Basic Research in Computer Science

Predicate Abstraction for Dense Real-Time Systems

M. Oliver Möller
Harald Rueß
Maria Sorea

BRICS Report Series **RS-01-44**

ISSN 0909-0878 **November 2001**

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/01/44/`

Predicate Abstraction for Dense Real-Time Systems*

M. Oliver Möller[‡] Harald Rueß[§]
Maria Sorea[§]

[‡] BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
8000 Århus C, Denmark
omoeller@brics.dk

[§] **SRI International**
Computer Science Laboratory
333 Ravenswood Avenue
Menlo Park, CA 94025
USA
{ruess,sorea}@csl.sri.com

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.

*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 Introduction

Timed Automata [AD94] are state-transition graphs augmented with a finite set of real-valued clocks. The clocks proceed at a uniform rate and constraint the times at which transitions may occur. Given a timed automaton and a property expressed in a timed logic such as TCTL [ACD90] or T_μ [HNSY94], 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 [ACD90] 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 [Yov98].

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* [CC77], and, in particular, of *predicate abstraction* [GS97]. 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 [SS99]. 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 [Uri98] 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 fourth 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 [CGJ$^+$00, DD01, LBBO01]). 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.

Dill and Wong-Toi [DWT95] 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 [NK00] 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, since there is no explicitly stated upper bound on the number of abstraction rounds and abstraction predicates needed for convergence. In contrast, for the special case of timed automata, we are able to predetermine a finite set of abstraction predicates. Tripakis and Yovine [TY01] 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 Sec-

tion 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.

2 Timed Systems

We review some basic notions of transition systems and timed systems. Furthermore, we introduce the notion of time-progressing systems by syntactically restricting the delay steps. These restrictions, however, are not observable in a version of the propositional μ-calculus without a next-step operator. This sets 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 [ACD90].[1] 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 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 (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.

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

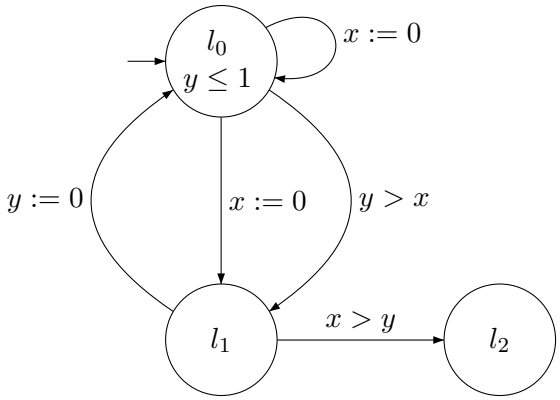


Figure 1: Example of a Timed System.

We write $l \xrightarrow{g,r} l'$ for $\langle l, g, r, l' \rangle \in T$. Firing a transition does not only change 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 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 $\nu : C \to \mathbb{R}_0^+$ 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 $\nu \models g$. A set $\mathcal{X} \subseteq \mathcal{V}_C$ of clock evaluations satisfies $g \in \mathit{Constr}$, written as $\mathcal{X} \models g$, if and only if $\nu \models g$ 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, $\nu \models I(l)$ if $\nu \models g$ for every invariant $g \in I(l)$. Alur, Courcoubetis, and Dill [ACD90] introduce the fundamental notion of clock regions, which partition the space of possible clock evaluation for a timed automaton into finitely many regions.

Definition 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 $\nu_1 \models g$ if and only if $\nu_2 \models g$. 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 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 $\nu + \delta \models I(l)$ 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 $\nu \models g$, $\nu' = \nu[r{:=}0]$, and $\nu' \models I(l')$. The union of delay and state transition steps defines the timed transition relation $\Rightarrow$ of a timed system $\mathcal{S}$. Now, a *path* is an infinite or maximally extended finite 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

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

for example, never reaches time point 1. Systems with paths 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 6 (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.

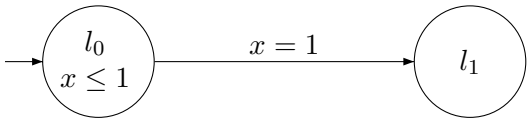
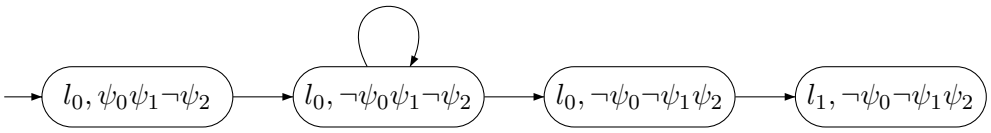


Figure 2: Timed System for Example 7.

Example 7 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 path of this system.

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

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 assumption 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 path $(*)$ above, for example, are explicitly ruled out.

Definition 8 (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

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

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 sub-relation of $\Rightarrow$. The restriction of delay steps above does not necessarily enforce time to progress, as is demonstrated by the following restricted path for the system in Example 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 paths 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 8.

Definition 9 (Semantics of Timed Systems) Let $\mathcal{S} = \langle L, P, C, T, l_0, I\rangle$ be a timed system. We associate two transition systems $\mathcal{M}$ and $\mathcal{M}_R$ with $\mathcal{S}$ as follows.

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

The symbol ν_0 denotes the special clock evaluation, that maps every clock to 0. $\mathcal{M}$ is called the *natural semantics* of $\mathcal{S}$, 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 [Koz83] is a branching-time temporal logic, where formulas are built from atomic propositions, boolean connectives, the least-fixpoint operator, 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 paths are considered to be observationally equivalent. Logics without explicit next-step operator have also been considered, for example, by Dams [Dam96] and Tripakis and Yovine [TY01].

Definition 10 (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

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

In addition, every variable is assumed to appear under an even number of negations. 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. We use the abbreviations:

ff	$:= \neg tt$	
$\nu Z.\varphi(Z)$	$:= \neg\mu Z.\neg\varphi(Z)$	greatest fixpoint
$\Diamond^*\,\varphi$	$:= \exists\,(tt\,U\varphi)$	on some path, φ holds eventually
$\Box^*\,\varphi$	$:= \forall\,(tt\,U\varphi)$	for all paths, φ holds eventually

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 $s = (l, \nu)$ for which the formula holds. Sub-formulas 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 11 (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.

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

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

Definition 12 (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 valuations are in the same clock region if and only if they are μ-equivalent. Consequently, μ-equivalence is of finite index.

Lemma 13 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 $(l, \nu) \equiv_{\mathcal{M}} (l, \nu')$.

Proof. Following Definition 12 two time configurations (l, ν) and (l, ν') are μ-equivalent if and only if

$$\forall\, \varphi \in \mathcal{L}_\mu .\, (l, \nu) \in [\![\, \varphi \,]\!]^{\mathcal{M}} \Leftrightarrow (l, \nu') \in [\![\, \varphi \,]\!]^{\mathcal{M}}$$

For arbitrary sentences $\varphi \in \mathcal{L}_\mu$ we show $(l, \nu) \in [\![\, \varphi \,]\!]^{\mathcal{M}}$ if and only if $(l, \nu') \in [\![\, \varphi \,]\!]^{\mathcal{M}}$. The proof works by a straightforward structural induction on φ.

$\boxed{\varphi = tt}$ From Definition 11 we have $[\![\, tt \,]\!]^{\mathcal{M}} = S$. Therefore $(l, \nu) \in [\![\, tt \,]\!]^{\mathcal{M}}$ and $(l, \nu') \in [\![\, tt \,]\!]^{\mathcal{M}}$.

$\boxed{\varphi = p}$ Also following Definition 11 we obtain $[\![\, p \,]\!]^{\mathcal{M}} = \{\tilde{l}, \tilde{\nu} \mid p \in \mathcal{P}(l)\}$. And since the configurations (l, ν) and (l, ν') have the same locations, both are contained in $[\![\, p \,]\!]^{\mathcal{M}}$.

$\boxed{\varphi = \varphi_1 \wedge \varphi_2}$ By Definition 11 we have that $[\![\, \varphi_1 \wedge \varphi_2 \,]\!]^{\mathcal{M}} = [\![\, \varphi_1 \,]\!]^{\mathcal{M}} \cap [\![\, \varphi_2 \,]\!]^{\mathcal{M}}$, and by Induction Hypothesis it follows

$$\begin{array}{rcl} (l, \nu) \in [\![\, \varphi_1 \,]\!]^{\mathcal{M}} & \Leftrightarrow & (l, \nu') \in [\![\, \varphi_1 \,]\!]^{\mathcal{M}} \text{ and} \\ (l, \nu) \in [\![\, \varphi_2 \,]\!]^{\mathcal{M}} & \Leftrightarrow & (l, \nu') \in [\![\, \varphi_2 \,]\!]^{\mathcal{M}} \end{array}$$

Thus, $(l, \nu) \in [\![\, \varphi_1 \,]\!]^{\mathcal{M}} \cap [\![\, \varphi_2 \,]\!]^{\mathcal{M}} \Leftrightarrow (l, \nu') \in [\![\, \varphi_1 \,]\!]^{\mathcal{M}} \cap [\![\, \varphi_2 \,]\!]^{\mathcal{M}}$.

$\boxed{\varphi = \neg\varphi_1}$ By Definition 11 we have that $[\![\, \neg\varphi_1 \,]\!]^{\mathcal{M}} = S \setminus [\![\, \varphi_1 \,]\!]^{\mathcal{M}}$. By Induction Hypothesis we obtain $(l, \nu) \notin [\![\, \varphi_1 \,]\!]^{\mathcal{M}} \Leftrightarrow (l, \nu') \notin [\![\, \varphi_1 \,]\!]^{\mathcal{M}}$, and therefore $(l, \nu) \in [\![\, \neg\varphi_1 \,]\!]^{\mathcal{M}} \Leftrightarrow (l, \nu') \in [\![\, \neg\varphi_1 \,]\!]^{\mathcal{M}}$.

$\boxed{\varphi = \exists\, (\varphi_1\, U\, \varphi_2)}$ By Definition 11

$[\![\exists (\varphi_1 \, U \varphi_2)]\!]^{\mathcal{M}} =$
$\{(l_0, \nu_0) \in S \mid$ there exists a path $\tau = ((l_0, \nu_0) \Rightarrow (l_1, \nu_1) \Rightarrow \ldots)$, s.t.
$(l_i, \nu_i) \in [\![\varphi_2]\!]^{\mathcal{M}}$ for some $i \geq 0$, and for all $0 \leq j < i$, $(l_j, \nu_j) \in [\![\varphi_1]\!]^{\mathcal{M}}\}$
By Induction Hypothesis we have that $(l, \nu'_i) \in [\![\varphi_2]\!]^{\mathcal{M}}$ and $(l, \nu'_j) \in [\![\varphi_1]\!]^{\mathcal{M}}$, for all $0 \leq j < i$.

From the assumption $\nu \equiv_{\mathcal{S}} \nu'$ it follows by Definitions 4 and 5 that there exists a path $\tau' = ((l_0, \nu'_0) \Rightarrow (l_1, \nu'_1) \Rightarrow \ldots)$ such that $(l_i, \nu'_i) \in [\![\varphi_2]\!]^{\mathcal{M}}$ for some $i \geq 0$, and $(l_j, \nu'_j) \in [\![\varphi_2]\!]^{\mathcal{M}}$ for all $0 \leq j < i$. Thus, $(l, \nu') \in [\![\exists (\varphi_1 \, U \varphi_2)]\!]^{\mathcal{M}}$.

$\boxed{\varphi = \forall (\varphi_1 \, U \varphi_2)}$ By Definition 11
$[\![\forall (\varphi_1 \, U \varphi_2)]\!]^{\mathcal{M}} =$
$\{(l, \nu) \in S \mid$ for every path $\tau = ((l_0, \nu_0) \Rightarrow (l_1, \nu_1) \Rightarrow \ldots)$ with $(l_0, \nu_0) = (l, \nu)$, there exists $i \geq 0$ s.t. $(l_i, \nu_i) \in [\![\varphi_2]\!]^{\mathcal{M}}$, and for all $0 \leq j < i$, $(l_j, \nu_j) \in [\![\varphi_1]\!]^{\mathcal{M}}\}$
By Induction Hypothesis we have that $(l, \nu'_i) \in [\![\varphi_2]\!]^{\mathcal{M}}$ and $(l, \nu'_j) \in [\![\varphi_1]\!]^{\mathcal{M}}$, for all $0 \leq j < i$.

From the assumption $\nu \equiv_{\mathcal{S}} \nu'$ it follows by Definitions 4 and 5 that for all paths $\tau' = ((l_0, \nu'_0) \Rightarrow (l_1, \nu'_1) \Rightarrow \ldots)$ there exists $i \geq 0$ such that $(l_i, \nu'_i) \in [\![\varphi_2]\!]^{\mathcal{M}}$, and $(l_j, \nu'_j) \in [\![\varphi_2]\!]^{\mathcal{M}}$ for all $0 \leq j < i$. Thus, $(l, \nu') \in [\![\forall (\varphi_1 \, U \varphi_2)]\!]^{\mathcal{M}}$.

$\boxed{\varphi = \mu Z.\varphi_1}$ Assume $(l, \nu) \in [\![\mu Z.\varphi_1]\!]^{\mathcal{M}}_{\vartheta}$, that is, by Definition 11

$$(l, \nu) \in \bigcap \left\{ \mathcal{E} \subseteq S \mid [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta[Z:=\mathcal{E}]} \subseteq \mathcal{E} \right\}$$

By Induction Hypothesis it follows that $(l, \nu') \in [\![\mu Z.\varphi_1]\!]^{\mathcal{M}}_{\vartheta}$. □

We now show that the natural semantics and the restricted semantics of a timed system as introduced in Definition 9 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 14 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. The proof works by structural induction on the formula φ, where we strengthen the claim to $[\![\varphi]\!]^{\mathcal{M}}_{\vartheta} = [\![\varphi]\!]^{\mathcal{M}_R}_{\vartheta}$ for arbitrary valuation functions ϑ.

$\boxed{\varphi = tt}$ By Definition 11, $[\![tt]\!]^{\mathcal{M}}_{\vartheta} \stackrel{def}{=} S \stackrel{def}{=} [\![tt]\!]^{\mathcal{M}_R}_{\vartheta}$

$\boxed{\varphi = p}$ By Definition 11, $[\![p]\!]^{\mathcal{M}}_{\vartheta} \stackrel{def}{=} \{(l,\nu) \,|\, p \in P(l)\} \stackrel{def}{=} [\![p]\!]^{\mathcal{M}_R}_{\vartheta}$

Induction Hypothesis: assume we already established $[\![\varphi']\!]^{\mathcal{M}}_{\vartheta} = [\![\varphi']\!]^{\mathcal{M}_R}_{\vartheta}$ for all sub-formulas φ' of φ and all valuation functions ϑ.

$\boxed{\varphi = \varphi_1 \wedge \varphi_2}$ By Definition 11 and by Induction Hypothesis we have that

$$[\![\varphi_1 \wedge \varphi_2]\!]^{\mathcal{M}}_{\vartheta} \stackrel{def}{=} [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta} \cap [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta} \stackrel{I.H.}{=} [\![\varphi_1]\!]^{\mathcal{M}_R}_{\vartheta} \cap [\![\varphi_2]\!]^{\mathcal{M}_R}_{\vartheta} \stackrel{def}{=} [\![\varphi_1 \wedge \varphi_2]\!]^{\mathcal{M}_R}_{\vartheta}$$

$\boxed{\varphi = \neg\varphi_1}$ By Definition 11 and by Induction Hypothesis we have that

$$[\![\neg\varphi_1]\!]^{\mathcal{M}}_{\vartheta} \stackrel{def}{=} S \setminus [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta} \stackrel{I.H}{=} S \setminus [\![\varphi_1]\!]^{\mathcal{M}_R}_{\vartheta} \stackrel{def}{=} [\![\neg\varphi_1]\!]^{\mathcal{M}_R}_{\vartheta}$$

$\boxed{\varphi = \exists\,(\varphi_1\, U \varphi_2)}$ According to Definition 11, $s \in [\![\exists\,(\varphi_1\, U \varphi_2)]\!]^{\mathcal{M}}_{\vartheta}$ iff there exists an path starting at s such that

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

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 $(**)$, there exists a path in the restricted semantics which also validates $(**)$. 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 8:

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

Consequently, it is the case that $\nu \equiv_{\mathcal{S}} (\nu + \delta)$. Using Lemma 13, 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 $s_i \in [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta}$ and $\forall\, 1 \leq j < i.\ s_j \in [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta}$. 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 13, 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 sub-path

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

such that $s_{k_m} \in [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta}$ and for all $h < m$, $s_{k_h} \in [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta}$. By induction hypothesis, $s_{k_m} \in [\![\varphi_2]\!]^{\mathcal{M}_R}_{\vartheta}$ and for all $h < m$, $s_{k_h} \in [\![\varphi_1]\!]^{\mathcal{M}_R}_{\vartheta}$. 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 $[\![\exists (\varphi_1 \, U \varphi_2)]\!]^{\mathcal{M}}_{\vartheta} = [\![\exists (\varphi_1 \, U \varphi_2)]\!]^{\mathcal{M}_R}_{\vartheta}$.

$\boxed{\varphi = \forall (\varphi_1 \, U \varphi_2)}$ According to Definition 11, $s \in [\![\forall (\varphi_1 \, U \varphi_2)]\!]^{\mathcal{M}}_{\vartheta}$ iff for all paths starting at s the following holds:

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

Every path in the restricted semantics is also a path in the natural semantics. We have to establish, that if a path in the natural semantics violates the condition $(***)$, then also a path in the restricted semantics does.

Assume a path $\tau = s_1 \Rightarrow s_2 \Rightarrow \cdots$ in the natural semantics, that *violates* the condition

$$(\star) \quad := \quad \exists i.\ s_i \in [\![\varphi_2]\!]^{\mathcal{M}}_{\vartheta} \text{ and for all } 1 \leq j < i,\ s_j \in [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta}.$$

Now we show, that there exists also a path $\tau_R = (s = s_{k_1} \Rightarrow_R s_{k_2} \Rightarrow_R \cdots)$ in the restricted semantics, that violates $(\star)$.

If τ is either finite or contains infinitely many state transition steps, then—by the same argument as in the previous case—there exists also a sub-path τ_R of τ, where no two subsequent configurations have clock evaluations in the same region and τ_R violates $(\star)$.

Suppose τ is infinite and contains only finitely many state transition steps. By the non-convergence assumption it cannot contain an infinite suffix of delay steps, without exceeding the largest constant c for every clock at some point s_k. By Lemma 13, $s_k \equiv_{\mathcal{M}} s_{k'}$ for all $k' \geq k$. Therefore, if τ violates $(\star)$, then already the finite prefix $s_1 \Rightarrow \cdots \Rightarrow s_k$ does. For this finite prefix we can construct a sub-path τ_R according to the restricted semantics as before.

Thus $[\![\forall (\varphi_1 \, U \varphi_2)]\!]^{\mathcal{M}}_{\vartheta} = [\![\forall (\varphi_1 \, U \varphi_2)]\!]^{\mathcal{M}_R}_{\vartheta}$.

$\boxed{\varphi = Z}$ By Definition 11 it follows $[\![Z]\!]^{\mathcal{M}}_{\vartheta} \stackrel{def}{=} \vartheta(Z) \stackrel{def}{=} [\![Z]\!]^{\mathcal{M}_R}_{\vartheta}$.

$\boxed{\varphi = \mu Z.\varphi_1}$ By Definition 11 and Induction Hypothesis it follows

$$\begin{aligned} [\![\mu Z.\varphi_1]\!]^{\mathcal{M}}_{\vartheta} &\stackrel{def}{=} \bigcap \left\{ \mathcal{E} \subseteq S \mid [\![\varphi_1]\!]^{\mathcal{M}}_{\vartheta[Z:=\mathcal{E}]} \subseteq \mathcal{E} \right\} \stackrel{I.H}{=} \\ &\stackrel{I.H}{=} \bigcap \left\{ \mathcal{E} \subseteq S \mid [\![\varphi_1]\!]^{\mathcal{M}_R}_{\vartheta[Z:=\mathcal{E}]} \subseteq \mathcal{E} \right\} \stackrel{def}{=} [\![\mu Z.\varphi_1]\!]^{\mathcal{M}_R}_{\vartheta}. \end{aligned}$$

□

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 [GS97,BLO98,SS99] 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 15 (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 valuations ν to a *bit-vector* 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 bit-vectors 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 bit-vector to the set of clock valuations which satisfy all ψ_i whenever the i-th component of the bit-vector is set. Thus, a set of concrete states (l, ν) is transformed by the abstraction function α into the abstract state $(l, \alpha(\nu))$, and an abstract state (l, b) is mapped by γ to a set of concrete states $(l, \gamma(b))$.

Definition 16 (Abstraction/Concretization) 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* $\alpha : \mathcal{V}_C \to B_n$ is defined by

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

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

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

By a slight abuse of notation, we also write $\alpha(l, \nu)$ instead of $(l, \alpha(\nu))$ and $\gamma(l, b)$ instead of $(l, \gamma(b))$. Furthermore, we 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.

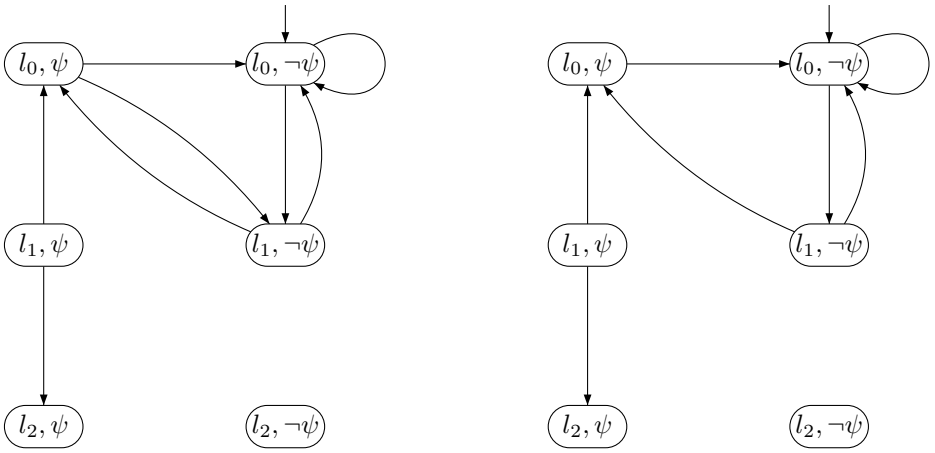
Definition 17 (Over-/Under-approximation) Given a (concrete) transition system $\mathcal{M} = \langle S^C, P, \Rightarrow, s_0^C \rangle$, where $S^C = L \times \mathcal{V}_C$ and $s_0^C = (l_0, \nu_0)$, and a set Ψ of abstraction predicates, we construct two (abstract) transition systems $\mathcal{M}_\Psi^+ = \langle S^A, P, \Rightarrow^+, s_0^A \rangle$, 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 $\exists \nu \in \gamma(b).\ \exists \nu' \in \gamma(b').\ (l,\nu) \Rightarrow (l',\nu')$
- $(l,b) \Rightarrow^- (l',b')$ iff $\forall \nu \in \gamma(b).\ \exists \nu' \in \gamma(b').\ (l,\nu) \Rightarrow (l',\nu')$.
- ${s^A}_0 := (l_0, b_0)$, where $b_0(i) = 1$ iff $\nu_0 \models \psi_i$.

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

Since $\gamma(b) \neq \emptyset$, we have that $\Rightarrow^- \subseteq \Rightarrow^+$.

Example 18 Figure 3 shows the over- and under-approximation of the (concrete) system from Figure 1 with respect to the predicate set $\Psi = \{x > y\}$.



a: Over-approximation b: Under-approximation

Figure 3: Over-/Under-approximation of the timed system from Figure 1 with $\psi \equiv x > y$.

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

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

Lemma 19 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

1. $\gamma(\Rightarrow^-) \subseteq \Rightarrow \subseteq \gamma(\Rightarrow^+)$, and
2. $\Rightarrow^- \subseteq \alpha(\Rightarrow) \subseteq \Rightarrow^+$.

Proof. Follows from Definition 17. □

Definition 20 (Predicate Abstraction) Let $\mathcal{M} = \langle S^C, P, \Rightarrow, s_0^C \rangle$ be a transition system with corresponding over-approximation $\mathcal{M}_\Psi^+ = \langle S^A, P, \Rightarrow^+, s_0^A \rangle$, and under-approximation $\mathcal{M}_\Psi^- = \langle S^A, P, \Rightarrow^-, s_0^A \rangle$, as given in Definition 17. Then, the *predicate abstracted* semantics $[\![\varphi]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma}$, 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 $\mathcal{M}_\Psi^\sigma$ is defined in a mutually inductive way. The notation $\overline{\sigma}$ is used to toggle the sign σ.

$$
\begin{array}{rcl}
[\![tt]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} & := & S^A \\
[\![p]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} & := & \{(l,b) \in S^A \mid p \in P(l)\} \\
[\![\varphi_1 \wedge \varphi_2]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} & := & [\![\varphi_1]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} \cap [\![\varphi_2]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} \\
[\![\neg\varphi]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} & := & S^A \setminus [\![\varphi]\!]_\vartheta^{\mathcal{M}_\Psi^{\overline{\sigma}}} \\
[\![\exists (\varphi_1 \, U \varphi_2)]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} & := & \{s_0 \in S^A \mid \text{there exists a path } \tau = (s_0 \Rightarrow^\sigma s_1 \Rightarrow^\sigma \ldots), \\
 & & \quad \text{s.t. } s_i \in [\![\varphi_2]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} \text{ for some } i \geq 0, \text{ and} \\
 & & \quad \text{for all } 0 \leq j < i,\ s_j \in [\![\varphi_1]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma}\} \\
[\![\forall (\varphi_1 \, U \varphi_2)]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} & := & \{s_0 \in S^A \mid \text{for every path } \tau = (s_0 \Rightarrow^{\overline{\sigma}} s_1 \Rightarrow^{\overline{\sigma}} \ldots), \\
 & & \quad \text{there exists } i \geq 0, \text{s.t. } s_i \in [\![\varphi_2]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma}, \\
 & & \quad \text{and for all } 0 \leq j < i,\ s_j \in [\![\varphi_1]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma}\} \\
[\![Z]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} & := & \vartheta(Z) \\
[\![\mu Z.\varphi]\!]_\vartheta^{\mathcal{M}_\Psi^\sigma} & := & \bigcap \{\mathcal{E} \subseteq S^A \mid [\![\varphi]\!]_{\vartheta[Z:=\mathcal{E}]}^{\mathcal{M}_\Psi^\sigma} \subseteq \mathcal{E}\}
\end{array}
$$

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 21 (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 is by induction on the structure of φ. We show here only the case $\varphi = \exists\,(\varphi_1\, U \varphi_2)$. By induction hypothesis we have that

$$\gamma([\![\,\varphi_1\,]\!]^{\mathcal{M}^-_\Psi}) \;\subseteq\; [\![\,\varphi_1\,]\!]^{\mathcal{M}} \;\subseteq\; \gamma([\![\,\varphi_1\,]\!]^{\mathcal{M}^+_\Psi})$$

and

$$\gamma([\![\,\varphi_2\,]\!]^{\mathcal{M}^-_\Psi}) \;\subseteq\; [\![\,\varphi_2\,]\!]^{\mathcal{M}} \;\subseteq\; \gamma([\![\,\varphi_2\,]\!]^{\mathcal{M}^+_\Psi}).$$

Let $\rightsquigarrow^\sigma_\gamma$, $\rightsquigarrow_\alpha$ denote the transition relations $\gamma(\Rightarrow^\sigma)$, $\alpha(\Rightarrow)$ respectively.
$\gamma([\![\,\exists\,(\varphi_1\, U \varphi_2)\,]\!]^{\mathcal{M}^-_\Psi}) \;=$

$=$ /⋆ by Definition 20 ⋆/
$\gamma(\{s_0 \in S^A \mid$ there exists a path $\tau = (s_0 \Rightarrow^- s_1 \Rightarrow^- \ldots)$, s.t.
$s_i \in [\![\,\varphi_2\,]\!]^{\mathcal{M}^-_\Psi}$ for some $i \geq 0$, and for all $0 \leq j < i$, $s_j \in [\![\,\varphi_1\,]\!]^{\mathcal{M}^-_\Psi}\})$

$=$ /⋆ by Definition 20 ⋆/
$\{s \in \gamma(S^A) \mid$ there exists a path $\tau = (\gamma(s_0) \rightsquigarrow^-_\gamma \gamma(s_1) \rightsquigarrow^-_\gamma \ldots)$
with $s = \gamma(s_0)$,
and $\rightsquigarrow^-_\gamma = \gamma(\Rightarrow^-)$, s.t. $\gamma(s_i) \in \gamma([\![\,\varphi_2\,]\!]^{\mathcal{M}^-_\Psi})$ for some $i \geq 0$,
and for all $0 \leq j < i$, $\gamma(s_j) \in \gamma([\![\,\varphi_1\,]\!]^{\mathcal{M}^-_\Psi})\}$

$=$ /⋆ by induction hypothesis ⋆/
$\{s \in \gamma(S^A) \mid$ there exists a path $\tau = (\gamma(s_0) \rightsquigarrow^-_\gamma \gamma(s_1) \rightsquigarrow^-_\gamma \ldots)$
with $s = \gamma(s_0)$,
and $\rightsquigarrow^-_\gamma = \gamma(\Rightarrow^-)$, s.t. $\gamma(s_i) \in [\![\,\varphi_2\,]\!]^{\mathcal{M}}$ for some $i \geq 0$,
and for all $0 \leq j < i$, $\gamma(s_j) \in [\![\,\varphi_1\,]\!]^{\mathcal{M}}\}$

$\subseteq$ /⋆ by Definition 17 and Lemma 19 ⋆/
$\{s^c \in S^C \mid$ there exists a path $\tau = ({s^c}_0 \Rightarrow {s^c}_1 \Rightarrow \ldots)$ with $s^c = {s^c}_0$,
and ${s^c}_i \in \gamma(s_i)$, for all $i \geq 1$, s.t. ${s^c}_i \in [\![\,\varphi_2\,]\!]^{\mathcal{M}}$ for some $i \geq 0$,
and for all $0 \leq j < i$, ${s^c}_j \in [\![\,\varphi_1\,]\!]^{\mathcal{M}}\}$

$\subseteq$ /⋆ by Definition 11 ⋆/
$[\![\,\exists\,(\varphi_1\, U \varphi_2)\,]\!]^{\mathcal{M}}$

$[\![\,\exists\,(\varphi_1\, U \varphi_2)\,]\!]^{\mathcal{M}} \;=$

$=$ /⋆ by Definition 11 ⋆/
$\{{s^c}_0 \in S^C \mid$ there exists a path $\tau = ({s^c}_0 \Rightarrow {s^c}_1 \Rightarrow \ldots)$, s.t.
${s^c}_i \in [\![\,\varphi_2\,]\!]^{\mathcal{M}}$ for some $i \geq 0$, and

$$\text{for all } 0 \le j < i,\ {s^c}_j \in [\![\, \varphi_1 \,]\!]^{\mathcal{M}}$$

$\subseteq$ /⋆ by Definitions 16 and 17 ⋆/

$$\gamma(\{s^a \in \alpha(S^C) \mid \text{there exists a path } \tau = ({s^a}_0 \rightsquigarrow_\alpha {s^a}_1 \rightsquigarrow_\alpha \ldots)$$
$$\text{with } s^a = {s^a}_0,$$
$$\text{and } \rightsquigarrow_\alpha = \alpha(\Rightarrow),\ \text{and } {s^a}_i = \alpha(s_i),\ \text{for all } i \ge 1,\ \text{s.t.}$$
$${s^a}_i \in [\![\, \varphi_2 \,]\!]^{\mathcal{M}^+_\Psi} \text{ for some } i \ge 0,\ \text{and}$$
$$\text{for all } 0 \le j < i,\ {s^a}_j \in [\![\, \varphi_1 \,]\!]^{\mathcal{M}^+_\Psi}\})$$

$\subseteq$ /⋆ by Lemma 19-2 and since γ is monotone ⋆/

$$\gamma(\{s^a \in \alpha(S^C) \mid \text{there exists a path } \tau = ({s^a}_0 \Rightarrow^+ {s^a}_1 \Rightarrow^+ \ldots)$$
$$\text{with } s^a = {s^a}_0,$$
$$\text{and } {s^a}_i = \alpha(s_i),\ \text{for all } i \ge 1,\ \text{s.t. } {s^a}_i \in [\![\, \varphi_2 \,]\!]^{\mathcal{M}^+_\Psi} \text{ for some } i \ge 0,$$
$$\text{and for all } 0 \le j < i,\ {s^a}_j \in [\![\, \varphi_1 \,]\!]^{\mathcal{M}^+_\Psi}\})$$

$=$ /⋆ by Definition 20 ⋆/

$$\gamma([\![\, \exists\, (\varphi_1\, U \varphi_2) \,]\!]^{\mathcal{M}^+_\Psi})$$

□

Example 22 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. The over-approximation of $\mathcal{M}$ with respect to the abstraction predicate $\psi \equiv (x > y)$ is shown in Figure 3. According to Definition 20, the set of abstract states of $\mathcal{M}^+_{\{\psi\}}$ which validate φ is given by

$$[\![\, \neg\exists\, (tt\, U\, at\_l_2) \,]\!]^{\mathcal{M}^+_{\{\psi\}}} = S^A \backslash [\![\, \exists\, (tt\, U\, at\_l_2) \,]\!]^{\mathcal{M}^-_{\{\psi\}}} = \{(l_0, \psi), (l_0, \neg\psi), (l_1, \neg\psi)\}\,.$$

Since the initial state $(l_0, \neg\psi)$ of $\mathcal{M}^+_{\{\psi\}}$ is contained in this set, the formula φ holds on the abstract transition system. Thus, $\mathcal{M}^+_{\{\psi\}}, (l_0, b_0) \models^A \varphi$ holds. By Theorem 21, 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 the over- and under-approximations with respect to only a single abstraction predicate ψ already coincide. We now give a criterion, based on the notion of regions, for a set of abstraction predicates, which is sufficient for guaranteeing convergence 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 23 (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, $\nu_2 \in \mathcal{V}_C$

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

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 $Constr(c)$ with largest constant c as the largest constant of $\mathcal{S}$, 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 24 The set $\Psi{:=}\{x = 0,\ y = 0,\ x \leq 1,\ x \geq 1,\ y \leq 1,\ y \geq 1,\ x > y,\ x < y\}$ is a basis for the timed system in Figure 1.

Theorem 25 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 $\varphi \in \mathcal{L}_\mu$,

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

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 17, there exist $\nu \in \gamma(b)$ and $\nu' \in \gamma(b')$ such that $(l, \nu) \Rightarrow (l', \nu')$.

First, in case $(l, \nu) \Rightarrow (l', \nu')$ holds due to a state transition step, all the guards g of some transition $l \xrightarrow{g,r} l'$ evaluate to the true value for $\nu \in \gamma(b)$. Since Ψ is a basis, the guards then evaluate to the true value for all clock evaluations $\tilde{\nu} \in \gamma(b)$ (Definition 23). The clock values at ν' are either identical to ν, or are reset to 0 (for clocks $x \in r$). Thus for all clock evaluations $\tilde{\nu} \in \gamma(b)$, the state transition step $l \xrightarrow{g,r} l'$ can be applied and leads to a clock evaluation $\tilde{\nu}'$, such that $\tilde{\nu}' \equiv_{\mathcal{S}} \nu' \wedge (l, \tilde{\nu}) \Rightarrow (l', \tilde{\nu}')$. Then, by Definition 17, $(l, b) \Rightarrow^- (l', b')$.

Second, in case $(l, \nu) \Rightarrow (l', \nu')$ holds due to a delay step, then $l = l'$ and the invariants $I(l)$ evaluate to the true value for $\nu' \in \gamma(b')$. Since invariant expressions are taken from *Inv*, by Definition 23, the invariants evaluate to

the true value for all $\tilde{\nu}' \in \gamma(b')$. Moreover, Definition 8 requires that ν and ν' are not in the same region, and thus a delay step according to the restricted semantics is possible. Consequently, at location l, for all $\tilde{\nu} \in \gamma(b)$, a delay step to some $\tilde{\nu}' \in \gamma(b')$ is possible. Again, by Definition 17, $(l, b) \Rightarrow^{-} (l', b')$. □

Corollary 26 (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 $\gamma(b_0) = \nu_0$. Then for any sentence $\varphi \in \mathcal{L}_\mu$:

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

Proof. By Theorem 21, $\gamma([\![\varphi]\!]^{\mathcal{M}^{-}_{\Psi}}) \subseteq [\![\varphi]\!]^{\mathcal{M}} \subseteq \gamma([\![\varphi]\!]^{\mathcal{M}^{+}_{\Psi}})$. By Theorem 25, $[\![\varphi]\!]^{\mathcal{M}^{-}_{\Psi}} = [\![\varphi]\!]^{\mathcal{M}^{+}_{\Psi}}$, and thus $\gamma([\![\varphi]\!]^{\mathcal{M}^{-}_{\Psi}}) = \gamma([\![\varphi]\!]^{\mathcal{M}^{+}_{\Psi}})$. □

5 Refinement of the Abstraction

Given a concrete model $\mathcal{M}$ of a timed system, 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)-(4) in Figure 4). First an over-approximation of $\mathcal{M}$ with respect to the set Ψ_{act} is computed. Now it is checked if this over-approximation satisfies the given formula φ by calling a finite-state μ-calculus model checker. If indeed the approximation satisfies φ, then, by Corollary 26, $\mathcal{M}$ also satisfies φ and the algorithm returns **true** (line (6)). Otherwise, $\mathcal{M}^{+}_{\psi} \not\models \varphi$ and the μ-calculus model checker returns a counterexample of the form $s_0 \Rightarrow^{+} s_1 \Rightarrow^{+} \cdots \Rightarrow^{+} s_n$, where s_0 is the initial state of $\mathcal{M}^{+}_{\psi}$ (see [Kic96]). 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 (9)-(12)). 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 [FORS01]. 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 $i \in \{0, \cdots, k\}$, $y_i \in \gamma(s_i)$, such that there is no (concrete) transition from y_k to y_{k+1}, where $y_{k+1} \in \gamma(s_{k+1})$ (lines (13)-(16)). We choose a minimal set of new abstraction predicates from Ψ_{new} such that

the transition from s_k to s_{k+1} is eliminated (lines (17)-(20)). This new set of abstraction predicates is chosen in such a way that the formula

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

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

abstract_and_refine$(\mathcal{M}, \Psi, \varphi)$ (1)
  choose $\Psi' = \{\psi_1, \ldots, \psi_i\}$ from Ψ; (2)
  $\Psi_{new} := \Psi \setminus \Psi'$; (3)
  $\Psi_{act} := \Psi'$; (4)
  loop (5)
    if $\mathcal{M}^+_{\Psi_{act}} \models \varphi$ **then return** true (6)
    else (7)
      let $(s_0 \Rightarrow^+ s_1 \cdots \Rightarrow^+ s_n)$ be a counterexample; (8)
      if there exists a path $\tau = (y_0 \Rightarrow y_1 \cdots \Rightarrow y_n)$ (9)
        such that $y_0 = s_0^C$ and $y_i \in \gamma(s_i)$ for all $0 \leq i \leq n$ (10)
      then return false (11)
      else (12)
        let k be the index such that (13)
          there exists a path $\tau = (y_0 \Rightarrow y_1 \cdots \Rightarrow y_k)$ where (14)
          $y_0 = s_0^C$ and $y_i \in \gamma(s_i)$ for all $0 \leq i \leq k$ and (15)
          $\forall\, y_{k+1} \in \gamma(s_{k+1}).\, y_k \not\Rightarrow y_{k+1}$; (16)
        choose minimal $\Psi' = \{\psi_1, \ldots, \psi_i\}$ from Ψ_{new}, such that (17)
          $\Psi_{act} := \Psi_{act} \cup \{\Psi'\}$ and (18)
          $\forall\, y_1 \in \gamma(s_k), y_2 \in \gamma(s_{k+1}).\, y_1 \not\Rightarrow y_2$ and (19)
          $\Psi_{new} := \Psi_{new} \setminus \Psi'$ (20)
      endif (21)
    endif (22)
  endloop (23)
end abstract_and_refine. (24)

Figure 4: Iterative Abstraction-Refinement Algorithm.

Theorem 27 (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. The proof follows directly from Theorem 25 and Corollary 26. From Theorem 25 we have that

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

Then, by Theorem 21 it follows that

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

and, by Corollary 26, $\mathcal{M}^+_\Psi$ satisfies the formula φ if and only if $\mathcal{M}$ satisfies φ. Let n be the cardinality of the basis. Then the loop in the abstract_and_refine algorithm terminates after at most n steps. □

Example 28 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 $\Psi{:=}\{x = 0, y = 0, x \leq 1, x \geq 1, y \leq 1, y \geq 1, x > y, x < y\}$. The transition system of the initial over-approximation with the single abstraction predicate $\{x = 0\}$ is shown in Figure 5, where ψ_0 denotes the abstraction predicate.

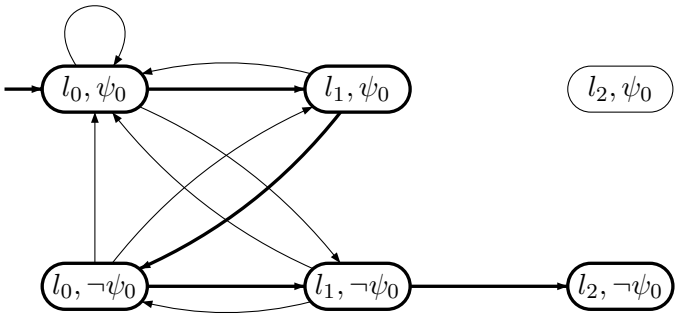


Figure 5: Over-approximation of the timed system from Figure 1 with $\psi_0 \equiv x = 0$.

Model checking the formula φ with transition system $\mathcal{M}^+_{\{x=0\}}$ yields the counterexample

$$(l_0, \psi_0) \Rightarrow^+ (l_1, \psi_0) \Rightarrow^+ (l_0, \neg\psi_0) \Rightarrow^+ (l_1, \neg\psi_0) \Rightarrow^+ (l_2, \neg\psi_0)$$

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 $(l_1,\, x = 0 \wedge y \geq 0)$.

$$\begin{array}{rcl}
\gamma(s_0) & = & \gamma((l_0,\psi_0)) \ = \ (l_0,\gamma(\psi_0)) \ = \ (l_0,\ x = 0 \ \wedge \ y \geq 0) \\
\gamma(s_1) & = & \gamma((l_1,\psi_0)) \ = \ (l_1,\gamma(\psi_0)) \ = \ (l_1,\ x = 0 \ \wedge \ y \geq 0) \\
\gamma(s_2) & = & \gamma((l_0,\neg\psi_0)) \ = \ (l_0,\gamma(\neg\psi_0)) \ = \ (l_0,\ x > 0 \ \wedge \ y \geq 0) \\
\gamma(s_3) & = & \gamma((l_1,\neg\psi_0)) \ = \ (l_1,\gamma(\neg\psi_0)) \ = \ (l_1,\ x > 0 \ \wedge \ y \geq 0) \\
\gamma(s_4) & = & \gamma((l_2,\neg\psi_0)) \ = \ (l_2,\gamma(\neg\psi_0)) \ = \ (l_2,\ x > 0 \ \wedge \ y \geq 0)
\end{array}$$

Now we have to check if there is a corresponding counterexample on the concrete transition system, that is, if there exists a path $y_0 \Rightarrow y_1 \Rightarrow y_2 \Rightarrow y_3 \Rightarrow y_4$, where $y_0,\ y_1,\ y_2,\ y_3,\ y_4 \in S^C$, such that $y_0 \in \gamma(s_0)$, $y_1 \in \gamma(s_1)$, $y_2 \in \gamma(s_2)$, $y_3 \in \gamma(s_3)$, $y_4 \in \gamma(s_4)$, and $y_0 = s_0^C$. This is the case if the formula

$$\begin{array}{rl}
F_1 := & \exists\, y_0, y_1, y_2, y_3, y_4 \in S^C. \\
& y_0 \in \gamma(s_0) \ \wedge \ y_1 \in \gamma(s_1) \ \wedge \ y_2 \in \gamma(s_2) \ \wedge \ y_3 \in \gamma(s_3) \ \wedge \ y_4 \in \gamma(s_4) \ \wedge \\
& y_1 \Rightarrow y_2 \ \wedge \ y_2 \Rightarrow y_3 \ \wedge \ y_3 \Rightarrow y_4 \ \wedge \\
& y_0 = s_0^C
\end{array}$$

is valid. In our example, F_1 is unsatisfiable, since on the concrete transition system there is no transition between y_3 and y_4, as it is illustrated by the following path.

$$\underbrace{(l_0, x = y = 0)}_{\ni y_0} \Rightarrow \underbrace{(l_1, x = 0 \wedge 0 \leq y \leq 1)}_{\ni y_1} \Rightarrow$$
$$\underbrace{(l_0, x > 0 \wedge y \leq 1 \wedge x \geq y)}_{\ni y_2} \Rightarrow \underbrace{(l_1, x > 0 \wedge y > x)}_{\ni y_3}$$

Thus, $k = 3$ 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 $y_1 \in \gamma(s_3)$ and $y_2 \in \gamma(s_4)$ 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 $\exists\, y_1, y_2 \in S^C.\, y_1 \in \gamma(s_3) \ \wedge \ y_2 \in \gamma(s_4) \ \wedge \ y_1 \not\Rightarrow y_2$ can be shown to hold using a verification procedure for this decidable fragment of arithmetic. Figure 6 shows the reachable fragment of the resulting over-approximation $\mathcal{M}^+_{\{\psi_0,\psi_1\}}$. Model checking the formula $\varphi = \neg\exists\,(tt\ U\ at\_l_2)$ on $\mathcal{M}^+_{\{\psi_0,\psi_1\}}$ succeeds, since no state $(l_2, \_)$ is reachable in $\mathcal{M}^+_{\{\psi_0,\psi_1\}}$.

6 Conclusion

We have developed a verification algorithm for timed automata based on predicate abstraction, untimed model checking, and decision procedures for the

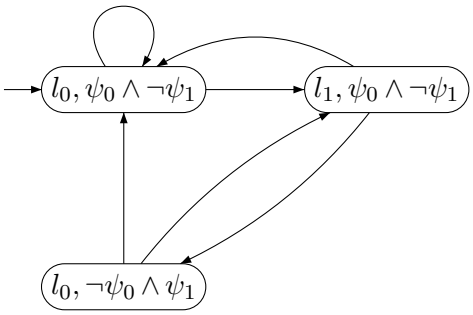


Figure 6: Over-approximation (reachable part) of the timed system from Figure 1 with $\Psi = \{x = 0, x > y\}$.

Boolean combination of linear arithmetic constraints. The main advantage of this approach is that bisimilar 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 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. Such an observation has already been made by Alur, Itai, Kurshan, and Yannakakis [AIKY95] in a similar context. 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 as bounded and unbounded response.

Acknowledgments. We would like to thank Hassen Saïdi, Natarajan Shankar, and Tomás Uribe for their valuable comments on this paper.

References

[ACD90] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking for real-time systems. *5th Symp. on Logic in Computer Science (LICS 90)*, pages 414–425, 1990.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 25 April 1994.

[AIKY95] Rajeev Alur, Alon Itai, Robert P. Kurshan, and Mihalis Yannakakis. Timing verification by successive approximation. *Information and Computation*, 118(1):142–157, April 1995.

[BLO98] Saddek Bensalem, Yassine Lakhnech, and Sam Owre. Computing abstractions of infinite state systems compositionally and automatically. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer-Aided Verification (CAV'98)*, number 1427 in Lecture Notes in Computer Science, pages 319–331, Vancouver, Canada, June 1998. Springer–Verlag.

[CC77] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis. In *4th ACM Symposium on Principles of Programming Languages*. Association for Computing Machinery, January 1977.

[CGJ+00] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In E.A. Emerson and A.P. Sistla, editors, *Proc. of the 12th Conference on Computer-Aided Verification, CAV'2000*, number 1855 in Lecture Notes in Computer Science. Springer–Verlag, 2000.

[Dam96] Dennis René Dams. *Abstract Interpretation and Partition Refinement for Model Checking*. PhD thesis, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, July 1996.

[DD01] Satyaki Das and David L. Dill. Successive approximation of abstract transition relations. In *Proc. of Logic in Computer Science (LICS2001)*, 2001.

[DWT95] David L. Dill and Howard Wong-Toi. Verification of real-time systems by successive over and under approximation. In P. Wolper, editor, *Proc. of the 7th Conference on Computer-Aided Verification, CAV'95*, number 939 in Lecture Notes in Computer Science, pages 409–422. Springer–Verlag, 1995.

[FORS01] Jean-Christophe Filliâtre, Sam Owre, Harald Rueß, and Natarajan Shankar. ICS: Integrated canonizer and solver. In A. Finkel G. Berry, H. Comon, editor, *CAV 01: Computer-Aided Verification*, number 2101 in Lecture Notes in Computer Science. Springer–Verlag, 2001.

[GS97] Susanne Graf and Hassen Saïdi. Construction of abstract state graphs with PVS. In Orna Grumberg, editor, *Computer Aided Verification. 9th International Conference (CAV97)*, number 1254 in Lecture Notes in Computer Science, pages 72–83. Springer–Verlag, 1997.

[HNSY94] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, June 1994.

[Kic96] Alexander Kick. *Generation of counterexamples and witnesses for the Mu-calculus*. PhD thesis, University of Karlsruhe, Germany, 1996.

[Koz83] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[LBBO01] Yassine Lachnech, Saddek Bensalem, Sergey Berezin, and Sam Owre. Incremental verification by abstraction. In T. Margaria and W. Yi, editors, *Tools and Algorithms for the Construction and Analysis of Systems: 7th International Conference, TACAS 2001*, number 2031 in Lecture Notes in Computer Science, pages 98–112, Genova, Italy, April 2001. Springer–Verlag.

[NK00] Kedar S. Namjoshi and Robert P. Kurshan. Syntactic program transformations for automatic abstraction. In E.A. Emerson and A.P. Sistla, editors, *Proc. of the 12th Conference on Computer-Aided Verification, CAV'2000*, number 1855 in Lecture Notes in Computer Science, pages 435–449, Chicago, IL, 2000. Springer–Verlag.

[SS99] Hassen Saïdi and Natarajan Shankar. Abstract and model check while you prove. In Nicolas Halbwachs and Doron Peled, editors, *Computer-Aided Verification (CAV'99)*, number 1633 in Lecture Notes in Computer Science, pages 443–454, Trento, Italy, July 1999. Springer–Verlag.

[TY01] Stavors Tripakis and Sergio Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, January 2001.

[Uri98] Tomás E. Uribe. *Abstraction-based Deductive-Algorithmic Verification of Reactive Systems*. PhD thesis, Computer Science Department, Stanford University, December 1998. Technical report STAN-CS-TR-99-1618.

[Yov98] Sergio Yovine. Model-checking timed automata. In G. Rozenberg and F. Vaandrager, editors, *Embedded Systems*, number 1494 in Lecture Notes in Computer Science, pages 114–152. Springer–Verlag, 1998.

Recent BRICS Report Series Publications

RS-01-44 **M. Oliver Möller, Harald Rueß, and Maria Sorea.** ***Predicate Abstraction for Dense Real-Time Systems.*** **November 2001. 27 pp.**

RS-01-43 **Ivan B. Damgård and Jesper Buus Nielsen.** ***From Known-Plaintext Security to Chosen-Plaintext Security.*** **November 2001. 18 pp.**

RS-01-42 **Zoltán Ésik and Werner Kuich.** ***Rationally Additive Semirings.*** **November 2001. 11 pp.**

RS-01-41 **Ivan B. Damgård and Jesper Buus Nielsen.** ***Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor.*** **October 2001. 43 pp.**

RS-01-40 **Daniel Damian and Olivier Danvy.** ***CPS Transformation of Flow Information, Part II: Administrative Reductions.*** **October 2001. 9 pp.**

RS-01-39 **Olivier Danvy and Mayer Goldberg.** ***There and Back Again.*** **October 2001. 14 pp.**

RS-01-38 **Zoltán Ésik.** ***Free De Morgan Bisemigroups and Bisemilattices.*** **October 2001. 13 pp.**

RS-01-37 **Ronald Cramer and Victor Shoup.** ***Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption.*** **October 2001. 34 pp.**

RS-01-36 **Gerth Stølting Brodal, Rolf Fagerberg, and Riko Jacob.** ***Cache Oblivious Search Trees via Binary Trees of Small Height.*** **October 2001.**

RS-01-35 **Mayer Goldberg.** ***A General Schema for Constructing One-Point Bases in the Lambda Calculus.*** **September 2001. 6 pp.**

RS-01-34 **Flemming Friche Rodler and Rasmus Pagh.** ***Fast Random Access to Wavelet Compressed Volumetric Data Using Hashing.*** **August 2001. 31 pp.**

RS-01-33 **Rasmus Pagh and Flemming Friche Rodler.** ***Lossy Dictionaries.*** **August 2001. 14 pp. Short version appears in Meyer auf der Heide, editor,** ***9th Annual European Symposiumon on Algorithms,*** **ESA '01 Proceedings, LNCS 2161, 2001, pages 300–311.**