$

is valid. In our example F_1 is unsatisfiable, since on the concrete transition system there is no transition between y_2 and y_3, as it is illustrated by the path

$$\underbrace{(l_0, x = y = 0)}_{\ni y_0} \Rightarrow \underbrace{(l_1, x = 0 \wedge 0 \leq y \leq 1)}_{\ni y_1} \Rightarrow \underbrace{(l_1, x > 0 \wedge x \leq y)}_{\ni y_2}$$

Thus, $k = 2$ in our algorithm, and we choose a new set of abstraction predicates such that there exist concrete configurations y_1, $y_2 \in S^C$ with $y_1 \in \gamma(q_2)$ and $y_2 \in \gamma(q_3)$ such that there is no transition from y_1 to y_2. For example, by choosing the new abstraction predicate $\psi_1 \equiv x > y$ the formula $\exists\, y_1, y_2 \in S^C.\, y_1 \in \gamma(q_2) \ \wedge\ y_2 \in \gamma(q_3) \ \wedge\ y_1 \not\Rightarrow y_2$ can be shown to hold using a verification procedure for this decidable fragment of arithmetic. Figure 6 shows the reachable fragment of the resulting approximation of $\mathcal{M}$ with $\Psi = \{\psi_0, \psi_1\}$. Model checking the formula $\varphi = \neg\exists\,(tt\, U\, at_l_2)$ 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 developed 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 time-abstractions are computed lazily. This results in substantial savings in computation whenever coarse abstractions are sufficient to prove the property at hand. Initial investigations are encouraging in

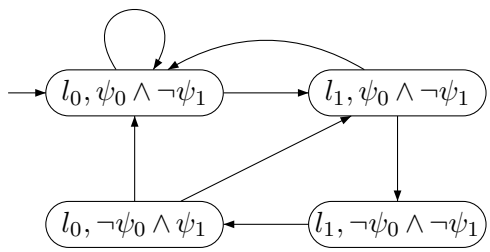


Fig. 6. Over-/Under-approximation (reachable part) of the timed system from Figure 1 with $\Psi = \{x = 0, x > y\}$.

that standard benchmark examples for timed systems such as the train-gate controller and a version of the Fischer mutual exclusion protocol can generally be proved using only a few abstraction predicates. However, more experimentation is needed to corroborate the thesis that many real-life timed systems can already be verified with rather coarse-grain abstractions.

The algorithm as described in this paper is restricted to deal with real-time systems with finite control only. The predicate abstraction of timed systems, however, can readily be extended to also apply to richer models such as parameterized timed automata and even to timed automata with other infinite data types such as counters or stacks. The price to pay, of course, is that such extensions are necessarily incomplete. In future work we would also like to address time-abstracting formulas with arithmetic and other constraints instead of only supporting propositional variables. In this way we could express and verify further interesting properties such a bounded response.

Acknowledgement

We would like to thank H. Saidi, N. Shankar, and T. Uribe for their valuable comments on this paper.

References

[1] Alur, R., C. Courcoubetis and D. Dill, *Model-checking for real-time systems*, 5th Symp. on Logic in Computer Science (LICS 90) (1990), pp. 414–425.

[2] Alur, R. and D. L. Dill, *A theory of timed automata*, Theoretical Computer Science **126** (1994), pp. 183–235.

[3] Bensalem, S., Y. Lakhnech and S. Owre, *Computing abstractions of infinite state systems compositionally and automatically*, Lecture Notes in Computer Science **1427** (1998), pp. 319–331.

[4] Clarke, E., O. Grumberg, S. Jha, Y. Lu and H. Veith, *Counterexample-guided abstraction refinement*, Lecture Notes in Computer Science **1855** (2000), pp. 154–169.

[5] Cousot, P. and R. Cousot, *Abstract intrepretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints*, in: *Conference Record of the 4th ACM Symposium on Principles of Programming Languages*, Los Angeles, CA, 1977, pp. 238–252.

[6] Dams, D. R., "Abstract Interpretation and Partition Refinement for Model Checking," Ph.D. thesis, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands (1996).

[7] Das, S. and D. L. Dill, *Successive approximation of abstract transition relations*, in: *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS-01)* (2001), pp. 51–60.

[8] Dill, D. and H. Wong-Toi, *Verification of real-time systems by successive over and under approximation*, Lecture Notes in Computer Science **939** (1995), pp. 409–422.

[9] Filliâtre, J.-C., S. Owre, H. Rueß and N. Shankar, *ICS: Integrated Canonization and Solving*, in: G. Berry, H. Comon and A. Finkel, editors, *Proceedings of CAV'2001*, Lecture Notes in Computer Science **2102** (2001), pp. 246–249.

[10] Graf, S. and H. Saïdi, *Construction of abstract state graphs with PVS*, Lecture Notes in Computer Science **1254** (1997), pp. 72–83.

[11] Henzinger, T. A., X. Nicollin, J. Sifakis and S. Yovine, *Symbolic model checking for real-time systems*, Information and Computation **111** (1994), pp. 193–244.

[12] Kick, A., "Generation of counterexamples and witnesses for the Mu-calculus," Ph.D. thesis, University of Karlsruhe, Germany (1996).

[13] Kozen, D., *Results on the propositional μ-calculus*, Theoretical Computer Science **27** (1983), pp. 333–354.

[14] Lachnech, Y., S. Bensalem, S. Berezin and S. Owre, *Incremental verification by abstraction*, Lecture Notes in Computer Science **2031** (2001), pp. 98–112.

[15] Möller, M. O., H. Rueß and M. Sorea, *Predicate abstraction for dense real-time systems*, Technical Report BRICS-RS-01-44, Department of Computer Science, University of Aarhus, Denmark (2001), Available online at http://www.brics.dk/RS/01/44/.

[16] Namjoshi, K. and R. Kurshan, *Syntactic program transformations for automatic abstraction*, Lecture Notes in Computer Science **1855** (2000), pp. 435–449.

[17] Saïdi, H. and N. Shankar, *Abstract and model check while you prove*, Lecture Notes in Computer Science **1633** (1999), pp. 443–454.

[18] Tripakis, S. and S. Yovine, *Analysis of timed systems using time-abstracting bisimulations*, Formal Methods in System Design **18** (2001), pp. 25–68.

[19] Uribe, T. E., "Abstraction-based Deductive-Algorithmic Verification of Reactive Systems," Ph.D. thesis, Computer Science Department, Stanford University (1998), technical report STAN-CS-TR-99-1618.

[20] Yovine, S., *Model-checking timed automata*, in: G. Rozenberg and F. Vaandrager, editors, *Embedded Systems*, number 1494 in Lecture Notes in Computer Science, 1998, pp. 114–152.