Reachability Relations and Sampled Semantics of Timed Systems

Pavel Krčál
Department of Information Technology,
Uppsala University, Sweden
pavelk@it.uu.se

Radek Pelánek
Department of Computer Science, Faculty of Informatics,
Masaryk University Brno, Czech Republic
xpelanek@fi.muni.cz

December 5, 2005

Abstract

Timed systems can be considered with two types of semantics – dense time semantics and discrete time semantics. The most typical examples of both of them are *real* semantics and *sampled* semantics (i.e., discrete semantics with a fixed time step ϵ). In this work, we study different aspects of sampled semantics. At first, we study reachability relations between configurations of a timed automaton and provide a novel effective characterization of reachability relations. This result is used for proving our main result — decidability of non-emptiness of a timed automaton ω-language in some sampled semantics (this problem was previously wrongly classified as undecidable). Also, we study relations between real semantics and sampled semantics with respect to different behavioral equivalences. Finaly, we study decidability of reachability problem for stopwatch automata with sampled semantics.

1 Introduction

In this paper we are concerned with formal verification of timed systems. As models of timed systems we consider mainly timed automata (TA) [2]; some results are also shown for stopwatch automata (SWA) [17] — an extension of timed automata which allows clocks to be stopped in some locations.

The semantics of these models can be defined over various time domains. The usual approach is to use *dense* time semantics, particularly *real* time semantics (time domain is $\mathbb{R}_0^+$). From many points of view, this semantics is very plausible. One does not need to care about the granularity of time during the modeling phase. This semantics leads to an uncountable structure with a finite quotient (for timed automata) and thus it is amenable to verification with finite state methods. Moreover, theoretical and often also practical complexity of problems for dense time semantics is usually the same as for various discrete semantics.

Discrete semantics, particularly *sampled* semantics with fixed time step $\epsilon \in \mathbb{Q}_{>0}^+$ (time domain is $\{k \cdot \epsilon \mid k \in \mathbb{Z}_0^+\}$), is also often considered, e.g., in [10, 9, 6]. One of the advantages is that with the sampled time domain we have a wider choice of representations for sets of clock valuations, e.g., explicit representation or symbolic representation using decision diagrams. Another important issue is implementability. If a system is realized on a hardware then there is always some granularity of time (e.g., clock cycle, sampling period). Therefore, sampled semantics is closer to the implementation then more abstract dense time semantics.

Dense time semantics can even give us misleading verification results. Assume that we have a model of a timed system such that it satisfies some property in dense time semantics. Now the question is whether there is an implementation (realized on a discrete time hardware) such that it preserves this property. Dense time semantics allows behaviors which are not realizable in any real system. If satisfaction of the property depends on these behaviors then there might not be an implementation satisfying the property.

Verification problems can be stated as the (ω-)language non-emptiness. By verification of a model A with respect to sampled semantics we mean answering the question whether there exists ϵ such that the (ω-)language of A is empty in sampled semantics with ϵ as the sampling period. We show that this problem is decidable for timed automata and that one can also synthesize such ϵ. This problem for ω-languages was previously wrongly classified as undecidable [3]. The same problem was studied in the control setting with slightly different sampled behavior in [11]. The automaton is a model of a controller with the periodic control loop which always gets a sampled data and performs a control action. Therefore, the automaton has to perform an action at every sampled time point. This is a difference from our definition of sampled semantics – the automaton can idle for several sampling periods. The fact that the automaton has to react at every sampled time point makes the problem undecidable even for the (finite word) language non-emptiness.

Our proof uses a novel characterization of reachability relations in timed automata. Representations of reachability relations were studied before: using additive theory of real numbers [13] and $2n$-automata [14]. Our novel representation is based on simple linear (in)equalities (comparisons of clock differences). This representation is of independent interest, since it is simpler and more specific then previously considered characterizations and it gives a better insight into reachability relations.

We also systematically study relations between dense time semantics and sampled semantics for different timed systems. We study these relations in terms of behavioral equivalences, as it is well known which verification results are preserved by which equivalence. These results are summarized in Table 1. For sampled semantics, a given result means that there exists an ϵ such that a given equivalence is guaranteed. All considered equivalences are "untimed" – the only important information for an equivalence are actions performed and not the precise timepoints at which these actions are taken.

Finally, we summarize the (un)decidability of the reachability problem in timed systems (Table 2). Particularly, we provide a new undecidability proof for the reachability problem in sampled semantics for stopwatch automata with diagonal constraints and one stopwatch. One stopwatch suffices for all undecidability results.

Related Work

There has been a considerable amount of work related to discretization issues and verifying dense time properties using discrete time methods, e.g., [18, 20, 21, 5]. The main difference is that usually only a fixed sampling rate and trace equivalence are considered.

Implementability issues are discussed in conection with robust semantics of timed automata in [24, 23]. Goal of these works is to decide whether a set of bad states is reachable if the clock rates drift a little bit or the guards are enlarged a little bit. It is argued that a real hardware can never produce synchronized clocks and measure them with infinite precision.

Discretization of timed automata preserving reachability has also been studied in [15]. Authors present an elaborate discretization scheme which preserves reachability (and in fact even bisimilarity) for any timed automaton. Our discretization scheme is just sampling with a fixed rate.

Practical aspects of verification with the use of sampled semantics are discussed in [10, 9, 6, 7, 4]. These works are concerned mainly with data structures for representing sets of discrete valuations (e.g., different types of decision diagrams). They do not consider theoretical problems concerning relations between dense and sampled semantics in greater depth.

The expressive power of stopwatch automata is studied in [12]. The usefullness of stopwatch automata for modeling scheduling problems is studied in [1, 19].

Table 1: A summary of the equivalences: each field gives the relation to real semantics.

equivalences	closed TA	TA	SWA
rational semantics	bisimilar	bisimilar	trace eq.
sampled semantics	similar	reachability eq.	reachability eq.

Table 2: A summary of decidability results for the reachability problem.

reachability	TA	diagonal-free SWA	SWA
dense semantics	PSPACE-complete	undecidable	undecidable
sampled semantics	PSPACE-complete	PSPACE-complete	undecidable

2 Preliminaries

In this section we define syntax and semantics of the automata. We define a stopwatch automaton and a timed automaton as a special case of the stopwatch automaton. Semantics is defined as a labeled transition system (LTS). We also define usual untimed behavioral equivalences on LTS, equivalences on valuations, and a region graph.

Labeled Transition Systems

An LTS is a tuple $T = (S, Act, \rightarrow, s_0)$ where S is a set of states, *Act* is a finite set of actions, $\rightarrow \subseteq S \times Act \times S$ is a transition relation, $s_0 \in S$ is an initial state. A *run* of T over a trace $w \in Act^* \cup Act^\omega$ is a sequence of states $\pi = q_0, q_1, \ldots$ such that $q_0 = s_0$ and $q_i \xrightarrow{w(i)} q_{i+1}$. The set of finite (resp. infinite) traces of the transition system is $L(T) = \{w \in Act^* \mid$ there exists a run of T over $w\}$ (resp. $L_\omega(T) = \{w \in Act^\omega \mid$ there exists a run of T over $w\}$).

Equivalences

Let $T_1 = (S_1, Act, \rightarrow_1, s_0^1)$, $T_2 = (S_2, Act, \rightarrow_2, s_0^2)$ be two labeled transitions systems. A relation $R \subseteq S_1 \times S_2$ is a *simulation relation* iff for all $(s_1, s_2) \in R$ and $s_1 \xrightarrow{a}_1 s_1'$ there is s_2 such that $s_2 \xrightarrow{a}_2 s_2'$ and $(s_1', s_2') \in R$. System T_1 is simulated by T_2 if there exists a simulation R such that $(s_0^1, s_0^2) \in R$. A relation R is a *bisimulation relation* iff R is a symmetric simulation relation. A *bisimulation* $\sim$ is the largest bisimulation relation. A set of *reachable actions* $RA(T)$ is the set $\{a \in Act \mid s_0 \rightarrow^* s_n \xrightarrow{a} s_{n+1}\}$. Systems T_1, T_2 are:

- *reachability equivalent* iff $RA(T_1) = RA(T_2)$,
- *trace equivalent* iff $L(T_1) = L(T_2)$,
- *infinite trace equivalent* iff $L_\omega(T_1) = L_\omega(T_2)$,
- *simulation equivalent* iff T_1 simulates T_2 and vice versa,
- *bisimulation equivalent* (bisimilar) iff $s_0^1 \sim s_0^2$.

Syntax

Let $\mathcal{C}$ be a set of non-negative real-valued variables called *clocks*. The set of guards $G(\mathcal{C})$ is defined by the grammar $g := x \bowtie c \mid x - y \bowtie c \mid g \wedge g$ where $x, y \in \mathcal{C}, c \in \mathbb{N}_0$ and $\bowtie \in \{<, \leq, \geq, >\}$. A *stopwatch automaton* is a tuple $A = (Q, Act, \mathcal{C}, q_0, E, stop)$, where:

- Q is a finite set of locations,
- $\mathcal{C}$ is a finite set of clocks,
- $q_0 \in Q$ is an initial location,
- $E \subseteq Q \times Act \times G(\mathcal{C}) \times 2^{\mathcal{C}} \times Q$ is a set of edges labeled by an action name, a guard, and a set of clocks to be reset,

- $stop : Q \to 2^{\mathcal{C}}$ assigns to each location a set of clocks that are stopped at this location.

A clock $x \in \mathcal{C}$ is called a *stopwatch* clock if $\exists q \in Q : x \in stop(q)$. We use the following special types of stopwatch automata:

- a *timed automaton* is a stopwatch automaton such that there are no stopwatch clocks (i.e., $\forall q \in Q : stop(q) = \emptyset$),
- a *closed* automaton uses only guards with $\{\leq, \geq\}$,
- a *diagonal-free* automaton uses only guards defined by $g := x \bowtie c \mid g \wedge g$.

We also consider combinations of these types, e.g., closed timed automaton.

Semantics

Semantics is defined with respect to a given time domain D. We suppose that a time domain is a subset of real numbers which contains 0 and is closed under addition. A *clock valuation* is a function $\nu : \mathcal{C} \to D$. If $\delta \in D$ then a valuation $\nu + \delta$ is such that for each clock $x \in \mathcal{C}$, $(\nu + \delta)(x) = \nu(x) + \delta$. If $Y \subseteq \mathcal{C}$ then a valuation $\nu[Y := 0]$ is such that for each clock $x \in \mathcal{C} \setminus Y, \nu[Y := 0](x) = \nu(x)$ and for each clock $x \in Y, \nu[Y := 0](x) = 0$. The satisfaction relation $\nu \models g$ for $g \in G(\mathcal{C})$ is defined in the natural way.

The semantics of a stopwatch automaton $A = (Q, Act, \mathcal{C}, q_0, E, stop)$ with respect to the time domain D is an LTS $[\![A]\!]_D = (S, Act, \to, s_0)$ where $S = Q \times D^{\mathcal{C}}$ is the set of states, $s_0 = (q_0, \nu_0)$ is the initial state, $\nu_0(x) = 0$ for all $x \in \mathcal{C}$. Transitions are defined with the use of two types of basic steps:

- time step: $(q, \nu) \xrightarrow{delay(\delta)} (q, \nu')$ if $\delta \in D, \forall x \in stop(q) : \nu'(x) = \nu(x), \forall x \in \mathcal{C} \setminus stop(q) : \nu'(x) = \nu(x) + \delta$,
- action step: $(q, \nu) \xrightarrow{action(a)} (q', \nu')$ if there exists $(q, a, g, Y, q') \in E$ such that $\nu \models g, \nu' = \nu[Y := 0]$.

The transition relation of $[\![A]\!]_D$ is defined by concatenating these two types of steps: $(q, \nu) \xrightarrow{a} (q', \nu')$ iff there exists (q'', ν'') such that $(q, \nu) \xrightarrow{delay(\delta)} (q'', \nu'') \xrightarrow{action(a)} (q', \nu')$.

We consider the following time domains: $\mathbb{R}_0^+, \mathbb{Q}_0^+, \{k \cdot \epsilon \mid k \in \mathbb{Z}_0^+\}$. The semantics with respect to the last domain is denoted $[\![A]\!]_\epsilon$ (also called *sampled semantics*). We use the following shortcut notation: $L(A) = L([\![A]\!]_{\mathbb{R}_0^+}), L_\omega(A) = L_\omega([\![A]\!]_{\mathbb{R}_0^+}), L^\epsilon(A) = L([\![A]\!]_\epsilon), L_\omega^\epsilon(A) = L_\omega([\![A]\!]_\epsilon)$.

Equivalences on valuations

For any $\delta \in \mathbb{R}$, $\mathsf{int}(\delta)$ denotes the integral part of δ and $\mathsf{fr}(\delta)$ denotes the fractional part of δ. Let k be an integer constant. We define the following relations on the valuations. The equivalence $\cong_k$ is a standard region equivalence (its equivalence classes are regions), the equivalence $\sim_k$ is an auxiliary relation which allows us to forget about the clocks whose values are above k.

- $\nu \cong_k \nu'$ iff all the following conditions hold:
  - for all $x \in \mathcal{C} : \mathsf{int}(\nu(x)) = \mathsf{int}(\nu'(x))$ or $\nu(x) > k \wedge \nu'(x) > k$,
  - for all $x, y \in \mathcal{C}$ with $\nu(x) \leq k$ and $\nu(y) \leq k : \mathsf{fr}(\nu(x)) \leq \mathsf{fr}(\nu(y))$ iff $\mathsf{fr}(\nu'(x)) \leq \mathsf{fr}(\nu'(y))$,
  - for all $x \in \mathcal{C}$ with $\nu(x) \leq k : \mathsf{fr}(\nu(x)) = 0$ iff $\mathsf{fr}(\nu'(x)) = 0$;
- $\nu \sim_k \nu'$ iff for all $x \in \mathcal{C} : \nu(x) = \nu'(x)$ or $\nu(x) > k \wedge \nu'(x) > k$.

Note that $\sim_k$ is a refinement of $\cong_k$, $\cong_k$ has a finite index for all semantics, $\sim_k$ has a finite index for sampled semantics. Let A be a diagonal-free timed automaton and K be a maximal constant which occurs in some guard in A. For each location $l \in Q$ and two valuations $\nu \cong_K \nu'$ it holds that (l, ν) is bisimilar to (l, ν').

Region Graph

In the following we suppose that timed automata are without diagonal constraints and that each transition resets at most one clock. Each timed automaton can be transformed into an automaton which satisfies these constraints and which is equivalent to the original one with respect to simulation equivalence.

Given a region D (an equivalence class of $\cong_K$) we define:

- *integral*(D) is a set of clocks x such that $x \leq K$ and $\mathsf{fr}(x) = 0$ in D,
- *fractional*(D) is a set of clocks x such that $x \leq K$ and $\mathsf{fr}(x) \neq 0$ in D,
- *maxfractional*(D) is a set of clocks x such that $x \leq K$ and the fractional part of x is maximal in D,
- *minfractional*(D) is a set of clocks x such that $x \leq K$ and the fractional part of x is minimal in D,

- *above*(D) is a set of clocks x such that $x > K$ in D,
- $D \models g, D' = D[Y := 0]$ are defined naturally.

A region D' is an *immediate time successor* of a region D iff one of the following holds:

- *integral*$(D) \neq \emptyset$: *integral*$(D') = \emptyset$; the ordering of fractional parts is the same in D' as in D; and *above*(D') is the same as *above*(D) except that it contains $x \in$ *integral*(D) such that $D(x) = K$, or
- *integral*$(D) = \emptyset$: *integral*(D') = *maxfractional*(D), integer values of these clocks are incremented; ordering of fractional parts in D' is the same as in D except for clocks in *maxfractional*(D) which become the smallest; *above*(D') = *above*(D).

A *region graph* of a timed automaton A is defined as follows[1]:

- states are tuples (l, D) where l is a location and D is a region such that *integral*$(D) \neq \emptyset$,
- there is a transition $(l, D) \rightarrow (l', D')$ iff one of the following holds
  - **Time**: $l = l'$ and there exists D'' such that D'' is an immediate time successor of D and D' is an immediate time successor of D'',
  - **Reset**: there exists a transition $(l, a, g, Y, l') \in E$ such that $D \models g$ and $D' = D[Y := 0]$,
  - **Time&Reset**: there exists a region D'' and a transition $(l, a, g, Y, l') \in E$ such that D'' is a time successor of D, $D'' \models g$ and $D' = D''[Y := 0]$.

3 Reachability Relations

In this section we study reachability relations and their efficient representations. Reachability relations describe which valuations can be reached from a given valuation. Representations of reachability relations were studied before[2]: using additive theory of real numbers [13] and $2n$-automata [14].

[1]Note that we use non-standard definition, because for the proof we need to work only with regions such that *integral*$(D) \neq \emptyset$ and we want to have 'as small steps as possible'.

[2]Definition of reachability relations in these works is slightly different from ours. This difference is not significant with respect to results about representations.

We present a novel simple representation for reachability relations (called clock difference relations). We use this characterization in the next section to prove our main result (Theorem 4.2). The fact that such simple (in)equalities are sufficient to capture the reachability relations and that these relations can be computed effectively is of independent interest and can be used in other applications (as described in [13, 14]).

3.1 Definition, Representation, and Operations

Let $(l, D), (l, D')$ be two states in a region graph. Then a *reachability relation* of the tuple $(l, D), (l, D')$ is a relation on valuations $C_{(l,D)(l',D')} \subseteq D \times D'$ such that for each $\nu \in D, \nu' \in D'$:

$$(\nu, \nu') \in C_{(l,D)(l',D')} \Longleftrightarrow \exists \nu'' \sim_K \nu' : (l, \nu) \to^+ (l, \nu'')$$

A clock difference relation (CDR) over a set of clocks $\mathcal{C}$ is a set of (in)equalities of the following form:

- $x' - y' \bowtie u - v$
- $x' - y' \bowtie 1 - (u - v)$

where $\bowtie \in \{<, >, =\}$, $x, y, u, v \in \mathcal{C}$. The semantics of a CDR B is defined as follows. We say that a pair of valuations (ν, ν') satisfies B ($(\nu, \nu') \models B$) if and only if:

- if $x' - y' \bowtie u - v \in B$ then $\mathsf{fr}(\nu'(x)) - \mathsf{fr}(\nu'(y)) \bowtie \mathsf{fr}(\nu(u)) - \mathsf{fr}(\nu(v))$,
- if $x' - y' \bowtie 1 - (u - v) \in B$ then $\mathsf{fr}(\nu'(x)) - \mathsf{fr}(\nu'(y)) \bowtie 1 - (\mathsf{fr}(\nu(u)) - \mathsf{fr}(\nu(v)))$,

Theorem 3.1 *Reachability relations $C_{(l,D)(l',D')}$ are effectively definable as finite unions of clock difference relations.*

The proof of this theorem is based on the following key facts:

- If there is a transition $(l, D) \to (l', D')$ then the reachability relation can be directly expressed as a CDR.
- If $(l, D) \to^+ (l'', D'') \to (l', D')$ and the reachability relation over (l, D), (l'', D'') is expressed as a union of CDRs then the reachability relation over (l, D), (l', D') can be expressed as a union of CDRs as well.

- Using these two steps, reachability relations can be computed by a standard dynamic programming algorithm; termination is guaranteed, because there is only a finite number of CDRs over a fixed set of clocks; correctness is proved by induction with respect to the length of a path between (l, D) and (l', D').

In the rest of this section, we will formalize these facts and prove them. We use the letter B (possibly with subscripts) to denote a clock difference relation.

Our goal is to show that a reachability relation $C_{(l,D)(l',D')} \subseteq D \times D'$ can be represented by a CDR B in the following way. For a pair of valuations (ν, ν') we want that $(\nu, \nu') \in C_{(l,D)(l',D')}$ if and only iff $\nu \in D, \nu' \in D'$ and $(\nu, \nu') \models B$. Moreover, we show that for a given pair of regions $(l, D), (l', D')$ where at least one unnormalized clock has zero fractional part we can compute a CDR which represents it. This can be easily extended to arbitrary regions by observing that a time pass from a region where at least one clock has zero fractional part to its immediate successor does not change the CDR.

The algorithm for computing a CDR representation is based on two operations: B^{init}, which computes the representation for one-step reachability relations, and B^{step}, which computes for a given representation a representation of one step longer paths. The definition of B^{init} uses the operation B^{step} which is defined below.

Definition 3.2 *The set* $B^{init}((l, D) \to (l', D'))$ *for a transition* $(l, D) \to (l', D')$ *in the region graph is a CDR defined as follows:*

- *If* $(l, D) \not\twoheadrightarrow (l', D')$ *then* $B^{init} = \emptyset$.
- *Otherwise:*
  - *Add equalities* $u' - v' = u - v$ *for all* $u, v \in \mathcal{C}(D)$, $\mathsf{fr}(D(u)) > \mathsf{fr}(D(v))$ *into* B^{start}.
  - $B^{init} = B^{step}(B^{start}, (l, D) \not\twoheadrightarrow (l', D'))$.

Now we show how to compute a new CDR for a one step longer path $B^{step}(B, (l'', D'') \to (l', D'))$ when we have a CDR B and a transition $(l'', D'') \to (l', D')$. Based on (in)equalities in B we put different (in)equalities into B^{step} according to the type of the transition $(l'', D'') \to (l', D')$. Some (in)equalities might be dropped without adding anything into B^{step}. Finally, we perform a normalization — all (in)equalities containing clocks with a value above the maximal constant are removed.

For simplicity of presentation, when we write an inequality of the form $x' - y' \bowtie exp$ then we mean both $x' - y' \bowtie u - v$ and $x' - y' \bowtie 1 - (u - v)$. Also, $\bowtie \in \{<, >\}$ and $\bowtie^{-1}$

denotes the reverse inequality, e.g., if $\bowtie$ is $<$ then $\bowtie^{-1}$ is $>$. If one line contains two $\bowtie$ signs then it means that inequalities are the same in both cases. We will also sometimes use $x' - y' \leq exp$ or $x' - y' \geq exp$ when $\bowtie \in \{<, =\}$ or $\bowtie \in \{>, =\}$, respectively.

Definition 3.3 *Let* B *be a CDR and* $(l'', D'') \rightarrow (l', D')$. *Then the step operation* $B^{step}(B, (l'', D'') \rightarrow (l', D'))$ *is computed as follows:*

- *Compute the* $B^{step}(B, (l'', D'') \rightarrow (l', D'))$ *clock difference relation according to the Table 3.*
- *Normalize* $B^{step}(B, (l'', D'') \rightarrow (l', D'))$*: remove all (in)equalities containing clocks* $x \in$ *above*(D') *from* $B^{step}(B, (l'', D'') \rightarrow (l', D'))$.

Table 3: Computation of the one-step clock difference relation according to the type of the transition $(l'', D'') \rightarrow (l', D')$. Note that y can be also equal to x or z.

Condition	Inequality in B	Inequality in B^{step}
Time, $x \in \mathit{integral}(D''), y \in \mathit{integral}(D')$		
$u, v \in \mathcal{C}(D''), u, v \neq y$	$u' - v' \bowtie exp$	$u' - v' \bowtie exp$
$u \in \mathcal{C}(D'), u \neq y$	$y' - u' \bowtie exp$	$u' - y' \bowtie^{-1} 1 - (exp)$
Reset, $x \in \mathit{integral}(D'')$, resetting y		
$u, v \in \mathcal{C}(D''), u, v \neq y$	$u' - v' \bowtie exp$	$u' - v' \bowtie exp$
$u \in \mathcal{C}(D''), u \neq y$	$u' - x' \bowtie exp$	$u' - y' \bowtie exp$
Time&Reset, $x \in \mathit{integral}(D''), z \in \mathit{maxfractional}(D'')$, resetting y		
$u, v \in \mathcal{C}(D''), u, v \neq y$	$u' - v' \bowtie exp$	$u' - v' \bowtie exp$
$u \in \mathcal{C}(D''), u \neq x, u \neq y$	$u' - x' \geq exp$	$u' - y' > exp$
$u \in \mathcal{C}(D''), u \neq z, u \neq y$	$z' - u' \geq exp$	$u' - y' < 1 - (exp)$

In the following we write just B^{init} and B^{step} in the case that the arguments are clear from the context. Also, a valuation v always belongs to the starting region in B^{init}. The below given algorithm computes relations $C_{(l,D)(l',D')}$ by iteratively taking the step operation. The correctness is proved by showing that by taking the step operation k-times we compute a CDR representing reachability relations for the paths of the length k.

To prove this we need the following lemmas, which establish correctness of operations B^{init} and B^{step}.

For the proofs we need some technical definitions:

- by $\nu(exp)$, where $exp = x - y$ or $exp = 1 - (x - y)$ we mean $\mathsf{fr}(\nu(x)) - \mathsf{fr}(\nu(y))$ or $1 - (\mathsf{fr}(\nu(x)) - \mathsf{fr}(\nu(y)))$ respectively,
- by $(\nu, \nu') \models_y B$, where B is a CDR, $\nu' \in D$, we mean that there exists $\nu'' \in D$ such that $\nu''(x) = \nu'(x), \forall x \neq y$ and $(\nu, \nu'') \models B$. When B is a singleton then we write $(\nu, \nu') \models_y i$, i is an (in)equality from a CDR,
- a valuation ν *respects a region* D for a set of clocks Cl if and only if there is a valuation $\nu' \in D$ such that $\nu(x) = \nu'(x)$ for all $x \in Cl$ and $\nu(y) = 0$ for all other clocks,
- by saying that an (in)equality i contains x, y as primed variables we mean that i is of the form $x' - y' \bowtie exp$ or $y' - x' \bowtie exp$.

At first, we show that satisfaction of (in)equalities is not changed by the duration of time when the order of the fractional parts of the clock values is preserved. Also, an (in)equality can be modified to preserve its satisfaction when the order of the fractional parts of the clock values changes.

Lemma 3.4 *Let ν' be a valuation, i an (in)equality from a CDR of the form $x' - y' \bowtie exp$.*

1. *Let t be a real number such that $\nu' + t$ has the same order of fractional parts of x and y. Then $(\nu, nu') \models i \Leftrightarrow (\nu, \nu' + t) \models i$.*
2. *Let t be a real number such that $\nu' + t$ has the opposite order of fractional parts of x and y. Then $(\nu, nu') \models x' -' y \bowtie exp \Leftrightarrow (\nu, \nu' + t) \models y' - x' \bowtie^{-1} 1 - (exp)$.*

Proof

1. The following (in)equalities are equivalent:

$$\mathsf{fr}(\nu'(x)) - \mathsf{fr}(\nu'(y)) \bowtie \nu(exp)$$
$$\mathsf{fr}(\nu'(x)) + t - (\mathsf{fr}(\nu'(y)) + t) \bowtie \nu(exp)$$
$$\mathsf{fr}(\nu' + t(x)) - \mathsf{fr}(\nu' + t(y)) \bowtie \nu(exp)$$

2. The following (in)equalities are equivalent:

$\mathsf{fr}(\nu'(x)) - \mathsf{fr}(\nu'(y)) \bowtie \nu(exp)$

$\mathsf{fr}(\nu'(x)) + t - (\mathsf{fr}(\nu'(y)) + t) \bowtie \nu(exp)$

$\mathsf{fr}(\nu' + t(x)) - \mathsf{fr}(\nu' + t(y)) - 1 \bowtie \nu(exp)$ (because $\mathsf{int}(\nu' + t(y)) - \mathsf{int}(\nu'(y)) = \mathsf{int}(\nu' + t(x)) - \mathsf{int}(\nu'(x)) + 1$)

$\mathsf{fr}(\nu' + t(y)) - \mathsf{fr}(\nu' + t(x)) \bowtie^{-1} 1 - \nu(exp)$

□

3.2 Closure Under Projection

We need to prove that a CDR computed using B^{init} and B^{step} is "closed under projection" for the correctness proof. We want the following fact to hold true: we can drop (in)equalities containing some clock x and there will still remain (in)equalities ensuring existence of a valuation for x such that the dropped (in)equalities are satisfied. The proofs and auxiliary lemmas are rather technical.

Lemma 3.5 *Let* B *be a CDR computed by iterating the* B^{step} *operation on the result of the* B^{init} *operation with the resulting region* D.

For each (in)equality $i_1 \in B$ *containing* u, y *as primed variables and for all clocks* $v \in \mathcal{C}(D)$ *there is an (in)equality* $i_2 \in B$ *containing* u, v *as primed variables such that for all* D' *time successors of* D *(*$D' = D + t$*) where* v *is integral:*

$$(\nu, \nu') \models i_2 \Rightarrow (\nu, \nu') \models_y i_1$$

where ν' *is a valuation respecting* D' *for* u, v, y.

Proof First note that if i_1 contains $<$ then the lemma holds trivially for any i_2. The proof is by induction on the number of iterations of B^{step} with the basic step for B^{init} (zero iterations of B^{step}).
Basic step: The equality $v' - u' = v - u$ guarantees the existence of a valuation for y.
Induction step:

Time: Holds by induction hypothesis and Lemma 3.4.

Reset: All needed (in)equalities are inherited from a clock x ($x \in$ *integral*(D)) by induction hypothesis.

Time&Reset: There are new inequalities for which we should also get an (in)equality i_2. For an inequality of the form $u' - y' > exp$ and a clock v there is an (in)equality for

$u' - x' > exp$ or $u' - x' = exp$ and a clock v (induction hypothesis). Together with the information from the region ($\mathsf{fr}(\nu(x)) > \mathsf{fr}(\nu(y))$ for any $\nu \in D'$, D' is a time successor of D such that $v \in \mathit{integral}(D')$) we know that such (in)equality can be used as i_2. For an inequality of the form $u' - y' < 1 - exp$ and a clock v there is an (in)equality for $z' - u' > exp$ or $z' - u' = exp$, $z \in \mathit{maxfractional}(D)$ and a clock v (induction hypothesis). Together with the Lemma 3.4 and an information from the region ($\mathsf{fr}(\nu(y)) > \mathsf{fr}(\nu(z))$ for any $\nu \in D'$, D' is a time successor of D such that $v \in \mathit{integral}(D')$) we know that such (in)equality can be used as i_2.

We have to take care about the clock y in the position of the clock v in the lemma. I.e., for each (in)equality of the form $u' - v' \bowtie exp$ or $v' - u' \bowtie exp$ and the clock y there is an (in)equality $u' - y' \bowtie exp'$ such that the condition from the lemma holds. But there is an (in)equality $u' - x' \bowtie_1 exp'$ for the clock x or $z' - u' \bowtie_2 exp'$ for the clock z instead of y (induction hypothesis). If $\bowtie_1, \bowtie_2 \in \{>, =\}$ then we can use an inequality resulting from the step operation ($u' - y' > exp'$ or $u' - y' < 1 - exp'$), because each of them implies the original (in)equality. We have that $(\nu, \nu') \models u' - y' > exp' \Rightarrow (\nu, \nu') \models_x u' - x' \bowtie_1 exp', \nu' \in D$, by the information from the region D, $\mathsf{fr}(\nu'(x)) > \mathsf{fr}(\nu'(y))$. Similarly for the other inequality. It remains to show that the situation where both $\bowtie_1$ and $\bowtie_2$ are $<$ is not possible in the situation where $\bowtie \in \{>, =\}$. But if $\mathsf{fr}(\nu'(u)) > \mathsf{fr}(\nu'(v)), \nu' \in D$ then $u' - x' < exp'$ would not satisfy the induction hypothesis, otherwise $z' - u' < exp'$ would not satisfy the induction hypothesis.

□

Now we observe that Lemma 3.5 gives us more than just the existence of a valuation for y when v is integral (not conflicting with u). It gives us the existence of a valuation for y which does not conflict with any pair of clocks, not necessarily with zero fractional parts. But we have to add an assumption that it does not conflict with u alone.

Lemma 3.6 *Let B be a CDR computed by iterating the B^{step} operation on the result of the B^{init} operation with the resulting region D.*

For each (in)equality $i_1 \in B$ containing u, y as primed variables and for all clocks $v \in \mathcal{C}(D)$ there is an (in)equality $i_2 \in B$ containing u, v as primed variables such that:

$$(\nu, \nu') \models i_2 \wedge (\nu, \nu') \models_{y,v} i_1 \Rightarrow (\nu, \nu') \models_y i_1,$$

where ν' respects D for u, v, y.

Proof By examination of all possible permutations of three clocks in a region and all possible (in)equalities. We have to show that either an existence of a valuation for y is always guaranteed or that i_2 is the same as in Lemma 3.5 and in this case the (in)equality is the correct one (for which there is an i_2 according to Lemma 3.5).

Consider a region D where $\mathsf{fr}(\nu(y)) < \mathsf{fr}(\nu(u)) < \mathsf{fr}(\nu(v)), \nu \in D$. If $u' - y' > exp \in B$ or $u' - y' = exp \in B$ then there is an (in)equality i_2 satisfying Lemma 3.5. Now for every (ν, ν') such that $(\nu, \nu') \vDash_{y,v} u' - y' \bowtie exp$, ν' respecting D for u, y, v we get from Lemma 3.4 that $(\nu, \nu') \vDash i_2 \Rightarrow (\nu, \nu') \vDash_y u' - y' \bowtie exp$.

If $u' - y' < exp \in B$ then for any $(\nu, \nu'), \nu' \in D$ we have that $(\nu, \nu') \vDash_y u' - y' < exp$.

All other cases are similar. $\square$

To summarize the result of Lemma 3.5 and Lemma 3.6, we know that for any clock y the existence of a valuation for y satisfying an individual (in)equality is ensured by other (in)equalities not containing y. In other words, we have shown that no (in)equality from B^{step} conflicts with constraints from a region for valuations satisfying other (in)equalities from B^{step}. Therefore, it allows us to drop such (in)equality when necessary, because its "projections" with (in)equalities generated by a region are contained in a CDR computed by B^{init} and B^{step}.

Lemma 3.7 *Let* B *be a CDR computed by iterating the* B^{step} *operation on the result of the* B^{init} *operation with the resulting region* D*,* $Y \subset B = \{i_1, \dots, i_n\}$ *be the greatest set containing (in)equalities with* y *as a primed variable:*

$$\forall 1 \le j \le n, \nu' \in D.(\nu, \nu') \vDash B - Y \Rightarrow (\nu, \nu') \vDash_y i_j$$

Proof There is at least one clock with fractional part equal to zero in D which enables us to use Lemma 3.5. Its result is then used in Lemma 3.6. All (in)equalities contain two primed clocks as well as all constraints induced by the region and hence it is enough to show that the value of any pair of them does not exclude the existence of a valuation for the third one. $\square$

Now we need to show that no two such i_j, i_k are in conflict.

Lemma 3.8 *Let* B *be a CDR computed by iterating the* B^{step} *operation on the result of the* B^{init} *operation with the resulting region* D.

For each (in)equality $i_1 \in B$ *containing* u, y *as primed variables and an (in)equality* $i_2 \in B$ *containing* y, v *as primed variables there is an (in)equality* $i_3 \in B$ *containing* u, v *as primed variables such that:*

$$(\nu, \nu') \models i_3 \land (\nu, \nu') \models_{y,v} i_1 \land (\nu, \nu') \models_{y,u} i_2 \Rightarrow (\nu, \nu') \models_y \{i_1, i_2\},$$

where ν' *respects* D *for* u, v, y.

Proof The proof is by induction on the number of iterations of B^{step} with the basic step for B^{init} (zero iterations of B^{step}).
Basic step: The equality $v' - u' = v - u$ guarantees the existence of a valuation for y for i_1 and i_2 together.
Induction step:

Time: Holds by induction hypothesis and Lemma 3.4.

Reset: All needed (in)equalities are inherited from a clock x ($x \in$ *integral*(D)) by induction hypothesis.

Time&Reset: There are new inequalities for which there should also be a corresponding i_3.

- Consider a pair of inequalities $u' - y' > e_1$ and $v' - y' < 1 - e_2$. These inequalities were introduced because of inequalities $u' - x' \geq e_1$ and $z' - v' \geq e_2$. From the latter (in)equality we can derive that there was also an inequality $v' - x' < e_3$ in the previous CDR using Lemma 3.6 and the clock x. Then according to the induction hypothesis, there was also an inequality i_3 with u, v as primed variables which is preserved in B. Now we will show that i_3 is the inequality we want to have. Let ν' be a valuation respecting D for x, y, z, u, v.

  We know that $(\nu, \nu') \models_{x,y,z} u' - y' > e_1$ together with constraints from D implies that $(\nu, \nu') \models_{y,y,z} u' - x' \geq e_1$. Also, $(\nu, \nu') \models_{x,y,z} v' - y' < 1 - e_2$ together with constraints from D implies that $(\nu, \nu') \models_{x,y,z} z' - v' \geq e_2$. Then

  $$(\nu, \nu') \models_{y,z} i_3 \land (\nu, \nu') \models_{x,y,z} u' - y' > e_1 \land (\nu, \nu') \models_{x,y,z} v' - y' < 1 - e_2$$

  implies (the last conjunct is for free)

  $$(\nu, \nu') \models_{y,z} i_3 \land (\nu, \nu') \models_{x,y,z} u' - x' \geq e_1 \land (\nu, \nu') \models_{x,y,z} v' - x' < e_3$$

  and this implies

  $$(\nu, \nu') \models_{x,y,z} \{u' - x' \geq e_1, v' - x' < e_3\}$$

  which implies (we get back to z)

  $$(\nu, \nu') \models_{x,y,z} \{u' - x' \geq e_1, z' - v' \geq e_2\}$$

which finally together with constraints from D implies

$$(\nu, \nu') \vDash_{x,y,z} \{u' - y' > e_1, \nu' - y' < 1 - e_2\}.$$

- Consider a pair of (in)equalities $u' - y' > e_1$ and $u' - \nu' \leq e_2$. The first one was introduced because of an (in)equality $u' - x' \geq e_1$ in the previous CDR. Then, according to the induction hypothesis there is an (in)equality $\nu' - x' \geq e_3$ in the previous CDR and due to B^{step} operation there is an inequality $\nu' - y' > e_3$ in B. From Lemma 3.4 and region constraints we get that $(\nu, \nu') \vDash \nu' - y' > e_3 \Rightarrow (\nu, \nu') \vDash_x \nu' - x' \geq e_3, \nu'$ respecting D for ν, y, x.

  Because x has the smallest non-zero fractional part in D and it is not constrained from below by $\nu' - x' \geq e_3$, we have that $(\nu, \nu') \vDash_{u,x} u' - x' \geq e_1$ for a ν' respecting D for u, x, ν, y. Therefore, $(\nu, \nu') \vDash \nu' - y' > e_3 \wedge (\nu, \nu') \vDash_u u' - \nu' \leq e_3 \wedge (\nu, \nu') \vDash_{u,x} u' - x' \geq e_1 \Rightarrow (\nu, \nu') \vDash_{u,x} \{u' - \nu' \leq e_2, u' - x' > e_1\}$. But this together with the region constraints implies $(\nu, \nu') \vDash_u \{u' - \nu' \leq e_2, u' - y' > e_1\}$.

- The argument is exactly the same for a pair of (in)equalities $u' - y' > e_1$ and $\nu' - u' \geq e_2$.

- For pairs of (in)equalities $u' - y' > e_1$ and $u' - \nu' > e_2$ or $u' - y' > e_1$ and $\nu' - u' < e_2$ there is always a valuation for u such that the constraints are satisfied together (when they are satisfied separately). Therefore, i_3 is any (in)equality.

- Consider a pair of (in)equalities $u' - y' < 1 - e_1$ and $\nu' - u' \leq e_2$. The first one was introduced because of an (in)equality $z' - u' \geq e_1$ in the previous CDR. Then, according to the induction hypothesis there is an (in)equality $z' - \nu' \geq e_3$ in the previous CDR and due to B^{step} operation there is an inequality $\nu' - y' < 1 - e_3$ in B. The argument that $\nu' - y' < 1 - e_3$ is the inequality we want is the same as above.

- The argument is exactly the same for a pair of (in)equalities $u' - y' < 1 - e_1$ and $u' - \nu' \geq e_2$.

- For pairs of (in)equalities $u' - y' < 1 - e_1$ and $u' - \nu' < e_2$ or $u' - y' < 1 - e_1$ and $\nu' - u' > e_2$ there is always a valuation for u such that the constraints are satisfied together (when they are satisfied separately). Therefore, i_3 is any (in)equality.

□

Finally, the following lemma summarizes all the results. Here we abuse the notation, because we want to find a valuation where a clock which was reset has its value before the reset. Such a clock can have a different position in the region before and after the reset. We assume that the valuation respects the region before the reset for these clocks (they need not have the fractional part equal to zero).

Lemma 3.9 *Let* B *be a CDR computed by iterating the* B^{step} *operation on the result of the* B^{init} *operation with the resulting region* D*,* $Y = \{i_1, \dots, i_n\}$ *be the greatest set containing (in)equalities with* $x, y, \dots, z$ *as a primed variables which were removed during the* B^{step} *operation (due to either reset or normalization). Then*

$$\forall v' \in D.(v, v') \models B \Rightarrow (v, v') \models_{x,y,\dots,z} Y.$$

Proof Follows from the lemmas above.

□

3.3 Correctness of B^{init} and B^{step}

Now we can state and prove the correctness of the operations B^{step} and B^{init}.

Lemma 3.10 (Correctness of B^{init})
The set represented by $B^{init}((l, D) \to (l', D'))$ *is equal to* $\{(v, v') \mid v \in D, v' \in D', \exists \bar{v} \sim_K v'.(l, v) \to (l, \bar{v})\}$.

Proof

We show that the set represented by B^{start} (i.e., after the first two steps of the construction) is equal to the set $\{(v, v') \mid v = v'\}$. The rest follows from the correctness of the B^{step} operation.

"$\subseteq$":

We know that there is a clock $x \in$ *integral*(D). Equalities $y' - x' = y - x$ ensure that only (v, v) satisfy B^{start}.

"$\supseteq$":

Holds trivially.

□

Lemma 3.11 (Correctness of B^{step})

Let $C \subseteq \{(v, v') \mid v \in D, v' \in D''\}$ *be a set represented by a CDR* $B_{(l,D),(l'',D'')}$ *obtained by iterating* B^{step} *on a result of* B^{init} *with initial region* (l, D) *and final region* (l'', D'') *and* $(l'', D'') \rightarrow (l', D')$ *be a transition in the region graph. Then the set represented by* $B^{step}(B_{(l,D),(l'',D'')}, (l'', D'') \rightarrow (l', D'))$ *is equal to* $\{(v, v') \mid v \in D, v' \in D', \exists v'' \in D'', \exists \bar{v} \sim_K v'.(v, v'') \in C \wedge (l'', v'') \rightarrow (l', \bar{v})\}$.

Proof

We keep the notation from the Table 3., i.e., $x \in$ *integral*(D''), $z \in$ *maxfractional*(D''), and either $y \in$ *integral*(D') or y is reset by the transition. We show the equivalence by proving the two inclusions. For each inclusion we give separate argumentation for each possible type of transition $(l'', D'') \rightarrow (l', D')$.

"$\subseteq$":

Time: Let us set $v'' = \bar{v} - t$ for some t, where $\bar{v}(\nu) = K + t$ for all $\nu \in$ *integral*$(D'') \cup$ *above*(D') and $\bar{v}(\nu) = v'(\nu)$ otherwise. We need to show that there exists t such that $(v, v'') \in B$.

If at least one $w \in$ *integral*(D'') was not normalized in D' then $t = v'(w) - \lfloor v'(w) \rfloor$. Otherwise, let $w \in$ *minfractional*(D'). We choose t s.t. $0 < t < v'(w) - \lfloor v'(w) \rfloor$ and for all (in)equalities $\nu' - x' \bowtie exp \in B$ holds that $\mathsf{fr}(v'(\nu)) - t \bowtie v(exp)$. Existence of such t follows from Lemma 3.9.

Obviously, $(l'', v'') \rightarrow (l', v')$. The fact that $(v, v'') \in B$ follows from Lemma 3.4.

Reset: Existence of a valuation v'' such that $(l'', v'') \rightarrow (l', v')$ and $(v, v'') \models B$ follows from Lemma 3.4.

Time& Reset: We define v'' as follows. We first define $\bar{v}$ as $\bar{v}(\nu) = K + t_1$ for all $\nu \in$ *integral*$(D'') \cap$ *above*(D') and for some t_1 and $\bar{v}(\nu) = v'(\nu)$ otherwise. Now $v''(w) = \bar{v}(w) - t_1$ for w which was not reset and $v''(y) = t_2$ and for some t_2.

Obviously, $(l'', v'') \rightarrow (l', v')$. We need to show that there exist t_1, t_2 such that $(v, v'') \in B$.

Let us define t_1 similarly as for the Time transition. If at least one $w \in$ *integral*(D'') was not normalized in D' then $t_1 = v'(w) - \lfloor v'(w) \rfloor$. Otherwise, let $w \in$ *minfractional*(D'). We choose a t_1 s.t. $0 < t_1 < v'(w) - \lfloor v'(w) \rfloor$ and for all (in)equalities $\nu' - x' \bowtie exp \in B$ holds that $\mathsf{fr}(v'(\nu)) - t \bowtie v(exp)$.

Existence of such t_1 and t_2 follows from Lemma 3.9.

"$\supseteq$":

Time: Follows directly from Lemma 3.4.

Reset: Trivially, (ν, ν') satisfies all (in)equalities that are in both B and B^{step}. It also satisfied new (in)equalities $u' - y' \bowtie exp$ because $u' - x' \bowtie exp$ was satisfied by ν, ν'' and $\nu'(y) = \nu''(x)$.

Time&Reset: Let t be such that $\bar{\nu} = \nu'' + t$. Assume that there is some (in)equality $u' - v' \bowtie exp$ in B^{step} such that it is not satisfied by (ν, ν').

- For the first rule it follows from Lemma 3.4.
- For the other rules, $\mathsf{fr}(\nu'(u)) > \mathsf{fr}(\nu'(u)) - t = \mathsf{fr}(\nu''(u))$. So (ν, ν'') does not satisfy B.
- For the fourth and the fifth rule, we know that $t < 1 - \mathsf{fr}(\nu''(z))$. From the assumption, $\mathsf{fr}(\nu'(u)) \geq 1 - \nu(exp)$. The following inequalities are equivalent:

$$\mathsf{fr}(\nu'(u)) \geq 1 - \nu(exp)$$

$$\mathsf{fr}(\nu'(u) - t) \geq (1 - t) - \nu(exp)$$

$$\mathsf{fr}(\nu''(u)) \geq (1 - t) - \nu(exp)$$

$$\mathsf{fr}(\nu''(u)) > \mathsf{fr}(\nu''(z)) - \nu(exp) \text{ (because } t < 1 - \mathsf{fr}(\nu''(z)))$$

$$\mathsf{fr}(\nu''(z)) - \mathsf{fr}(\nu''(u)) < \nu(exp)$$

□

3.4 Computing Reachability Relations

With the operations B^{init}, B^{step} it is easy to compute reachability relations and finish the proof of Theorem 4.2.

Proof[of Theorem 3.1] Representations can be computed by the algorithm in Figure 1 (a data structure B is used to denote sets of CDRs; the operation B^{step} is extended for this case in a natural way). The behavior of the algorithm is 'monotonous' (the size of sets B is only increased) and there is only finite number of CDRs over a fixed set of clocks. Therefore the algorithm will eventually terminate. Now we need to show that the computed sets are the reachability relations $C_{(l,D)(l',D')}$.

We show by induction with respect to k that "For each $(l, D), (l', D'), \nu \in D, \nu' \in D' : \exists \bar{\nu} \sim_K \nu', (l, \nu) \to^k (l, \bar{\nu})$ iff $(\nu, \nu') \in B^k_{(l,D)(l',D')}$". The basic step follows from Lemma 3.10. The induction step follows from Lemma 3.11. □

1. $i = 1$

2. For each pair of states $(l, D), (l', D')$ in region graph:
   $B^1_{(l,D)(l',D')} := B^{init}_{(l,D)(l',D')}$

3. For each pair of states $(l, D), (l', D')$ in region graph such that $(l', D'') \to (l', D')$:

$$B^{i+1}_{(l,D)(l',D')} := B^i_{(l,D)(l',D')} \cup B^{step}(B_{(l,D)(l',D')}, (l'', D'') \to (l', D'))$$

4. If some $B^{i+1}_{(l,D)(l',D')} \neq B^i_{(l,D)(l',D')}$ then $i := i + 1$ and go to step 3.

Figure 1: Algorithm for computing reachability relations

4 Non-emptiness of ω-language in Sampled Semantics

In this section we use the characterization of reachability relations (Theorem 3.1) to prove our main results: decidability of the existence of ϵ such that the (ω-)language of A is empty in sampled semantics with ϵ as the sampling period.

Examples in Figure 2 demonstrate that there are timed automata such that $L_\omega(A)$ is non-empty whereas for all ϵ the language $L^\epsilon_\omega(A)$ is empty. Non-emptiness is an important problem, because verification of liveness properties can be reduced to checking non-emptiness of an ω-language. Non-emptiness implies existence of a behavior which violates the given liveness property. But as examples in Figure 2 demonstrates, it may happen that all infinite traces are non-realizable. Therefore, on a real system the property would be satisfied.

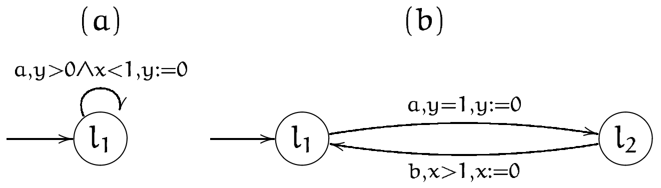

Figure 2: Difference between dense and sampled semantics (example (b) is taken from [3]).

What we really want is not the non-emptiness of $L_\omega(A)$ but non-emptiness of $L^\epsilon_\omega(A)$ for some ϵ. We show that the problem of deciding whether such an ϵ exists is decidable.

This problem was considered in a survey paper [3], where it is claimed that the problem is undecidable, with a reference to [11]. The work [11], however, deals with a slightly different problem: it is required that the timed automaton performs an action step after *each* discrete time step (this requirement is motivated by control theory).

Our proof is based on the classical region construction [2] together with the reachability relations. The region graph can be directly used for ω-language emptiness checking in dense semantics — the ω-language is non-empty if and only if there is a cycle in the region graph. This is, however, not true for sampled semantics, as illustrated by examples in Figure 2.

Intuitively, the problem is the following. The fact, that tere is a cycle in the region graph from a region (l, D) to itself means that there are some valuations $\nu, \nu' \in D$ such that $(l, \nu) \rightarrow^{+} (l, \nu')$. These valuations may be constrained, e.g., in example in Figure 2(b) the constraint on paths from state $(l_1, [x = 0, 0 < y < 1])$ to itself is that $1 > \nu'(y) > \nu(y) > 0$. In dense semantics we can have an infinite run which satisfies this constraint, but in sampled semantics we cannot. In sampled semantics we need a path $(l, \nu) \rightarrow^{+} (l, \nu')$ such that $\nu \sim_k \nu'$ (valuations may differ only in clocks above constants).

Lemma 4.1 *There exists an* ϵ *such that* $L^{\epsilon}_{\omega}(A)$ *is non-empty if and only if there exists a reachable state* (l, D) *in the region graph of* A *such that the following condition is satisfiable:*

$$\exists \nu, \nu' \in D : (\nu_0, \nu) \in C_{(l_0, D_0)(l,D)} \wedge (\nu, \nu') \in C_{(l,D)(l,D)} \wedge \nu \sim_K \nu'$$

Proof At first, suppose that the condition is satisfiable. Due to Theorem 3.1, the condition can be expressed as boolean combination of linear inequalities. The set of solutions is an union of convex polyhedrons and therefore there must be a rational solution ν, ν'. From the definition of reachability relations we get that there is a $\nu'' \sim_K \nu$ such that $(l_0, \nu_0) \rightarrow^{+} (l, \nu) \rightarrow^{+} (l, \nu'')$ in the real semantics. Since real and rational semantics are bisimilar, there is such a path in rational semantics as well. We take ϵ as the greatest common divisor of time steps on this path. Thus the path $(l_0, \nu_0) \rightarrow^{+} (l, \nu) \rightarrow^{+} (l, \nu'')$ is executable in $[\![A]\!]_{\epsilon}$ and since ν'' is bisimilar to ν (because $\nu \sim_K \nu''$) we can construct an infinite run. Therefore, $L^{\epsilon}_{\omega}(A)$ is non-empty.

On the other hand, if $L^{\epsilon}_{\omega}(A)$ is non-empty then there is an infinite run $(l_0, \nu_0) \rightarrow (l_1, \nu_1) \rightarrow (l_2, \nu_2) \rightarrow \ldots$. Since $\sim_K$ has a finite index (over sampled semantics) there must be i, j such that $l_i = l_j, \nu_i \sim_K \nu_j$. These valuations demonstrate the satisfiability of the condition. $\square$

Now, we can easily prove the main result :

Theorem 4.2 *Let A be a timed automaton. The problem of deciding whether $\bigcup_{\epsilon} L^{\epsilon}_{\omega}(A) \neq \emptyset$ is decidable.*

Proof The result now directly follows from Lemma 4.1, since the condition given in this lemma can be expressed by a clock difference relation (due to Theorem 3.1) and satisfiability of a clock difference relation can be decided (it is a special case of linear programming). □

5 Dense vs. Sampled Semantics

In this section, we present a set of results about relations between dense time semantics and sampled semantics of timed systems showing the limits of using discrete time verification methods for the dense time. We start with relations between real and rational semantics as it creates a connection between dense and sampled semantics. For timed automata, real and rational semantics are clearly bisimilar. For stopwatch automata, however, we can guarantee only trace equivalence — we show that there exists an SWA which has infinite traces realizable in real semantics, but not in rational one.

Lemma 5.1 *Let A be an SWA. Then $[\![A]\!]_{\mathbb{Q}_0^+}$ is trace equivalent to $[\![A]\!]_{\mathbb{R}_0^+}$.*

Proof Let us consider a run π in $[\![A]\!]_{\mathbb{R}_0^+}$. We can consider the delays on this run as parameters $\delta_1, \ldots, \delta_n$. The set of values of these parameters, which enable execution through the same sequence of location and over the same trace is described by a system of linear inequalities in $\delta_1, \ldots, \delta_n$ — these inequalities are obtained by substituting sums of $\delta_1, \ldots, \delta_n$ for $v(x)$ in guards. The set of solutions of this system of linear inequalities is a non-empty convex polyhedron and it has a rational solution. Therefore, there exists a run π' in $[\![A]\!]_{\mathbb{Q}_0^+}$ over the same trace as π. □

Lemma 5.2 *There is an SWA A such that $[\![A]\!]_{\mathbb{Q}_0^+}$ is* not *infinite trace equivalent to $[\![A]\!]_{\mathbb{R}_0^+}$.*

Proof[sketch] A skeleton of the example illustrating this observation is given in Figure 3. The operations $x = x - 1$ and $x = 2 \cdot x$ are not valid operations of SWA, but can be simulated using several locations and (stopwatch) clocks (see, e.g., [17]). The automaton in the first step non-deterministically chooses a value between 0 and 2 and then it accepts a sequence of a, b corresponding to the binary expansion of the chosen value. Note, that

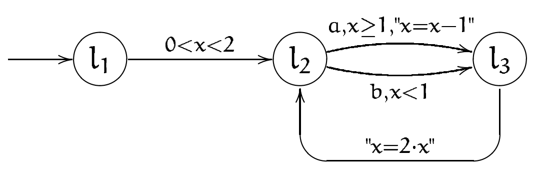

Figure 3: Stopwatch automaton for binary expansion. Clock x is stopped in the locations l_2 and l_3.

for this automaton, there is no countably branching LTS which would be infinite trace equivalent to $[\![A]\!]_{\mathbb{R}_0^+}$. □

Now we study relations between dense time and sampled semantics. We show that for a closed TA we can guarantee the simulation equivalence (i.e., that there is an ϵ such that $[\![A]\!]_{\mathbb{R}_0^+}$ is simulation equivalent to $[\![A]\!]_\epsilon$), but not bisimilarity. For general TA (as well as for SWA) the best what we can guarantee is the reachability equivalence (i.e., that there is an ϵ such that $[\![A]\!]_{\mathbb{R}_0^+}$ is reachability equivalent to $[\![A]\!]_\epsilon$).

Lemma 5.3 *Let* A *be a* closed *TA and* ϵ *is the greatest common divisor of constants in* A*. Then* $[\![A]\!]_\epsilon$ *is simulation equivalent to* $[\![A]\!]_{\mathbb{R}_0^+}$*.*

Proof Let A' is an automaton obtained from A by dividing all constants by ϵ. Then $[\![A]\!]_\epsilon$ is bisimilar to $[\![A']\!]_1$. Therefore, it is sufficient to prove the claim for $\epsilon = 1$.

Let ν be a clock valuation and $[\nu]$ denote a valuation such that $[\nu](x) = \lfloor \nu(x) \rfloor$ or $[\nu](x) = \lceil \nu(x) \rceil$ for all clocks x. Consider the relation:

$$S = \{(\nu, [\nu]) \mid \forall x, y.(([\nu](x) = \lfloor \nu(x) \rfloor) \wedge ([\nu](y) = \lceil \nu(y) \rceil)) \Rightarrow \mathsf{fr}(\nu(x)) < \mathsf{fr}(\nu(y))\}$$

Informally, $(\nu, \nu') \in S$ if and only if ν' is a corner point of a region containing ν. If A is closed then S is a simulation relation on $[\![A]\!]_{\mathbb{R}_0^+}$ and $[\![A]\!]_{\mathbb{Z}_0^+}$.

We show that S is a simulation relation. If a guard is enabled for ν and $(\nu, \nu') \in S$ then this guard is enabled also for ν', since A is closed. If A performs an action step resetting clocks in Y then obviously $(\nu[Y := 0], \nu'[Y := 0]) \in S$. We show that S is preserved by two special delay steps (immediate time successor) such that any delay step can be composed of them.

The first delay step changes a valuation where some clocks have their fractional part equal to zero to a valuation where all clocks have non-zero fractional parts, but the integral parts are the same. If A performs such a delay step in dense time semantics, we do not do anything in the sampled time semantics.

The second delay step changes a valuation where all fractional parts are non-zero and x has the greatest fractional part to a valuation ν_{delay} where $\nu_{delay}(x) = \lceil \nu(x) \rceil$. If $\nu'(x) = \lceil \nu(x) \rceil$ then $\nu'_{delay} = \nu'$, otherwise $\nu'_{delay} = \nu' + 1$. Note, that $(\nu_{delay}, \nu' + 1) \in S$ because x had a greatest fractional part and thus for all other clocks y it holds that $\nu'(y) = \lfloor \nu(y) \rfloor$.

$\square$

Lemma 5.4 *There is a closed TA A such that $[\![A]\!]_{\mathbb{R}_0^+}$ is* not *bisimilar to $[\![A]\!]_\epsilon$ for any ϵ.*

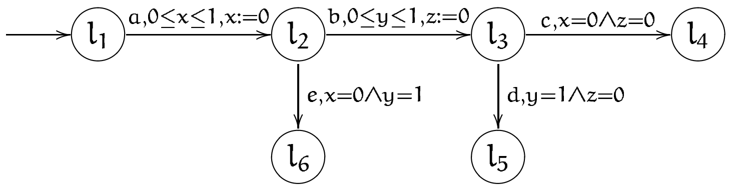

Figure 4: An automaton for which there is no ϵ such that discrete and dense semantics are bisimilar.

Proof Figure 4 shows an automaton for which there is no ϵ such that dense time and sampled semantics are bisimilar. For the proof we use a characterization of bisimulation in terms of a game between Challenger and Defender. We assume that ϵ divides 1 (i.e., there is an $n \in \mathbb{N}$ such that $n \cdot \epsilon = 1$). Otherwise, the automaton could be rescaled. Consider the following play of the bisimulation game. Let Challenger plays with the sampled semantics, delays for $1 - \epsilon$ and then takes a transition. Defender can delay for $0 \le \delta < 1$ and then take a transition. If Defender delays for 1 time unit then Challenger will take e transition, which Defender cannot take.

Now Challenger plays with dense time semantics, delays for $(1 - \delta)/2$ and then takes b transition. Defender can either delay for 0 or for ϵ and then take b transition. Challenger plays with sampled semantics again in the next step, delays for 0 and takes a transition according to the previous move of Defender. If Defender delayed for 0 then Challenger takes c transition, otherwise he takes d transition. Defender has no answer. $\square$

Lemma 5.5 *Let A be an SWA. Then there exists ϵ such that $[\![A]\!]_{\mathbb{R}_0^+}$ is reachability equivalent to $[\![A]\!]_\epsilon$.*

Proof From Lemma 5.1 we have that for each reachable action a there is a finite run π_a which contains action a and which has only rational delays. Let ϵ_a be the greatest common divisor of all delays on π_a. Let ϵ be the greatest common divisor of all ϵ_a where a is a reachable action. Then, clearly, each action is reachable in $[\![A]\!]_\epsilon$ if and only if it is reachable in $[\![A]\!]_{\mathbb{R}_0^+}$. $\square$

Lemma 5.6 *There is a TA A such that $[\![A]\!]_{\mathbb{R}_0^+}$ is* not *trace equivalent to $[\![A]\!]_\epsilon$ for any ϵ.*

Such an automaton could be easily obtained by enabling *zeno* behavior (arbitrary number of events in finite time), see Figure 2(a). Zeno behavior cannot be obtained in the sampled semantics. But there are also non-zeno automata which are not trace equivalent in dense and sampled domains, see Figure 2(b).

All results are summarized in Table 1. For the sampled semantics, a given result means that there exists an ϵ such that the given equivalence is guaranteed. In the case of closed TA, such ϵ can be easily constructed from the syntax of the automaton (as the greatest common divisor of all constants). For general TA, such ϵ can be constructed, but it requires to explore the region graph corresponding to the automaton. For SWA, such ϵ cannot be constructed algorithmically (because we do not know which actions are reachable).

6 Reachability Problem for Stopwatch Automata in Sampled Semantics

The reachability problem is to determine, for a given automaton A and an action a, whether a is reachable in $[\![A]\!]$ (for a given semantics). This is a fundamental problem, because verification of the most common type of properties (safety properties) can be reduced to the reachability problem. It is well-known that for timed automata the problem is PSPACE-complete and that the complexity depends neither on the time domain which we use nor on the choice of the type of constraints (diagonal-free, non-strict) [2]. The type of constraints becomes important if we allow more general updates in the reset operation [8].

Here we show that for stopwatch automata the choice of the time domain and the type of constraints are important. With dense semantics, the problem is known to be undecidable even for diagonal-free constraints and one stopwatch [17]. We show that in sampled semantics, the problem is decidable for SWA with diagonal-free constraints. However, if we allow diagonal constraints, the reachability problem becomes undecidable again. We have to use a different reduction than in the dense case, but, surprisingly, only one stopwatch suffices even in the case of sampled semantics.

Lemma 6.1 *Let A be a diagonal-free SWA and ϵ a given sampling period. Then the reachability problem in sampled semantics $[\![A]\!]_\epsilon$ is PSPACE-complete.*

Proof We use a standard 'normalization' approach — it is easy to check that the relation $\sim_K$ induces bisimulation on $[\![A]\!]_\epsilon$ (K is the largest constant occurring in guards) for a diagonal-free SWA. We can easily obtain unique representant of each bisimulation class (by 'normalizing' all clock values larger than K to value $K + 1$) and thus we can easily perform the search over the bisimulation collapse.

Complexity: PSPACE-membership follows from the algorithm (search in an exponential graph can be done in polynomial space), PSPACE-hardness follows from PSPACE-hardness for timed automata. □

Lemma 6.2 *Let A be an SWA with one stopwatch. Then the reachability problem in sampled semantics $[\![A]\!]_\epsilon$ is undecidable.*

Proof We show the undecidability by reduction from the halting problem for a two counter machine M. Since this is usual approach in this area (see e.g., [17, 11, 8]), we just describe the main idea — how to encode counter values and perform increment/decrement.

The value of a counter i is represented as the difference between two clocks: $x_i - y_i$. Before the start of the simulation of M the simulating SWA non-deterministically guesses the maximal value c of the counters during a computation of M and sets the stopwatch to the value $c+1$. From this moment on the stopwatch is stopped for the rest of the computation with the value $c + 1$.

Values of the clocks are kept in the interval $[0, c + 1]$ all the time. Whenever the value of a clock reaches $c + 1$, the clock is reseted. Testing the value of a counter for zero is straightforward: just testing $x_i = y_i$. Decrementing a counter i is performed by postponing the reset of a clock x_i by 1 time unit. Incrementing a counter i is performed by postponing the reset of a clock y_i by 1 time unit. During the increment we have to check for an 'overflow' — if the difference $x_i - y_i$ equals to c and we should perform increment then it means that the initial non-deterministic guess was wrong and the simulation should not continue.

Note that the stopwatch is used in a very limited fashion: it is stopped once and then keeps a constant value. □

7 Future Work

Sampled semantics gives natural under-approximations of timed systems, where the choice of sampling period ϵ can nicely tune the size of a state space of the under-approximation. This makes sampled semantics plausible as a base for under-approximation refinement scheme. This scheme starts with coarse-grained under-approximation of the systems and refines it until an error is found or the approximation is exact [16, 22]. This approach is suitable particularly for falsification (defect detection).

From a theoretical point of view, a successful application of the under-approximation refinement scheme based on sampled semantics has many obstacles. Complexity (decidability) of verification problems remains usually the same for sampled semantics as it is for dense semantics. Moreover, it is even not possible to efficiently decide whether ϵ-sampled and dense semantics are equivalent. Nevertheless, we believe that for practical examples this approach can give better results than classical complete methods (at least for the defect detection). One possible direction for future work is an experimental evaluation of this approach on real-life case studies.

Another direction for future work is provided by the decidability result for non-emptiness of ω-language in some sampled semantics. The result leads to standard questions about complexity of the problem and about the existence of a practically feasible algorithm.

References

[1] Y. Abdeddaïm and O. Maler. Preemptive job-shop scheduling using stopwatch automata. In *Proc. of Tools and Algorithms for Construction and Analysis of Systems (TACASt'02)*, volume 2280 of *LNCS*, pages 113–126. Springer, 2002.

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

[3] R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *LNCS*, pages 1–24. Springer, 2004.

[4] E. Asarin, M. Bozga, A. Kerbrat, O. Maler, A. Pnueli, and A. Rasse. Data-structures for the verification of timed automata. In *Proc. of Hybrid and Real-Time Systems (HART'97)*, pages 346–360. Springer, 1997.

[5] E. Asarin, O. Maler, and A. Pnueli. On discretization of delays in timed automata and digital circuits. In *Proc. of Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *LNCS*, pages 470–484. Springer, 1998.

[6] D. Beyer. Improvements in BDD-based reachability analysis of timed automata. In *Proc. of Formal Methods Europe (FME 2001)*, volume 2021 of *LNCS*, pages 318–343. Springer, 2001.

[7] D. Bosnacki. Digitization of timed automata. In *Proc. of Formal Methods for Industrial Critical Systems (FMICS'99)*, pages 283–302, 1999.

[8] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Are timed automata updatable? In *Proc. of Computer Aided Verification (CAV 2000)*, volume 1855 of *LNCS*, pages 464–479. Springer, 2000.

[9] M. Bozga, O. Maler, A. Pnueli, and S. Yovine. Some Progress in the Symbolic Verification of Timed Automata. In *Proc. of Computer Aided Verification (CAV'97)*, volume 1254 of *LNCS*, pages 179–190. Springer, June 1997.

[10] M. Bozga, O. Maler, and S. Tripakis. Efficient verification of timed automata using dense and discrete time semantics. In *Proc. of Charme'99*, volume 1703 of *LNCS*, pages 125–141. Springer, 1999.

[11] F. Cassez, T. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. of the Hybrid Systems: Computation and Control*, volume 2289 of *LNCS*, pages 134–148. Springer, 2002.

[12] F. Cassez and K. G. Larsen. The impressive power of stopwatches. In *Proc. of Conference on Concurrency Theory (CONCUR 2000)*, number 1877 in LNCS, pages 138–152. Springer, 2000.

[13] H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *Proc. of Conference on Concurrency Theory (CONCUR'99)*, volume 1664 of *LNCS*, pages 242–257. Springer, 1999.

[14] C. Dima. Computing reachability relations in timed automata. In *Proc. of Symp. on Logic in Computer Science (LICS 2002)*. IEEE Computer Society Press, 2002.

[15] A. Gollu, A. Puri, and P. Varaiya. Discretization of timed automata. In *Proc. of Conferene on Decision and Control*, pages 957–958, 1994.

[16] O. Grumberg, F. Lerda, O. Strichman, and M. Theobald. Proof-guided underapproximation-widening for multi-process systems. In *Proc. of Principles of programming languages (POPL'05)*, pages 122–131. ACM Press, 2005.

[17] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proc. of ACM symposium on Theory of computing (STOC'95)*, pages 373–382. ACM Press, 1995.

[18] T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proc. of Colloquium on Automata, Languages and Programming (ICALP'92)*, volume 623 of *LNCS*, pages 545–558. Springer, 1992.

[19] P. Krčál and W. Yi. Decidable and undecidable problems in schedulability analysis using timed automata. In *Proc. of Tools and Algorithms for Construction and Analysis of Systems (TACAS'04)*, volume 2988 of *LNCS*, pages 236–250. Springer, 2004.

[20] K. G. Larsen and W. Yi. Time-abstracted bisimulation: Implicit specifications and decidability. *Information and Computation*, 134(2):75–101, 1997.

[21] J. Ouaknine and J. Worrell. Revisiting digitization, robustness, and decidability for timed automata. In *Proc. of IEEE Symp. on Logic in Computer Science (LICS 2003)*, pages 198–207. IEEE Computer Society Press, 2003.

[22] C. Pasareanu, R. Pelánek, and W. Visser. Concrete search with abstract matching and refinement. In *Proc. of Computer Aided Verification (CAV 2005)*, LNCS. Springer, 2005. To appear.

[23] Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.

[24] M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. In *Proc. of Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *LNCS*, pages 296–310. Springer, 2004.