On Sampled Semantics of Timed Systems

Pavel Krčál*[1] and Radek Pelánek**[2]

[1] Uppsala University, Sweden.
Email: pavelk@it.uu.se.
[2] Masaryk University Brno, Czech Republic.
Email: xpelanek@fi.muni.cz.

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). We investigate the relations between real semantics and sampled semantics with respect to different behavioral equivalences. Also, we study decidability of reachability problem for stopwatch automata with sampled semantics. Finally, our main technical contribution is decidability of non-emptiness of a timed automaton ω-language in some sampled semantics (this problem was previously wrongly classified as undecidable). For the proof we employ a novel characterization of reachability relations between configurations of a timed automaton.

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) [1]; some results are also shown for stopwatch automata (SWA) [12] — 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 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 [6, 5]. 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

* Partially supported by the European Research Training Network GAMES.
** Partially supported by the Grant Agency of Czech Republic grant No. 201/03/0509 and by the Academy of Sciences of Czech Republic grant No. 1ET408050503.

implementability, e.g., discussed in [20, 18]. 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 [2]. The same problem was studied in the control setting with slightly different sampled behavior in [7]. In this setting, an automaton is a model of a controller with a periodic control loop which reacts on input data by a control action. Therefore, an automaton has to perform an action at *every* sampled time point which makes the problem undecidable even for (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 [8] and $2n$-automata [9]. Our novel representation is based on simple linear (in)equalities (comparisons of clock differences). This representation is of an 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 time points at which these actions are taken. There has been a considerable amount of work related to discretization issues and verifying dense time properties using discrete time methods, e.g. [13, 15, 16, 10, 3]. The main difference is that usually a fixed sampling rate and trace equivalence are considered.

Finally, we summarize the (un)decidability of the reachability problem in timed systems (Table 1.). 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.

Table 1. The first table gives a summary of the equivalences: each field gives the relation to real semantics. The second table gives a summary of decidability results for the reachability problem.

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

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 behavioral equivalences on LTSs and equivalences on valuations. Note that all languages and equivalences that we consider are untimed (this is sometimes denoted as $untime(L(A))$ and time abstracted equivalences in the literature).

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 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 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').

3 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. This follows directly from the region construction since each region contains at least one rational valuation. For stopwatch automata, however, we can guarantee only trace equivalence — we show that there exists a SWA which has infinite traces realizable in real semantics, but not in rational one.

Lemma 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 $\nu(x)$ 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. $\square$

Lemma 2. *There exist an SWA A such that $[\![A]\!]_{\mathbb{Q}_0^+}$ is* not *infinite trace equivalent to $[\![A]\!]_{\mathbb{R}_0^+}$.*

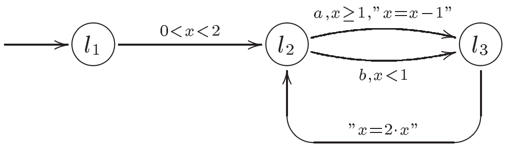

Fig. 1. Stopwatch automaton for binary expansion. Clock x is stopped in l_2, l_3.

Proof (sketch). A skeleton of the example illustrating this observation is given in Fig. 1. 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., [12]). The automaton in the first step nondeterministically chooses a value between 0 and 2 and then it accepts the sequence of a, b corresponding to binary expansion of the chosen value. Note, that for this automaton, there is no countably branching LTS which would be infinite trace equivalent to $[\![A]\!]_{\mathbb{R}_0^+}$. $\square$

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 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. See [14] for the proof.

Lemma 4. *There exits a closed TA A such that $[\![A]\!]_{\mathbb{R}_0^+}$ is* not *bisimilar to $[\![A]\!]_\epsilon$ for any ϵ.*

Proof. Fig. 2 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 [19]. 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 $0 \leq \delta < 1$ 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 d transition, otherwise he takes c transition. Defender has no answer. $\square$

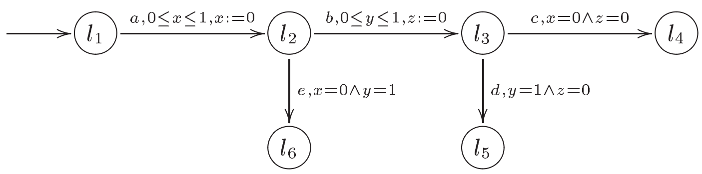

Fig. 2. An automaton for which there is no ϵ such that discrete and dense semantics are bisimilar.

Lemma 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 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 $[\![A]\!]_{\mathbb{R}_0^+}$. □

Lemma 6. *There exists a TA A such that $[\![A]\!]_{\mathbb{R}_0^+}$ is* not *trace equivalent to $[\![A]\!]_\epsilon$ for any ϵ.*

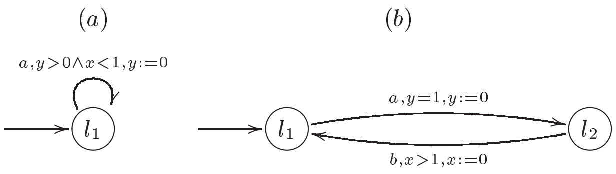

Fig. 3. Difference between dense and sampled semantics (example (b) is taken from [2]).

Such an automaton could be easily obtained by enabling *zeno* behavior (arbitrary number of events in finite time), see Fig. 3(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 Fig. 3(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 a 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).

4 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) [1]. The type of constraints becomes important if we allow more general updates in the reset operation [4].

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 [12]. We show that in sampled semantics, the problem is decidable for diagonal-free constraints. However, if we allow diagonal constraints, the reachability problem is again undecidable. 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 7. *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 representative 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 8. *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., [12, 7, 4]), 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 of two clocks: $x_i - y_i$. Before the start of the simulation of M the simulating SWA nondeterministically guesses the maximal value c of 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 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 nondeterministic 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 it keeps a constant value. □

5 Non-emptiness of ω-language in Sampled Semantics

Examples in Fig. 3 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 a given liveness property. But as examples in Fig. 3 demonstrate, it may happen that all infinite traces are non-realizable. Therefore, the property would be satisfied on a real system.

What we really want is non-emptiness of $L^\epsilon_\omega(A)$ for some ϵ instead of non-emptiness of $L_\omega(A)$. We show that the problem of deciding whether such an ϵ exists is decidable. This problem was considered in a survey paper [2], where it is claimed that the problem is undecidable, with a reference to [7]. The work [7], however, deals with a slightly different problem: it is required that the timed automaton performs action step after *every* discrete time step (this requirement is motivated by control theory).

Theorem 1. *Let A be a timed automaton. The problem of deciding whether $\bigcup_\epsilon L^\epsilon_\omega(A) \neq \emptyset$ is decidable.*

Our proof is based on the classical region construction [1] (for the definition of region graph and for technical aspects of the proof see [14]). 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 Fig. 3.

Intuitively, the problem is the following. Existence of a cycle in the region graph from a region (l, D) to itself means that there exists some valuations $\nu, \nu' \in D$ such that $(l, \nu) \to^+ (l, \nu')$. These valuations may be constrained, e.g., in example in Fig. 3(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 $(l, \nu) \to^+ (l, \nu')$ such that $\nu \sim_k \nu'$ (valuations may differ only in clocks above constants).

To formalize this intuition, we need a notion of reachability relation, which describes exactly which valuations can be reached from a given valuation. Let $(l, D), (l, D')$ be two states in region graph. Then *reachability relation* of the pair

$(l, D), (l, D')$ is a relation on valuations $C_{(l,D)(l',D')} \subseteq D \times D'$ such that for each $\nu \in D, \nu' \in D'$:

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

We present a simple representation for reachability relations. The fact that such simple (in)equalities are sufficient to capture the reachability relations and that these relations can be computed effectively is rather interesting.

Clock difference relations (CDR) structure 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:

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

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

– If there is an immediate transition $(l, D) \rightarrow (l', D')$ then the reachability relation can be directly expressed as a CDR.
– If $(l, D) \rightarrow^+ (l'', D'') \rightarrow (l', D')$ 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').

Lemma 9. *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 2, 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 exist a rational solution ν, ν'. From the definition of reachability relations we get that there exists $\nu'' \sim_K \nu$ such that $(l_0, \nu_0) \rightarrow^+ (l, \nu) \rightarrow^+ (l, \nu'')$ in the real semantics. Since real and rational semantics are bisimilar, there exists 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 exists 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 exists i, j such that $l_i = l_j, \nu_i \sim_K \nu_j$. These valuation demonstrate the satisfiability of the condition. $\square$

With the use of Theorem 2 and Lemma 9 it is now quite straightforward to prove Theorem 1 (see [14] for technical details).

6 Future Work

Sampled semantics gives natural under-approximations of timed systems, where the choice of sampling period ϵ can nicely tune the size of the state space of an 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 [11, 17]. 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 defect detection). One possible direction for future work is an experimental evaluation of this approach on practical 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.

Acknowledgments. We thank Wang Yi and Ivana Černá for their comments on previous drafts of this paper.

References

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

3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. C. Dima. Computing reachability relations in timed automata. In *Proc. of Symp. on Logic in Computer Science (LICS 2002)*, pages 177–188. IEEE Computer Society Press, 2002.
10. A. Gollu, A. Puri, and P. Varaiya. Discretization of timed automata. In *Proc. of Conferene on Decision and Control*, pages 957–958, 1994.
11. 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.
12. 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.
13. 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.
14. P. Krčál and R. Pelánek. Reachability relations and sampled semantics of timed systems. Technical Report FIMU-RS-2005-09, Masaryk University Brno, 2005.
15. K. G. Larsen and W. Yi. Time-abstracted bisimulation: Implicit specifications and decidability. *Information and Computation*, 134(2):75–101, 1997.
16. 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.
17. 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.
18. Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.
19. C. Stirling. Local model checking games. In *Proc. of Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *LNCS*, pages 1–11. Springer, 1995.
20. 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.