Fundamenta Informaticae 143 (2016) 317–338
DOI 10.3233/FI-2016-1316
IOS Press

Relating Reachability Problems in Timed and Counter Automata

Christoph Haase[*][†]
Laboratoire Spécification et Vérification (LSV), CNRS
École Normale Supérieure (ENS) de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex, France
haase@lsv.ens-cachan.fr

Joël Ouaknine, James Worrell
Department of Computer Science
University of Oxford
Parks Rd, OX1 3QD, Oxford, United Kingdom
joel.ouaknine@cs.ox.ac.uk; james.worrell@cs.ox.ac.uk

Abstract. We establish a relationship between reachability problems in timed automata and space-bounded counter automata. We show that reachability in timed automata with three or more clocks is logarithmic-space inter-reducible with reachability in space-bounded counter automata with two counters. We moreover show the logarithmic-space equivalence of reachability in two-clock timed automata and space-bounded one-counter automata. This last reduction has recently been employed by Fearnley and Jurdziński to settle the computational complexity of reachability in two-clock timed automata.

Keywords: timed automata, counter automata, reachability, infinite-state systems

[*]Supported by the French Agence Nationale de la Recherche (ANR), ReacHard (grant ANR-11-BS02-001). Parts of this paper appeared in the author's Ph.D. thesis [1].
[†]Address for correspondence: Laboratoire Spécification et Vérification (LSV), CNRS, École Normale Supérieure (ENS) de Cachan, 61, avenue du Président Wilson, 94235 CACHAN Cedex, France

1. Introduction

Timed automata [2] and counter automata [3] are prominent infinite-state formalisms for modeling and reasoning about quantitative behavior of systems. Timed automata comprise a finite-state controller with a finite number of clocks that can be compared to constants and reset along a transition between two control locations. Counter automata on the other hand extend finite-state machines with a finite number of counters ranging over the natural numbers that can be incremented, decremented or tested for zero along a transition. *Reachability*, the problem of deciding whether there is a path connecting two given configurations in the corresponding induced transition system, is the central decision problem for both timed and counter automata. In this paper, we establish a natural correspondence between reachability in timed automata and a restricted class of counter automata, namely *bounded* counter automata. In the latter class, counters are restricted to take values from an arbitrary but fixed finite interval over the naturals, and hence bounded counter automata posses an *a priori* finite state space. However, due to binary encoding of numbers, bounded counter automata succinctly encode a state space which is exponential in the size of their description.

The main contribution of this paper is to show how runs in transition systems of timed automata can naturally be simulated in bounded counter automata, and *vice versa*. From this we show in Section 3 that reachability in k-clock timed automata with $k \geq 3$ is logarithmic-space inter-reducible with reachability in bounded two-counter automata. The emphasis and the most interesting part in this section is on the naturalness of the simulation of timed automata in bounded counter automata. A more elaborate reduction is required in Section 4, where we show that reachability in two-clock timed automata is logarithmic-space inter-reducible with reachability in bounded one-counter automata. An interesting class of bounded one-counter automata for which the precise complexity of reachability remains open is discussed in Section 5.

An extended abstract of this paper [4] appeared in the proceedings of the 6th International Workshop on Reachability Problems held in September 2012 in Bordeaux, France. Prior to this the precise computational complexity of reachability in two-clock timed automata and in bounded one-counter automata were both long-standing open problems. One of the contributions of [4] was to show that these two problems are essentially equivalent. At the 40th International Colloquium on Automata, Languages and Programming held in July 2013 in Riga, Latvia, Fearnley and Jurdziński gave a PSPACE-hardness proof, finally showing that reachability in bounded one-counter automata is PSPACE-complete. By application of our inter-reducibility result it follows that reachability in two-clock timed automata is PSPACE-complete as well [5].

A brief survey of work on reachability problems in timed and counter automata is presented in the next section. Even though Fearnley and Jurdziński's result only requires showing that reachability in bounded one-counter automata can be reduced to reachability in two-clock timed automata, we believe that the relationships established in [4] are interesting in their own right since they provide insight into the structure of reachability problems of two prominent classes of automata. Similarly, a related technical characterisation of timed automata via channel machines has proved useful in the study of robustness problems for timed automata [6, 7]. It is conceivable that our reductions may in future assist in tackling problems of a similar nature as well.

The present paper extends [4] by including all proofs omitted from [4] for reasons of space, as well as additional details at various points. In addition, we show how bounded one-counter automata relate to automata with a single integer-valued counter sign tests: a model considered by Demri and Gascon [8]. We conclude by discussing an open problem concerning one-dimensional vector addition systems.

1.1. A Short Account of the Complexity of Reachability in Timed and Counter Automata

In this section, we give a brief account of the history of the study of the computational complexity of reachability in timed and counter automata. There is a rich body of literature on this topic; the present treatment is not meant to be exhaustive.

Reachability in timed automata was shown to be decidable and PSPACE-complete in Alur and Dill's seminal paper [2]. Subsequently, a multi-parameter analysis of this problem was conducted by Courcoubetis and Yannakakis [9], who showed that reachability is PSPACE-hard already in the presence of three clocks when numbers are encoded in binary, and also PSPACE-hard when the number of clocks is unbounded and numbers are encoded in unary. The cases with fewer than three clocks were considered by Laroussinie, Markey and Schnoebelen [10], who showed that reachability for one-clock timed automata is NL-complete, and NP-hard in the presence of two clocks. However, no matching upper bound for the latter problem was given in [10]. Naves showed in his Master's thesis [11] that reachability becomes PSPACE-hard when allowing for modulo tests on clocks under the assumption that numbers are encoded in binary. This result was later refined by Göller and Lohrey [12], who showed that reachability is also PSPACE-hard when all numbers, in particular those occurring in modulo tests, are encoded in unary.

For counter automata, the earliest result is that reachability is undecidable in the presence of at least two counters [3]. For that reason, restrictions on counter automata that lead to decidable reachability problems have been widely studied in the literature. Examples include the restriction to one counter [13, 14], restricting zero-tests [15, 16, 17, 18, 19], reversal-boundedness [20, 21] or flatness [22], all of which lead to a decidable reachability problem, with complexity dropping to NP in certain cases. The complexity of reachability in the presence of one unbounded counter is NL-complete when numbers are encoded in unary, see *e.g.* [8], and NP-complete when numbers are encoded in binary [14]. To the best of our knowledge, the class of bounded counter automata introduced in this paper have nowhere been studied in full generality. The complexity of reachability in bounded counter automata with only one counter was investigated by Bouyer *et al.* in [23] in the context of weighted timed automata, where the problem was shown to be NP-hard and in PSPACE. The reduction we established in [4] trivially entails that reachability in bounded two-counter automata is PSPACE-complete in the presence of two counters. Finally, Fearnley and Jurdziński showed in [5] that reachability in bounded one-counter automata is PSPACE-hard, which established PSPACE-completeness of the whole class of bounded counter automata and of two-clock timed automata.

2. Preliminaries

In this section, we give some of the definitions that we use in the remainder of this paper. The definitions of timed automata and bounded counter automata are tailored to our needs and as simplified as possible in order to ease the reductions provided in the main text of the paper. In the case of bounded counter automata, for technical convenience we additionally introduce some syntactic sugar and observe as a side note the log-space inter-reducibility between reachability in bounded one-counter automata and one-counter automata with sign tests which have been introduced by Demri and Gascon [8].

2.1. General Notation

By $\mathbb{R}$ we denote the set of *reals*, by $\mathbb{Q}$ the set of *rationals*, by $\mathbb{Z}$ the set of *integers*, by $\mathbb{N} \stackrel{\text{def}}{=} \{n \in \mathbb{Z} : n \geq 0\}$ the set of *naturals*, by $\mathbb{N}_{>0} \stackrel{\text{def}}{=} \{n \in \mathbb{N} : n > 0\}$ the set of *strictly positive naturals*, and by $\mathbb{R}_{\geq 0} \stackrel{\text{def}}{=} \{r \in \mathbb{R} : r \geq 0\}$ the set of *positive reals*. For $i, j \in \mathbb{Z}$, $[i, j]$ denotes the interval $\{z \in \mathbb{Z} : i \leq z \leq j\}$, and $[i]$ is an abbreviation for $[1, i]$. Otherwise, interval definitions are used in the standard way over subsets of $\mathbb{R}$, *e.g.*, (i, j) defines the interval $\{r \in \mathbb{R} : i < r < j\}$. The floor function on the reals is defined in the standard way, *i.e.*, $\lfloor r \rfloor \stackrel{\text{def}}{=} \max\{z \in \mathbb{Z} : z \leq r\}$. Given $M \subseteq \mathbb{R}$ and $r \in \mathbb{R}$, we denote by rM the set $\{rm : m \in M\}$, and $M + r$ is the set $\{m + r : m \in M\}$. Throughout this paper, we assume integers to be encoded in their natural binary encoding, and for any $z \in \mathbb{Z}$ denote by $size(z)$ the number of symbols required to represent z.

2.2. Transition Systems

A *transition system* is a tuple $T = (S, \rightarrow)$, where S is the set of *states* and $\rightarrow \subseteq S \times S$ is the *transition relation*. Given $s, t \in S$, we write $s \rightarrow t$ whenever $(s, t) \in \rightarrow$ and denote by $\rightarrow^*$ the reflexive transitive closure of $\rightarrow$. An *s-t path* π in T is a sequence of states $\pi : s_1, \ldots, s_n$ such that $s_1 = s$, $s_n = t$ and $s_i \rightarrow s_{i+1}$ for all $i \in [n-1]$. Given $s, t \in S$, *reachability* is to decide the existence of an s-t path in T, *i.e.*, whether $s \rightarrow^* t$.

2.3. Timed Automata

Let X be a finite set of *clock variables*. A *clock valuation* is a mapping $\vartheta : X \rightarrow \mathbb{R}_{\geq 0}$; we denote by $CV(X)$ the set of all *clock valuations*. Given $r \in \mathbb{R}_{\geq 0}$, we denote by $\vartheta + r$ the clock valuation defined by $(\vartheta + r)(x) = \vartheta(x) + r$ for all $x \in X$. An *atomic clock constraint* is a term of the form $x \sim n$, where $x \in X$, $\sim \in \{<, \leq, =, \neq, \geq, >\}$ and $n \in \mathbb{N}$. A *clock constraint* ϕ is a finite conjunction of atomic clock constraints $\phi = x_1 \sim n_1 \wedge \ldots \wedge x_m \sim n_m$. The set of all clock constraints over clocks X is denoted by $CC(X)$. A clock valuation ϑ maps $x \sim n$ to a Boolean value $\vartheta(x) \sim n$ and hence a clock constraint ϕ to a Boolean value. We write $\vartheta \models \phi$ whenever ϑ evaluates ϕ to true.

In this paper, a *k-clock timed automaton* is a tuple $\mathcal{A} = (Q, X, \Delta, \xi)$, where Q is a finite set of *control locations*, X is a set of k clock variables, $\Delta \subseteq Q \times Q$ is the *transition relation*, and $\xi : \Delta \rightarrow CC(X) \times 2^X$ is the *transition labeling function*. The map ξ assigns to each transition a clock constraint representing a pre-condition of the transition and a set of clocks to be *reset* to zero when the transition is taken. Given $x \in X$, the set of *x-constants* C_x comprises 0 and those $n \in \mathbb{N}$ such that an atomic clock constraint $x \sim n$ occurs as a conjunct in a clock constraint of some transition of $\mathcal{A}$. The set $C(\mathcal{A})$ of *configurations of* $\mathcal{A}$ is $Q \times CV(X)$. For brevity we write $q(\theta)$ for a configuration (q, ϑ). The *size* of a timed automaton is $|\mathcal{A}| \stackrel{\text{def}}{=} |Q| + |\Delta| \max\{size(n) + 1 : n \in C_x, x \in X\}$.

A timed automaton induces a transition system $T(\mathcal{A}) = (S_{\mathcal{A}}, \rightarrow_{\mathcal{A}})$ where $S_{\mathcal{A}} = C(\mathcal{A})$ and $q(\vartheta) \rightarrow_{\mathcal{A}} q'(\vartheta')$ iff one of the following conditions holds:

(i) $q = q'$ and there exists $d \in \mathbb{R}_{\geq 0}$ such that $\vartheta' = \vartheta + d$ (subsequently called *delay transitions*);

(ii) $(q, q') \in \Delta$, $\xi(q, q') = (\phi, Y), \vartheta \models \phi$ and ϑ' is such that $\vartheta'(y) = 0$ for every $y \in Y$ and $\vartheta'(x) = \vartheta(x)$ for every $x \in X \setminus Y$ (subsequently called *discrete transitions*).

Reachability for a k-clock timed automaton $\mathcal{A}$ is to decide $C \to_{\mathcal{A}}^{*} C'$ for given configurations $C, C' \in C(\mathcal{A}) \cap (Q \times \mathbb{N}^k)$ with integer-valued clocks.

2.4. Bounded Counter Automata

Let $k \in \mathbb{N}$ and $\mathsf{Op} \overset{\text{def}}{=} \{add_i(z) : i \in [k], z \in \mathbb{Z}\}$ be a set of *counter operations*. A *bounded k-counter automaton* is a tuple $\mathcal{A} = (Q, \Delta, \mathbf{b}, \xi)$, where Q is a finite set of *control locations*, $\Delta \subseteq Q \times Q$ is the *transition relation*, $\mathbf{b} = (b_1, \ldots, b_k) \in (\mathbb{N}_{>0})^k$ is a vector of *bounds*, and $\xi : \Delta \to \mathsf{Op}$ is the *transition labeling function*. The set $C(\mathcal{A})$ of *configurations* of $\mathcal{A}$ is $Q \times [0, b_1] \times \cdots \times [0, b_k]$; again we write $q(n_1, \ldots, n_k)$ or $q(\mathbf{n})$ to denote individual configurations. We call b_i the *bound* of counter i. The *size* of a bounded k-counter automaton is $|\mathcal{A}| \overset{\text{def}}{=} |Q| + |\Delta| \max\{size(b_i) : i \in [k]\}$.

A bounded k-counter automaton $\mathcal{A}$ induces a transition system $T(\mathcal{A}) = (S_{\mathcal{A}}, \to_{\mathcal{A}})$, where $S_{\mathcal{A}} = C(\mathcal{A})$ and there is a transition $q(n_1, \ldots, n_k) \to_{\mathcal{A}} q'(n'_1, \ldots, n'_k)$ iff both of the following hold:

(i) $(q, q') \in \Delta$; and

(ii) if $\xi(q, q') = add_i(z)$ then $n'_i = n_i + z$ and $n'_j = n_j$ for all $j \neq i$.

The *Reachability Problem* for bounded k-counter automata is to decide whether $C \to_{\mathcal{A}}^{*} C'$ for given configurations $C, C' \in C(\mathcal{A})$.

We conclude this section by noting that bounded counter automata can be viewed as bounded vector addition systems with states (VASS) [24].

2.4.1. Syntactic Extensions of Bounded Counter Automata

Without loss of generality we may assume that transitions of k-counter automata are endowed with guards which compare the counters to natural numbers. Formally, we can extend the set of operations to additionally contain operations $counter_i \sim n$, where $\sim \in \{<, \leq, =, \geq, >\}$, with the following semantics: for every transition $(q, q') \in \Delta$ such that $\xi(q, q') = counter_i \sim n$, $q(n_1, \ldots, n_k) \to_{\mathcal{A}} q'(n'_1, \ldots, n'_k)$ iff $n_i \sim n$ and $n_j = n'_j$ for all $j \in [k]$. It is not difficult to see that reachability in any such bounded k-counter automaton with an extended set of operations can be reduced in logarithmic space to reachability in a bounded k-counter automaton. For example, a transition $(q, q') \in \Delta$ with label $\xi(q, q') = counter_i < n$ can be simulated as follows:

- replace (q, q') with two new transitions (q, q'') and (q'', q'), where q'' is a fresh control location; and

- label (q, q'') with $add_i(b_i - n + 1)$ and (q'', q') with $add_i(-b_i + n - 1)$, where b_i is the bound of counter i.

The construction for the remaining relational symbols follows analogously.

Finally, we define a further generalisation that allows for the counters of a bounded counter automaton to take values from bounded intervals $(1/n)\mathbb{Z} \subseteq \mathbb{Q}$, $n \in \mathbb{N}_{>0}$. Moreover, this generalisation allows for adding and subtracting integer multiples of $1/n$ to and from the counters. Formally, for $n \in \mathbb{N}_{>0}$, such a bounded counter automaton is a tuple $\mathcal{A} = (Q, \Delta, \mathbf{b}, n, \xi)$ as above with $\mathbf{b} = (b_1, \ldots, b_k) \in ((1/n)\mathbb{N}_{>0})^k$, and its set of operations consists of operations $add_i(r)$ such that $r \in (1/n)\mathbb{Z}$. The set of configurations of $\mathcal{A}$ is $Q \times I_1 \times \cdots \times I_k$, where $I_j = \{r \in (1/n)\mathbb{Z} : -b_j \leq r \leq b_j\}$, and $T(\mathcal{A})$ is defined in the obvious way. An instance of a reachability problem in such a bounded counter automaton $\mathcal{A}$ can then be reduced in logarithmic space to reachability in a bounded counter automaton $\mathcal{A}'$ by the following procedure:

- replace each bound b_i with $2nb_i$; and
- replace each operation $add_i(r)$ with $add_i(nr)$.

It is then easily shown by induction on the length of the path that for all $z_1, \ldots, z_k \in nI_1 \times \cdots \times nI_k$,

$$\begin{aligned} & q((1/n)z_1, \ldots, (1/n)z_k) \to^*_{\mathcal{A}} q'((1/n)z'_1, \ldots, (1/n)z'_k) \\ \iff & q(z_1 + nb_1, \ldots, z_k + nb_k) \to^*_{\mathcal{A}'} q(z'_1 + nb_1, \ldots, z'_k + nb_k). \end{aligned}$$

2.4.2. Relationship to One-$\mathbb{Z}$-counter Automata with Sign Tests

In [8], Demri and Gascon consider reachability in one-$\mathbb{Z}$-counter automata with sign tests, for which they show that reachability is NP-hard and in PSPACE provided numbers are encoded in binary [8, Thm. 6 and the remarks below]. Formally, a one-$\mathbb{Z}$-counter automaton with sign tests is a tuple $\mathcal{A} = (Q, \Delta, \xi, \tau)$, where Q and Δ are defined as for bounded counter automata above, $\xi : \Delta \to \{add(z) : z \in \mathbb{Z}\}$, and $\tau : \Delta \to \{<, \leq, =, \neq, \geq, >\} \cup \{\mathbf{true}\}$ is a *transition guard* which allows the counter value to be compared to zero. The set of configurations of $\mathcal{A}$ is $C(\mathcal{A}) = Q \times \mathbb{Z}$, and $q(z) \to_{\mathcal{A}} q'(z')$ iff $(q, q') \in \Delta$, $\xi(q, q') = add(y)$, $z' = z + y$, $\tau(q, q') = \sim$ and $z \sim 0$ if $\sim \neq \mathbf{true}$. In particular note that the state space of a bounded one-$\mathbb{Z}$-counter automaton is infinite.

Here, we show that reachability in bounded one-counter automata is logarithmic-space reducible to reachability in one-$\mathbb{Z}$-counter automata with sign tests. This observation together with the PSPACE lower bound obtained by Fearnley and Jurdziński [5] for reachability in bounded one-counter automata then allows us to observe the PSPACE-completeness of reachability in one-$\mathbb{Z}$-counter automata.

Lemma 2.1. Reachability in bounded one-counter automata is logarithmic-space reducible to reachability in one-$\mathbb{Z}$-counter automata with sign tests.

Proof:
The idea is straightforward: we use transition guards in order to ensure that for a given bounded one-counter automaton $\mathcal{A} = (Q, \Delta, b, \xi)$, the counter always stays in the interval $[0, b]$. Formally, a one-$\mathbb{Z}$-counter automaton $\mathcal{A}' = (Q', \Delta', \xi', \tau')$ can be obtained from $\mathcal{A}$ as follows: replace each transition $(q, q') \in \Delta$ labeled with $add(z)$ by three consecutive transitions with fresh intermediate control locations that perform the following sequence of actions:

- add z to the counter and test that resulting value is non-negative;
- subtract b from the counter and test that the resulting value is at most zero;
- add b to the counter.

It is then easily established by induction on the length of a run that $q(n) \to^*_{\mathcal{A}} q'(n')$ iff $q(n) \to^*_{\mathcal{A}'} q'(n')$ for each pair of control locations $q, q' \in Q$ and counter values $n, n' \in [0, b]$. □

Corollary 2.2. Reachability in one-$\mathbb{Z}$-counter automata with sign tests is PSPACE-complete.

Remark 2.3. Note that in [8] it is also shown that if there is a run between two configurations of a one-$\mathbb{Z}$-counter automaton with sign tests $\mathcal{A}$ then there is one for which the maximum absolute value of the counter occurring along the run can be bounded by $p(|\mathcal{A}|)$ for some fixed polynomial p that is independent of $\mathcal{A}$. Hence, an adaptation of the construction provided in Section 2.4.1 can be used in order to reduce reachability in one-$\mathbb{Z}$-counter automata with sign tests to reachability in bounded one-counter automata.

3. The general case

In this section we prove the following theorem.

Theorem 3.1. Reachability in k-clock timed automata with $k \geq 3$ is logarithmic-space inter-reducible with reachability in bounded two-counter automata.

The proof of the theorem comprises three parts. We show that

(i) reachability in bounded k-counter automata with $k \geq 3$ can be reduced to reachability in bounded two-counter automata;

(ii) reachability in bounded two-counter automata can be reduced to reachability in three-clock timed automata; and

(iii) reachability in k-clock timed automata with $k \geq 3$ can be reduced to reachability in bounded $(2k+2)$-counter automata, which by (i) implies that this problem is reducible to reachability in bounded two-counter automata.

We describe each reduction in a separate section below.

3.1. Reduction (i)

In this section, we how that reachability in bounded k-counter automata with $k \geq 3$ can be reduced to reachability in bounded two-counter automata. Let $\mathcal{A} = (Q, \Delta, \mathbf{b}, \xi)$ be a bounded k-counter automaton with $k \geq 3$ and $\mathbf{b} = (b_1, \ldots, b_k)$. Our first observation is that we may assume all bounds of $\mathbf{b}$ to be identical, *i.e.*, for any $\hat{b} \geq \max\{b_i : i \in [k]\}$, reachability in $\mathcal{A}$ can be reduced in logarithmic space to reachability in a bounded k-counter automaton $\mathcal{A}' = (Q', \Delta', \hat{\mathbf{b}}, \xi')$, where $\hat{\mathbf{b}} = (\hat{b}, \ldots, \hat{b})$. We can obtain $\mathcal{A}'$ from $\mathcal{A}$ by the following procedure:

- replace each transition (q, q') labeled with $add_i(z)$ with two consecutive transitions $(q, q''), (q'', q')$, where q'' is a fresh control location; and
- label (q, q'') with $add_i(z)$ and (q'', q') with $counter_i \leq b_i$.

We can now establish the main lemma of this section.

Lemma 3.2. Let $\mathcal{A}$ be a bounded k-counter automaton with $k \geq 3$. There is a bounded two-counter automaton $\mathcal{A}'$ and a function $f : C(\mathcal{A}) \to C(\mathcal{A}')$ such that for all $q(\mathbf{n}), q'(\mathbf{n}') \in C(\mathcal{A})$, $q(\mathbf{n}) \to^*_{\mathcal{A}} q'(\mathbf{n}')$ iff $f(q(\mathbf{n})) \to^*_{\mathcal{A}'} f(q'(\mathbf{n}'))$. Moreover $\mathcal{A}'$, $f(q(\mathbf{n}))$ and $f(q'(\mathbf{n}'))$ are computable from $\mathcal{A}$, $q(\mathbf{n})$ and $q'(\mathbf{n}')$ in logarithmic space.

Proof:
Without loss of generality, let $b = 2^r - 1$ be the uniform bound of $\mathcal{A}$, so that r bits are sufficient to represent a counter value. The idea behind our reduction is to simulate counters two up to k of $\mathcal{A}$ in the (most significant) bits of the first counter of $\mathcal{A}'$, and to use the second counter of $\mathcal{A}'$ as temporary storage.

The control locations of $\mathcal{A}'$ contain those of $\mathcal{A}$ as a subset, however the transitions of $\mathcal{A}$ will be replaced with gadgets in $\mathcal{A}'$. We set the bound on the counters of $\mathcal{A}'$ to be $2^{kr} - 1$. In order to formalise our intuition about the relationship between configurations of $\mathcal{A}$ and $\mathcal{A}'$, we define

$$f : C(\mathcal{A}) \to C(\mathcal{A}') \overset{\text{def}}{=} q(n_1, \ldots, n_k) \mapsto q(\sum_{i \in [k]} 2^{(i-1)r} n_i, 0).$$

Our aim is to construct $\mathcal{A}'$ such that $q(\mathbf{n}) \to^*_{\mathcal{A}} q'(\mathbf{n}')$ iff $f(q(\mathbf{n})) \to^*_{\mathcal{A}'} f(q'(\mathbf{n}'))$. To this end, the transitions of $\mathcal{A}$ are replaced by gadgets in $\mathcal{A}'$ that, informally speaking, ensure that we do not underflow or overflow. Formally, any transition (q, q') labeled with $add_i(z), z \in \mathbb{Z}$ in $\mathcal{A}$ gets replaced in $\mathcal{A}'$ with a gadget that performs the following sequence of actions on the first and second counters of $\mathcal{A}'$:

(i) move all bits with index ir up to $kr - 1$ from the first to the second counter[1];

(ii) add $2^{(i-1)r} z$ to the first counter;

(iii) test that the value of the second counter is less than 2^{ir};

(iv) move the bits with index ir up to $kr - 1$ from the second to the first counter; and

(v) switch to control location q'.

A generic gadget $\mathcal{A}_{mov}(i, j)$ that enables moving bits with index i up to j is graphically depicted in Figure 1. The idea is to non-deterministically subtract the relevant bits from the first counter while at the same time adding them to the second counter, and to finally check that all bits were transferred. Note that the test in (iii) ensures that the simulation of adding z to the i-th counter does not result in an overflow which could occur since $\mathcal{A}$ and $\mathcal{A}'$ do not have the same bound.

It is now not difficult to verify that $q(\mathbf{n}) \to_{\mathcal{A}} q'(\mathbf{n}')$ iff there is a path $\pi : f(q(\mathbf{n})) \to^*_{\mathcal{A}'} f(q'(\mathbf{n}'))$ in $T(\mathcal{A}')$, which concludes the proof of the lemma. $\square$

3.2. Reduction (ii)

We now show that reachability in bounded two-counter automata can be reduced to reachability in three-clock timed automata with clocks x, y, z. By the observation made in Section 3.1, we may assume that $\mathcal{A}$ has a uniform bound b. We encode counter values as follows: for any clock valuation ϑ, whenever

[1]In this paper we start indexing from zero.

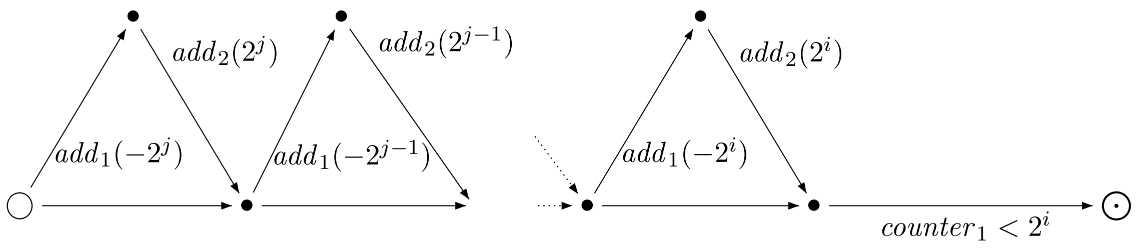


Figure 1. Generic gadget $\mathcal{A}_{mov}(i, j)$ used for moving the bits with index i up to j from the first to the second counter.

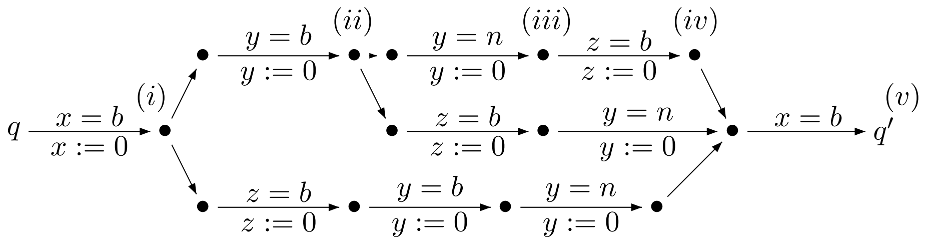


Figure 2. Gadget for simulating an increment of the first counter by $n \in \mathbb{N}$.

$\vartheta(x) = b$ the value of the first counter of $\mathcal{A}$ is encoded in $\vartheta(x) - \vartheta(y)$ and $\vartheta(x) - \vartheta(z)$ encodes the second counter of $\mathcal{A}$. A similar encoding has also been used in [25] in order to show undecidability of reachability in parametric three-clock timed automata.

Lemma 3.3. Let $\mathcal{A}$ be a bounded two-counter automaton and $q(\mathbf{n}), q'(\mathbf{n}') \in C(\mathcal{A})$. Then there is a three-clock timed automaton $\mathcal{A}'$ and a function $f : C(\mathcal{A}) \to C(\mathcal{A}')$ such that $q(\mathbf{n}) \to^*_{\mathcal{A}} q'(\mathbf{n}')$ iff $f(q(\mathbf{n})) \to^*_{\mathcal{A}'} f(q'(\mathbf{n}'))$. Moreover $\mathcal{A}'$, $f(q(\mathbf{n}))$, and $f(q'(\mathbf{n}'))$ are computable from $\mathcal{A}$, $q(\mathbf{n})$, and $q'(\mathbf{n}')$ in logarithmic space.

Proof:
Let b be the uniform bound of $\mathcal{A}$. The function f required in the lemma is defined as follows:

$$f : q(n_1, n_2) \mapsto q(\{x \mapsto b, y \mapsto (b - n_1), z \mapsto (b - n_2)\}),$$

which is clearly computable in logarithmic space.

We now sketch how $\mathcal{A}'$ can be obtained from $\mathcal{A}$. The timed automaton $\mathcal{A}'$ contains all control locations of $\mathcal{A}$ as a subset. However, the transitions from $\mathcal{A}$ are replaced by gadgets that manipulate the clocks in a way that simulates the action of the replaced transitions. As an invariant, we ensure that at any time $\mathcal{A}'$ reaches a control location that exists in $\mathcal{A}$, the value of the clock x is b. Suppose $(q, q') \in \Delta$ is a transition from $\mathcal{A}$ such that $\xi(q, q') = add_1(n)$ for some $n \in \mathbb{N}$. In $\mathcal{A}'$, we replace this transition by the gadget shown in Figure 2. There, clock constraints are written as *e.g.* $x = b$ and clock resets as *e.g.* $x := 0$.

Consider a configuration $q(n_1, n_2)$ of $\mathcal{A}$ which corresponds to the configuration $q(\{x \mapsto b, y \mapsto b - n_1, z \mapsto b - n_2\})$ of $\mathcal{A}'$. Since we want to simulate that the first counter of $\mathcal{A}$ increases, we need to increase the difference between the value of the clock x and the value of the clock y by n. To this end, the gadget first resets the clock x. It then non-deterministically guesses the order of the simulated counter values: it branches upwards if the value of the first counter is no greater than that of the second counter, *i.e.*, $n_1 \leq n_2$, and downwards otherwise. We only discuss the first case here. The gadget waits until clock y has value b. Then we aim at waiting for n time units in order to increase the difference of x and y by n. However, clock z could reach value b in the meantime, which occurs when $n_2 \leq n_1 + n$. Thus, again, a non-deterministic choice is performed to handle the two cases. If z reaches b before y reaches n, the downward branch can be taken, which first resets z as it reaches clock value b and then y when it reaches clock value n. The converse case can be shown analogously, see below. Finally, the gadget waits until clock x reaches clock value b in order to establish our agreed invariant when it reaches q'. Note that if the increment would result in a counter value larger than b, the automaton $\mathcal{A}'$ would block, as expected. It is easily checked that an analogous gadget can be constructed for the simulation of incrementing the second counter.

We demonstrate the correctness of our construction by determining the intermediate values of the clocks along the path labelled by (i)–(v), which, as discussed above, is traversed when n_1 and n_2 are such that $n_2 > n_1 + n$:

(i) $\vartheta(x) = 0$, $\vartheta(y) = b - n_1$ and $\vartheta(z) = b - n_2$ (by the invariant);

(ii) $\vartheta(x) = n_1$, $\vartheta(y) = 0$ and $\vartheta(z) = b - (n_2 - n_1) < b$, since $n_2 > n_1$;

(iii) $\vartheta(x) = n_1 + n$, $\vartheta(y) = 0$ and $\vartheta(z) = b - (n_2 - n_1 - n) < b$, since $n_2 > n_1 + n$;

(iv) $\vartheta(x) = n_1 + n + n_2 - n_1 - n = n_2$, $\vartheta(y) = n_2 - n_1 - n$ and $\vartheta(z) = 0$;

(v) $\vartheta(x) = b$, $\vartheta(y) = b - (n_1 - n)$ and $\vartheta(z) = b - n_2$.

Finally, the same approach can be used in order to simulate decrementing a counter. The main difference is that if we, say, wish to simulate decrementing the first counter by n, instead of waiting for the clock y to reach b and then n, as it is done in Figure 2, we wait instead for the clock y to reach $b - n$. □

3.3. Reduction (iii)

It remains to reduce reachability in k-clock timed automata to reachability in bounded $(2k + 2)$-counter automata. Let $\mathcal{A} = (Q, X, \Delta, \xi)$ be a timed automaton with clocks $X = \{x_1, \ldots, x_k\}$. Recall that a configuration of a timed automaton is a tuple consisting of a control state and a clock valuation. In order to abstract away from the *a priori* infinite state space, we employ the region abstraction as a reachability-preserving equivalence relation on the set of configurations of a timed automaton. Recall that for a clock $x \in X$, C_x denotes the maximum value of the constants occurring in the guards of $\mathcal{A}$ involving x. As defined in [2], the region abstraction relates two configurations $q(\vartheta) \sim q'(\vartheta')$ whenever

(a) their control locations are the same, *i.e.*, $q = q'$;

(b) the integral parts of the value of each clock with a value below the maximum constant appearing in $\mathcal{A}$ are the same, *i.e.*, for any $x \in X$, $\lfloor \vartheta(x) \rfloor = \lfloor \vartheta'(x) \rfloor$, or both $\vartheta(x)$ and $\vartheta'(x)$ are greater than C_x;

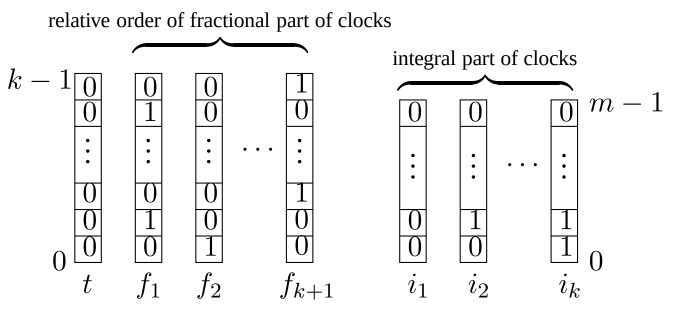


Figure 3. Illustration of the approach to simulating the region abstraction on bounded counters.

(c) the relative order of the fractional parts of the values of all relevant clocks are the same, *i.e.*, for any two different $x, y \in X$ such that $\vartheta(x) \leq C_x$ and $\vartheta(y) \leq C_y$, $\vartheta(x) - \lfloor\vartheta(x)\rfloor \leq \vartheta(y) - \lfloor\vartheta(y)\rfloor$ iff $\vartheta'(x) - \lfloor\vartheta'(x)\rfloor \leq \vartheta'(y) - \lfloor\vartheta'(y)\rfloor$; and

(d) the clocks with fractional part 0 are the same, *i.e.*, for all $x \in X$, $\vartheta(x) - \lfloor\vartheta(x)\rfloor = 0$ iff $\vartheta'(x) - \lfloor\vartheta'(x)\rfloor = 0$.

Given a k-clock timed automaton $\mathcal{A}$, we sketch how to construct a bounded $(2k+2)$-counter automaton $\mathcal{A}'$ such that any reachability problem for $\mathcal{A}$ translates into an instance of a reachability problem in $\mathcal{A}'$. The idea is to encode each $\sim$-equivalence class of a configuration of a timed automaton as a single configuration of $\mathcal{A}'$. The main difficulty is that conditions (b) – (d) allow for an exponential number of possibilities in $|\mathcal{A}|$, and in order to achieve a logarithmic-space reduction, the conditions (b) – (d) thus cannot directly be hard-wired into the control locations of $\mathcal{A}'$, but will instead be encoded into the $2k+2$ counters.

Lemma 3.4. Let $\mathcal{A}$ be a k-clock timed automaton and $q(\vartheta), q'(\vartheta') \in C(\mathcal{A})$. Then there is a bounded $(2k+2)$-counter automaton $\mathcal{A}'$ and a function $f : C(\mathcal{A}) \to C(\mathcal{A}')$ such that $q(\vartheta) \to^*_{\mathcal{A}} q'(\vartheta')$ iff $f(q(\vartheta)) \to^*_{\mathcal{A}'} f(q'(\vartheta'))$. Moreover $\mathcal{A}'$, $f(q(\vartheta))$, and $f(q'(\vartheta'))$ are computable from $\mathcal{A}$, $q(\vartheta)$, and $q'(\vartheta')$ in logarithmic space.

Proof:
Let $m \in \mathbb{N}$ be chosen such that m bits are sufficient to represent one plus the maximum integer constant appearing in $\mathcal{A}$. The bounded counter automaton $\mathcal{A}'$ has bounded counters $f_1, \ldots, f_{k+1}$, $i_1, \ldots, i_k$ and t, where the maximum value for the counters $f_1, \ldots, f_{k+1}$ and t is $2^k - 1$ and $2^m - 1$ for the counters $i_1, \ldots, i_k$. The bit representation of the counters is illustrated in Figure 3, where the least significant bit of each counter is at the bottom and the most significant bit on top.

The counter t serves as temporary storage space. In order to represent a configuration $q(\vartheta)$ of $\mathcal{A}$, $f_1, \ldots, f_{k+1}$ are used as slots that encode the relative order of the clocks with respect to their fractional parts induced by ϑ. The counter f_1 additionally indicates those clocks that have fractional part 0. Since there are k clocks, $k+1$ different slots are sufficient. The encoding is such that a clock j is in slot l if the j-th bit of the counter f_l is set, and for the encoding to be faithful, consequently the j-th bit must not be set for any other counter $f_{l'}$ for $l' \neq l$. For $l < l' \in [k]$, whenever clock j is in slot l and clock j' in l', *i.e.*, the j-th bit of the counter f_l and the j'-th bit of the counter $f_{l'}$ are set, this indicates that clock

j has a value whose fractional part is strictly smaller than the fractional part of the value of clock j'. If the j-th and the j'-th bit of a counter f_l are both set, this indicates that clocks x_j and $x_{j'}$ have the same fractional part. Finally, the counters $i_1, \ldots, i_k$ are used to store the integral parts of the clocks induced by ϑ in binary, *i.e.*, the counter i_1 encodes the integral part of the first clock, the counter i_2 the integral part of the second clock, *etc.*.

As an example, consider a clock valuation ϑ with $\vartheta(x_1) = 4.1$, $\vartheta(x_2) = 2.0$, $\vartheta(x_3) = 0.8$, $\vartheta(x_{k-1}) = 0.0$ and $\vartheta(x_k) = 3.8$ whose encoding is illustrated in Figure 3. Both clocks x_2 and x_{k-1} have fractional parts 0, hence the second and the $(k-1)$-th bit of counter f_1 are set. The fractional part of clock x_1 is greater than the fractional parts of x_2 and x_{k-1}, hence clock x_1 "resides" in the encoding in a slot to the right of the slot of x_2 and x_{k-1}, *i.e.*, in this example in counter f_2 whose first bit is set. Finally, the value of counter i_2 is 2 which corresponds to the integral part of clock x_2, the value of i_k is 3 which corresponds to the integral part of clock x_k, *etc.*.

Let us now describe how to simulate $\mathcal{A}$ and let us first consider delay transitions. The effect of a delay transition is that as time increases, clocks with the highest fractional part increase their integral part by one and have their fractional part set to zero. All other clocks do not change their integral parts and the relative order of their fractional parts, but are now in the relative order of their fractional parts to the right of those clocks that changed their integral part. Hence, delay transitions can be simulated by a gadget as follows: first, the value of the counter f_{k+1} is moved to the temporary counter t and the value of f_{k+1} is set to zero. Then, we rotate the values of the counters f_1 up to f_k by one, *i.e.*, move the value of f_1 to f_2, the value of f_2 to f_3 until eventually we move the value of the counter f_k to f_{k+1}. All clocks x_j that previously "resided" in f_{k+1} must now have a fractional part equal to zero and their integral part needs to be incremented by one. Setting the fractional part equal to zero corresponds to moving the value that was stored on the temporary counter t to f_1. Incrementing the integral part of x_j corresponds to incrementing the value of the counter i_j by one, provided that it has not yet reached its maximum value. If the maximum value has already been reached, no action is performed. In order to simulate $\mathcal{A}$, any control location of $\mathcal{A}$ is present in $\mathcal{A}'$ and has a loop which simulates an elapse of time as described above.

We now describe how to simulate discrete transitions of $\mathcal{A}$. To this end, checking the truth value of a guard of a transition against the currently abstracted clock valuation and resetting of clocks need to be simulated. We illustrate the reduction with the help of an example. Suppose the guard is $(x_1 < 6 \wedge x_2 = 4, \{x_1\})$. The constraint $x_1 < 6$ can be checked in $\mathcal{A}'$ with an edge that is labeled with $counter_{i_1} < 6$, checking $x_2 = 4$ can also be simulated with an edge $counter_{i_2} = 4$, but we additionally need to check that clock x_2 has fractional part zero, *i.e.*, is in the first slot, meaning that the second bit of f_1 is set. Simulating a reset of x_1 is also relatively straightforward: we non-deterministically choose the fractional class j of x_1, *i.e.*, the counter f_j whose first bit is set. We then set this bit to zero, *i.e.*, remove 2^0 from f_j, add 2^0 to the counter f_1 and set i_1 to zero. The latter can be implemented with the help of a loop that subtracts 1 from i_1 until a zero-test on i_1 is successful.

It remains to briefly discuss some further technical details left out so far. The task of moving contents between counters of $\mathcal{A}'$ can easily be realised by a slight adaptation of the gadget presented in Figure 1. Testing whether a particular bit of a counter, say the j-th bit of f_l, is set can also be realised in similar fashion: we first copy the value of counter f_l to counter t. Next, we run through a gadget which first subtracts 2^j from t and then non-deterministically subtracts all other powers of two. If a subsequent zero test is successful, the j-th bit of f_l had been set, otherwise we get stuck at some point.

In summary, in order to check $q(\vartheta) \to^*_{\mathcal{A}} q'(\vartheta')$, we construct $\mathcal{A}'$ in logarithmic space, compute counter values $\mathbf{n}, \mathbf{n}' \in \mathbb{N}^{2k+2}$ that represent the abstraction of the clock valuations ϑ, ϑ' and check $q(\mathbf{n}) \to^*_{\mathcal{A}'} q'(\mathbf{n}')$. The converse direction follows straightforwardly by defining a bijection between configurations $q(\mathbf{n})$ and the region abstraction of $\mathcal{A}$; we omit further details. □

4. The case of two clocks and one bounded counter

We now consider the special case of two-clock timed automata and show that reachability for this class of timed automata is logarithmic-space inter-reducible with reachability in bounded *one*-counter automata.

Our first observation is that the direction from bounded one-counter automata to two-clock timed automata can be obtained as a trivial adaptation of the construction given in Lemma 3.3, from which we obtain the following lemma.

Lemma 4.1. Let $\mathcal{A}$ be a bounded one-counter automaton and $q(n), q'(n') \in C(\mathcal{A})$. There exists a two-clock timed automaton $\mathcal{A}'$ and a function $f : C(\mathcal{A}) \to C(\mathcal{A}')$ such that $q(n) \to^*_{\mathcal{A}} q'(n')$ iff $f(q(n)) \to^*_{\mathcal{A}'} f(q'(n'))$. Moreover $\mathcal{A}'$, $f(q(n))$, and $f(q'(n'))$ are computable from $\mathcal{A}$, $q(n)$, and $q'(n')$ in logarithmic space.

The remainder of this section is devoted to a reduction in the converse direction, which is slightly more involved. We first formally define two gadgets that will be used in this reduction. The first gadget adds a number to the counter that is non-deterministically selected from an interval whose endpoints are given in binary. This is formalised in the following lemma.

Lemma 4.2. Let $a < b \in \mathbb{N}$. There exists a logarithmic-space computable bounded one-counter automaton $\mathcal{A}$ with control locations q, q' such that for all $n, n' \in \mathbb{N}$, $q(n) \to^*_{\mathcal{A}} q'(n')$ iff $n' - n \in [a, b]$.

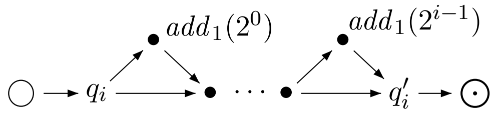
Proof:
The main idea is that any natural number can be expressed as a sum of powers of two minus one, and that we can construct a gadget which allows for adding any number between zero and a power of two minus one.

Let us first show how a natural number $b \in \mathbb{N}$ can be written as a sum of powers of two minus one. For any $m \in \mathbb{N}$, define

$$k(m) \overset{\text{def}}{=} \max\{i \in \mathbb{N} : (2^i - 1) \leq m\}.$$

We define a sequence $m_1 \geq m_2 \geq \ldots$ of values in $\mathbb{N}$ as follows: $m_1 \overset{\text{def}}{=} b$ and $m_{i+1} \overset{\text{def}}{=} m_i - (2^{k(m_i)} - 1)$ for $i > 0$. Let $(k_i)_{i>0}$ be the sequence of the $k(m_i)$, we have $b = \sum_{i>0}(2^{k_i} - 1)$. Since $m_{i+1} \leq m_i/2$ for all $i > 0$, we have $k_j > 0$ and $k_{j+1} = 0$ for some $j \leq \log b$ and hence $b = \sum_{i \in [j]}(2^{k_i} - 1)$. For example taking $b = 11$, we have $m_1 = 11$, $m_2 = 4$, $m_3 = 1$, $m_4 = 0$, and $11 = (2^3 - 1) + (2^2 - 1) + (2^1 - 1)$.

Next, a gadget $\mathcal{A}_i$ that allows for adding a value in the interval $[0, 2^i - 1]$ can be constructed straightforwardly:



Now for the construction of $\mathcal{A}$ from $a, b \in \mathbb{N}$ required in the lemma, we first consider the case $a = 0$ and proceed as follows. For the sequence of $(k_i)_{i>0}$ as defined above, we construct the above gadgets $\mathcal{A}_{k_i}$ such that $\mathcal{A}_{k_i}$ connects to $\mathcal{A}_{k_{i+1}}$ for $i \in [j-1]$. Let q be the incoming location $\bigcirc$ of $\mathcal{A}_{k_1}$ and q' the terminal location $\odot$ of $\mathcal{A}'_{k_j}$, it is easily verified that $q(n) \rightarrow^*_{\mathcal{A}} q'(n')$ iff $n' - n \in [0, b]$.

In the general case where a takes an arbitrary values from $\mathbb{N}$, we construct a one-counter automaton $\mathcal{A}$ as above that allows for representing any number in the interval $[0, b-a]$ and add a new initial location that has a transition to the initial control location of $\mathcal{A}$ that adds a to the counter. $\square$

The second gadget allows for checking that the current counter value lies in a certain interval without destroying it.

Lemma 4.3. Let $a < b \in \mathbb{N}$. There exists a logarithmic-space computable bounded one-counter automaton $\mathcal{A}$ with control locations q, q' such that for all $n \in \mathbb{N}$, $q(n) \rightarrow^*_{\mathcal{A}} q'(n')$ iff $n \in [a, b]$.

Proof:
The automaton $\mathcal{A}$ consists of two consecutive transitions, the first checks that the counter is greater or equal to a and the second that it is less or equal to b. As defined in Section 2.4.1, those test to not alter the value of the counter. $\square$

For the remainder of this section, fix a two-clock timed automaton $\mathcal{A} = (Q, X, \Delta, \xi)$ such that $X = \{x, y\}$. In the following, we describe how to construct in logarithmic space a bounded one-counter automaton $\mathcal{A}' = (Q', \Delta', b, \xi')$ that simulates $\mathcal{A}$. For technical convenience we assume that the counter of $\mathcal{A}'$ takes values from an interval in $(1/2)\mathbb{Z}$, *cf.* Section 2.4.1. The set of control locations Q' of $\mathcal{A}'$ contains as a subset the control locations of Q paired with *abstractions of clock valuations*. We first define these abstractions.

Let $C_x = \{x_1, \ldots, x_a\}$ be the ordered set of x-constants in $\mathcal{A}$, *i.e.*, $x_i < x_{i+1}$ for $i \in [a-1]$, and let $C_y = \{y_1, \ldots, y_b\}$ the ordered set of y-constants, where $x_1 = y_1 = 0$. We define the augmented sets C_x^∞ and C_y^∞ as $C_x^\infty \stackrel{\text{def}}{=} C_x \cup \{\infty\}$ respectively $C_y^\infty \stackrel{\text{def}}{=} C_y \cup \{\infty\}$, where x_{a+1} and y_{b+1} identify ∞ in C_x^∞ and C_y^∞, respectively. The set of *regions R of $\mathcal{A}$* is defined as

$$R \stackrel{\text{def}}{=} \{(x_i, y_j, x_{i+b_x}, y_{j+b_y}) : x_i \in C_x, y_j \in C_y, b_x, b_y \in \{0, 1\}\},$$

which is a subset of $C_x \times C_y \times C_x^\infty \times C_y^\infty$. Note that $|R| = \mathcal{O}(|\mathcal{A}|^2)$ and that R is computable in logarithmic space. Subsequently, we will write r to identify a region from R. With each region $r \in R$, we associate a set of clock valuations $\vartheta(r)$ in the obvious way, *i.e.*,

$$\begin{aligned}
\vartheta(x_i, y_j, x_i, y_j) &\stackrel{\text{def}}{=} \{\vartheta : \vartheta(x) = x_i, \vartheta(y) = y_j\} \\
\vartheta(x_i, y_j, x_{i+1}, y_j) &\stackrel{\text{def}}{=} \{\vartheta : x_i < \vartheta(x) < x_{i+1}, \vartheta(y) = y_j\} \\
\vartheta(x_i, y_j, x_i, y_{j+1}) &\stackrel{\text{def}}{=} \{\vartheta : \vartheta(x) = x_i, y_j < \vartheta(y) < y_{j+1}\} \\
\vartheta(x_i, y_j, x_{i+1}, y_{j+1}) &\stackrel{\text{def}}{=} \{\vartheta : x_i < \vartheta(x) < x_{i+1}, y_j < \vartheta(y) < y_{j+1}\}.
\end{aligned}$$

Hence R partitions the set of all clock valuations. Moreover, any two clock valuations in the same region r cannot be distinguished by the clock constraints of $\mathcal{A}$, *i.e.*, for any two $\vartheta, \vartheta' \in \vartheta(r)$ and any clock

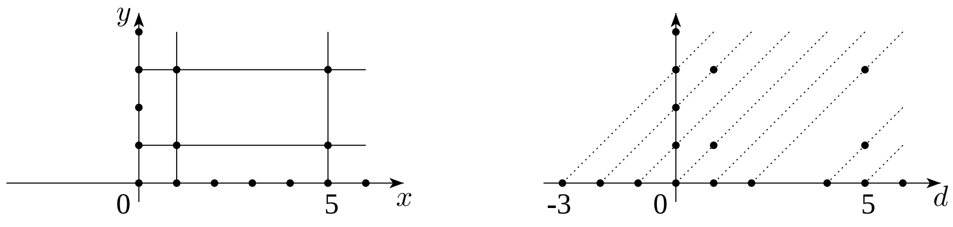


Figure 4. Illustration of the regions (left) and the clock difference zones (right) of a two-clock timed automaton with $C_x = \{0, 1, 5\}$ and $C_y = \{0, 1, 3\}$.

constraint ϕ occurring as a label of a transition of $\mathcal{A}$, we have $\vartheta \models \phi$ iff $\vartheta' \models \phi$. The left-hand side of Figure 4 depicts the regions of a two-clock timed automaton $\mathcal{A}$ with $C_x = \{0, 1, 5\}$ and $C_y = \{0, 1, 3\}$. The stroked lines in the first quadrant indicate the regions of $\mathcal{A}$, *e.g.*, $(1, 1, 5, 3)$ and $(5, 3, \infty, \infty)$ are regions of $\mathcal{A}$.

A further abstraction that we use builds upon the set of *clock differences* $D \subseteq \mathbb{Z}$ *of* $\mathcal{A}$, which is defined as $D \overset{\text{def}}{=} \{c_x - c_y : c_x \in C_x, c_y \in C_y\}$. We write D as the ordered set $D = \{d_1, \ldots, d_c\}$. Our abstraction is the set of *clock difference zones* Z *of* $\mathcal{A}$, which is a set of *symbolic intervals* on $\mathbb{Z}$ defined as

$$Z \overset{\text{def}}{=} \{[d, d] : d \in D\} \cup \{(d_i, d_{i+1}) : d_i \in D, i \in [c-1]\} \cup \{(-\infty, d_1), (d_c, \infty)\}.$$

Here, we also have $|Z| = \mathcal{O}(|\mathcal{A}|^2)$. We subsequently write z to identify a clock difference zone from Z. With each $z \in Z$, we associate a set of clock valuations

$$\vartheta(z) \overset{\text{def}}{=} \{\vartheta : \vartheta(x) - \vartheta(y) \in z\},$$

which gives us an abstraction. For instance, $[0, 0]$, $(-1, 0)$ and $(2, 4)$ are clock difference zones in the example illustrated in the right-hand side of Figure 4, where the dashed lines and the space between them indicate clock difference zones. Note that the set of clock difference zones Z partitions the set of all clock valuations as well. Informally speaking, suppose we know that a clock valuation ϑ is in some region r, then the clock difference zone adds additional information that allows for determining where the clock is located with respect to the corner points of r.

Applying the previous definitions, we now define those control locations of $\mathcal{A}'$ that we employ for simulating time delay transitions of $\mathcal{A}$. To this end, we pair each $q \in Q$ with a region and a clock difference zone:

$$Q \times \{(r, z) \in R \times Z : \vartheta(r) \cap \vartheta(z) \neq \emptyset\} \subseteq Q'.$$

The whole set Q' of control locations of $\mathcal{A}'$ will be defined subsequently and in addition contain control locations which simulate discrete transitions. Each tuple $(q, (r, z))$ represents a set $\{q(\vartheta) : \vartheta \in \vartheta(r) \cap \vartheta(z)\}$ of configurations of $\mathcal{A}$, and we can associate with every configuration $q(\vartheta)$ of $\mathcal{A}$ a control location $q(\vartheta)^\dagger$ of $\mathcal{A}'$ as

$$q(\vartheta)^\dagger \overset{\text{def}}{=} (q, (r, z)),$$

where r, z are uniquely chosen such that $\vartheta \in \vartheta(r) \cap \vartheta(z)$. Referring to the example given in Figure 4, we have $q(\{x \mapsto 3.5, y \mapsto 1.5\})^\dagger = (q, ((1,1,5,3), [2,2]))$ and $q(\{x \mapsto 3.75, y \mapsto 1.5\})^\dagger = (q, ((1,1,5,3), (2,4)))$.

Given $r \in R$ and $z \in Z$ such that $\vartheta(r) \cap \vartheta(z) \neq \emptyset$, in order to discretely simulate delay transitions of $\mathcal{A}$, we define the *successor* $succ(r, z)$ *of* r *with respect to* z. Informally speaking, elapse of time can be simulated by moving from region to region along the dashed lines in Figure 4. Let us first consider the case $z = [d, d]$ and suppose in the following that $r \in C_x \times C_y \times C_x \times C_y$:

- if $r = (x_i, y_j, x'_i, y'_j)$, and $x'_i = x_i$ or $y'_j = y_j$ then $succ(r, z) \overset{\text{def}}{=} (x_i, y_j, x_{i+1}, y_{j+1})$;
- if $r = (x_i, y_j, x_{i+1}, y_{j+1})$ and $x_{i+1} - y_{j+1} = d$ then $succ(r, z) \overset{\text{def}}{=} (x_{i+1}, y_{j+1}, x_{i+1}, y_{j+1})$;
- if $r = (x_i, y_j, x_{i+1}, y_{j+1})$ and $x_{i+1} - y_{j+1} < d$ then $succ(r, z) \overset{\text{def}}{=} (x_{i+1}, y_j, x_{i+1}, y_{j+1})$;
- if $r = (x_i, y_j, x_{i+1}, y_{j+1})$ and $x_{i+1} - y_{j+1} > d$ then $succ(r, z) \overset{\text{def}}{=} (x_i, y_{j+1}, x_{i+1}, y_{j+1})$.

Now if $z = (d_k, d_{k+1})$, we only sketch the definition of $succ(r, z)$, it can be extended in the obvious way. Again, suppose in the following that $x_{i+1} \neq \infty$ and $y_{j+1} \neq \infty$, we define

- if $r = (x_i, y_j, x_{i+1}, y_{j+1})$ and $d_{k+1} \leq x_{i+1} - y_{j+1}$ then $succ(r, z) \overset{\text{def}}{=} (x_i, y_{j+1}, x_{i+1}, y_{j+1})$;
- if $r = (x_i, y_j, x_{i+1}, y_{j+1})$ and $d_k \geq x_{i+1} - y_{j+1}$ then $succ(r, z) \overset{\text{def}}{=} (x_{i+1}, y_j, x_{i+1}, y_{j+1})$.

Regions which involve clocks whose value is ∞ can be handled analogously, *e.g.* as:

- if $r = (x, y, \infty, \infty)$ then $succ(r, z) \overset{\text{def}}{=} (\infty, \infty, \infty, \infty)$;
- if $r = (\infty, \infty, \infty, \infty)$ then $succ(r, z) \overset{\text{def}}{=} (\infty, \infty, \infty, \infty)$.

The definition of the remaining cases follows analogously. It is not difficult to check that $succ(r, z)$ can be computed in logarithmic space. As an example, referring to Figure 4 and letting $r = (1, 1, 5, 3)$ and $z = [0, 0]$, we have $succ(r, z) = (1, 3, 5, 3)$, and if $z = [2, 2]$ then $succ(r, z) = (5, 3, 5, 3)$. When $r = (1, 1, 5, 3)$ and $z = (2, 4)$, we have $succ(r, z) = (5, 1, 5, 3)$, and if $z = (-1, 0)$ then $succ(r, z) = (1, 3, 5, 3)$. Notice that the successor region in particular depends on the clock difference zone.

In order to simulate time delay steps, $\mathcal{A}'$ contains transitions from each $(q, (r, z))$ to $(q, (succ(r, z), z))$ which perform no action on the counter. Note that the clock difference zone remains unaffected by those transitions and only the region is changed. The following lemma now establishes the faithfulness of the simulation of delay transitions of $\mathcal{A}$ by $\mathcal{A}'$.

Lemma 4.4. Let $q(\vartheta) \in C(\mathcal{A})$ and $d \in \mathbb{R}_{\geq 0}$. Then for any $n \in \mathbb{N}$, $q(\vartheta) \to_{\mathcal{A}}^* q(\vartheta + d)$ iff $q(\vartheta)^\dagger(n) \to_{\mathcal{A}'}^* q(\vartheta + d)^\dagger(n)$.

Proof:
Let $q(\vartheta)^\dagger = (q, (r, z))$ and first observe that $\vartheta(x) - \vartheta(y) = (\vartheta + d)(x) - (\vartheta + d)(y)$ for all $d \in \mathbb{R}_{\geq 0}$. This in particular implies that for any $d \in \mathbb{R}_{\geq 0}$ we have $q(\vartheta + d)^\dagger = (q, (r', z))$ for some region r'; however the clock difference zone is always z.

Suppose $q(\vartheta) \to_{\mathcal{A}}^{*} q(\vartheta + d)$ then there is a path in $T(\mathcal{A})$ taking only time delay transitions. From the definition of $succ$, it is easily verified that there exist intermediate values $d_1, \ldots, d_k \in \mathbb{R}_{\geq 0}$ such that $d = \sum_{i\in[k]} r_i$ and $q(\vartheta + \sum_{i\in[k]} d_i)^{\dagger} = (q, (r_i, z))$ for regions $r_1, \ldots, r_k$ such that $succ(r_i, z) = r_{i+1}$. But then by the construction of $\mathcal{A}'$, we have:

$$q(\vartheta)^{\dagger}(n) \to_{\mathcal{A}'} (q, (r_1, z))(n) \to_{\mathcal{A}'} \cdots \to_{\mathcal{A}'} (q, (r_k, z))(n) \to_{\mathcal{A}'} q(\vartheta + d)^{\dagger}(n).$$

The converse direction follows analogously. □

Note that, informally speaking, we only simulate delay steps between regions but not inside regions. However, elapse of time inside regions only needs to be considered when resetting clocks. In order to handle clock resets, we define a further abstraction that establishes a correspondence between clock valuations and *counter values of* $\mathcal{A}'$. For our construction, we allow the counter to take values from a bounded interval in $0.5\mathbb{Z}$ and define the set of counter values as $V \overset{\text{def}}{=} \{d_1 - 0.5, d_1, d_1 + 0.5, \ldots, d_c - 0.5, d_c, d_c + 0.5\}$, where $d_1 \overset{\text{def}}{=} -\max C_y$ and $d_c \overset{\text{def}}{=} \max C_x$. We use the counter to partition the set of clock valuations. For $n \in V$, we define

$$\vartheta(n) \overset{\text{def}}{=} \begin{cases} \{\vartheta : \vartheta(x) - \vartheta(y) = n\} & \text{if } n \in V \cap \mathbb{Z} \\ \{\vartheta : n - 0.5 < \vartheta(x) - \vartheta(y) < n + 0.5 & \text{if } n \in V \setminus (\mathbb{Z} \cup \{d_1 - 0.5, d_k + 0.5\}) \\ \{\vartheta : \vartheta(x) - \vartheta(y) < d_1\} & \text{if } n = d_1 - 0.5 \\ \{\vartheta : \vartheta(x) - \vartheta(y) > d_c\} & \text{if } n = d_c + 0.5. \end{cases}$$

We use the definition of $\vartheta(n)$ to map configurations of $\mathcal{A}$ to configurations of $\mathcal{A}'$. For any clock valuation ϑ, let $\vartheta^{\ddagger}$ denote the unique $n \in V$ such that $\vartheta \in \vartheta(n)$. We define

$$q(\vartheta)^{\ddagger} \overset{\text{def}}{=} q(\vartheta)^{\dagger}(\vartheta^{\ddagger}).$$

Referring to the example in Figure 4, we have $V = \{-3.5, 3, -2.5, \ldots, 4.5, 5, 5.5\}$, and, for instance,

$$q(\{x \mapsto 3.5, y \mapsto 1.5\})^{\ddagger} = (q, ((1, 1, 5, 3), [2, 2]))(2)$$
$$q(x \mapsto 3.75, y \mapsto 1, 5\})^{\ddagger} = (q, ((1, 1, 5, 3), (2, 4)))(2.5).$$

The partitioning of the clock valuations through the counter value is less coarse than through clock difference zones. It classifies clock valuations according to whether the difference between the clocks is a fixed integer, lies strictly in a unit interval between two consecutive fixed integers, or lies outside the “relevant” integers. That, however, leads to a number of partitions which is exponential in the size of $\mathcal{A}$ due to binary encoding of numbers, which is the reason why we store this abstraction of clock valuations in the counter value and do not encode it into the control states as we did for the other abstractions discussed above. While simulating $\mathcal{A}$ through $\mathcal{A}'$, via the gadget defined in Lemma 4.3 we can always ensure that if we are in a configuration $(q, (r, z))(n)$ of $\mathcal{A}'$ then n is consistent with z, *i.e.*, $n \in z$ and *a fortiori* $n \in r$. In particular, this gadget allows for non-deterministically choosing the correct clock difference zone with respect to the current counter value.

The key point of this additional abstraction of the difference of the two clocks into the counter together with the abstraction of regions and clock difference zones provides sufficient information in order to faithfully simulate clock resets. In the general case, this is most relevant to regions of the form $r = (x_i, y_i, x_{i+1}, y_{i+1})$. Depending on the clock difference zone z, knowing that $\vartheta \in \vartheta(r) \cap \vartheta(z)$ and the difference $\vartheta(x) - \vartheta(y)$ allows for deriving bounds on $\vartheta(x)$ and $\vartheta(y)$, as shown by the following lemma.

Lemma 4.5. Let ϑ be such that $\vartheta(x) - \vartheta(y) = d$. Then the following hold:

(i) if $x_1 < \vartheta(x) < x_2$ then $d - x_2 < -\vartheta(y) < d - x_1$;

(ii) if $y_1 < \vartheta(y) < y_2$ then $d + y_1 < x < d + y_2$;

(iii) if $\vartheta(x) < x_2$ and $y_1 < \vartheta(y)$ then $d + y_1 < \vartheta(x) < x_2$ and $d - x_2 < -\vartheta(y) < -y_1$; and

(iv) if $x_1 < \vartheta(x)$ and $\vartheta(y) < y_2$ then $x_1 < \vartheta(x) < d + y_2$ and $-y_2 < -\vartheta(y) < d - x_1$.

Proof:
Immediate. □

Each case in Lemma 4.5 is induced by the boundaries of the intersection of r with a possible a clock difference zone. The benefit of the lemma is that in order to, for instance, faithfully simulate a reset of clock x in Case (i), we only have to subtract some value from the counter (which stores an abstraction of the difference between the clocks) that lies in the interval $[x_1, x_2]$, which can be achieved by an appropriate adaptation of the gadget constructed in Lemma 4.2.

We are now ready to describe the technical particularities of how to simulate discrete transitions and clock resets. Throughout the remainder of this section, whenever we consider a configuration $(q, (r, z))(n)$ of $\mathcal{A}'$ that corresponds to some configuration $q(\vartheta)$ of $\mathcal{A}$, it is helpful to think of ϑ to lie, if possible, at or, otherwise, infinitesimally close to the bottom left corner of $\vartheta(r) \cap \vartheta(n)$. In addition to the control locations mentioned above, q' contains control locations that we use to initiate the simulation of clock resets:

$$Q \times \{(r, z) \in R \times Z : \vartheta(r) \cap \vartheta(z) \neq \emptyset\} \times \{reset_x, reset_y, reset_{x,y}\} \subseteq Q'.$$

If $(q, q') \in \Delta$, $\xi(q, q') = (\phi, Y)$ and $\vartheta \models \xi(q, q')$ for all $\vartheta \in \vartheta(r) \cap \vartheta(z)$ then, depending on which clocks are required to be reset by Y, Δ' contains a transition from $(q, (r, z))$ to $(q', (r, z), reset_x)$, $(q', (r, z), reset_y)$ or $(q', (r, z), reset_{x,y})$, which perform no action on the counter. If no clock is required to be reset, *i.e.*, $Y = \emptyset$, then $(q, (r, z))$ directly connects to $(q', (r, z))$. Note that checking whether $\vartheta \models \phi$ for all $\vartheta \in \vartheta(r) \cap \vartheta(z)$ can be performed in logarithmic space, since $\vartheta \models \phi$ for all $\vartheta \in \vartheta(r) \cap \vartheta(z)$ iff $\vartheta \models \phi$ for any $\vartheta \in \vartheta(r) \cap \vartheta(z)$.

As discussed above, the way we deal with simulating clock resets through $\mathcal{A}'$ requires a change of the counter value $\mathcal{A}'$. The simplest case is the simulation a reset of both clocks x, y. This can easily be realised by a gadget which sets the counter to 0, changes r to $(0, 0, 0, 0)$ and z to $[0, 0]$. Thus we are left with the case of resetting *one* clock where things become slightly more complicated, in particular when we simulate a reset on a clock valuation ϑ such that $\vartheta \in \vartheta(r)$ for $r = (x_i, y_j, x_{i+1}, y_{j+1})$. As described above, so far we have only abstracted from delay transitions which change regions, but now we are confronted with also taking delays into account which happen inside regions. Informally speaking, when resetting a clock, those delays determine where we land on the x- respectively y-axis, *cf.* Figure 4.

In the following, we consider two representative cases that show how to simulate resetting a single clock of $\mathcal{A}$ in $\mathcal{A}'$, the other cases can be derived analogously.

(i) Case: $r = (x_i, y_j, x_{i+1}, y_{j+1})$, $z = [d, d]$, $d + y_{j+1} < x_{i+1}$ and we wish to reset clock y of a clock valuation $\vartheta \in \vartheta(r) \cap \vartheta(z)$. Consequently, the value of the counter is d. When region zone r the value of clock y can take any value in the interval (y_i, y_{j+1}), which corresponds to Case (ii) in Lemma 4.5. Consequently, after resetting y the value of clock x lies in the interval $(d+y_1, d+y_2)$. Such a counter value can be achieved as follows:

- connect $(q, (r, z), \mathit{reset}_y)$ to a gadget that non-deterministically adds some value from the interval $[y_j + 0.5, y_{j+1} - 0.5]$ to the counter, as defined in Lemma 4.2;
- then non-deterministically guess $z' \in Z$ and verify with the gadget defined in Lemma 4.3 that z' is consistent with the new counter value before switching to the control location $(q', (x_i, 0, x_{i+1}, 0), z')$.

Let us illustrate this case with the help of Figure 4, for example with $r = (1, 1, 5, 3)$ and $z = [1, 1]$. In this example, if we consider a clock valuation ϑ infinitesimally close to $(2, 1)$, if we let time elapse while staying inside r and then reset clock y, we obtain a new clock valuation ϑ' such that $\vartheta'(x) \in (2, 4)$ and hence $q(\vartheta')^{\ddagger} = (q, (r', z'))(n')$, where $r' = (1, 0, 5, 0)$, $z' \in \{(2, 3), [3, 3], (3, 4)\}$ and $n' \in \{2.5, 3, 3.5\}$ such that z' and n' are consistent. In particular, the new value of the counter is obtained by non-deterministically adding a value from the interval between the y-boundaries 1 and 3 of r to the counter.

In order to reset clock x, we observe that for a faithful simulation the new counter value has to lie in the interval $[-y_{j+1} + 0.5, -y_j - 0.5]$, which can easily be achieved by a gadget that non-deterministically guesses a counter value in that interval and then proceeds as in the case of resetting y.

(ii) Case: $r = (x_i, y_j, x_{i+1}, y_{j+1})$, $z = (d_k, d_{k+1})$ and the boundaries of the intersection of $\vartheta(z)$ and $\vartheta(r)$ lie at (x_i, y_j, x_i, y_{j+1}) and $(x_i, y_{j+1}, x_{i+1}, y_{j+1})$, and suppose that we wish to reset the clock y. When entering zone r, in this case when time elapses we always know that $x_i < \vartheta(x)$ and $\vartheta(y) < y_2$, which corresponds to Case (iv) in Lemma 4.5. By application of the lemma, resetting clock y formally boils down to the following procedure:

- connect $(q, (r, z), \mathit{reset}_y)$ to a gadget that adds $y_{j+1} - 0.5$ to the counter;
- then non-deterministically subtract 0.5 from the counter and check that the newly guessed counter value n' is strictly above x_i;
- and finally non-deterministically guess $z' \in Z$ and verify with the gadget defined in Lemma 4.3 that z' is consistent with the new configuration $(q', (x_i, 0, x_{i+1}, 0), z')(n')$.

In Figure 4, this case can be illustrated with $r = (1, 1, 5, 3)$ and $z = (-1, 0)$.

In order to simulate resetting clock x, we proceed analogously according to Lemma 4.5 and subtract x_i from the counter, non-deterministically subtract 0.5 and verify that the counter is strictly above $-y_{j+1}$.

All remaining cases have a symmetric counterpart that we discussed before, and it is not difficult to check that all constructions can be performed in logarithmic space. Dealing with resets in regions of the form (x_i, y_i, x_i, y_i), (x_i, y_i, x_{i+1}, y_i) and (x_i, y_i, x_i, y_{i+1}) can be simulated in the obvious way, since no elapse of time inside those regions can occur.

In order to reduce an arbitrary instance $q(\vartheta) \to_{\mathcal{A}}^{*} q'(\vartheta')$ of a reachability problem in a two-clock timed automaton $\mathcal{A}$ to a reachability problem in a bounded one-counter automaton, we construct $\mathcal{A}'$ as described above, but use the sets $C_x \cup \{\vartheta(x), \vartheta'(x)\}$ and $C_y \cup \{\vartheta(y), \vartheta'(y)\}$ in order to construct the regions and clock difference zones of $\mathcal{A}'$. Summing up, in this section we have demonstrated how to construct in logarithmic space from $\mathcal{A}$, $q(\vartheta)$ and $q'(\vartheta')$ a bounded one-counter automaton $\mathcal{A}'$ and compute in logarithmic space configurations $q(\vartheta)^{\ddagger}, q'(\vartheta')^{\ddagger} \in C(\mathcal{A}')$ such that $q(\vartheta) \to_{\mathcal{A}}^{*} q'(\vartheta')$ iff $q(\vartheta)^{\ddagger} \to_{\mathcal{A}'}^{*} q'(\vartheta')^{\ddagger}$. In summary, we have thus shown the following theorem.

Theorem 4.6. Reachability in two-clock timed automata is logarithmic-space inter-reducible with reachability in bounded one-counter automata.

5. An open problem

Here, we wish to discuss a particular subclass of bounded one-counter automata for which the precise computational complexity of reachability remains an open problem. This class is called *one-dimensional bounded vector addition systems (1-dim bounded VAS)*, which are essentially bounded one-counter automata consisting of a single state with a finite number of self loops. Formally, a 1-dimensional bounded VAS is a tuple $\mathcal{A} = (b, \Delta)$ with $b \in \mathbb{N}_{>0}$ being a bound and $\Delta \subseteq \{z \in \mathbb{Z} : |z| \in [0, b]\}$ being a finite set of transitions. As expected, their size is defined as $|\mathcal{A}| = |\Delta| size(b)$, and the induced finite transition system is $T(\mathcal{A}) = ([0, b], \to_{\mathcal{A}})$ such that $n \to_{\mathcal{A}} n'$ iff $n' = n + z$ for some $z \in \Delta$ and all $n, n' \in [0, b]$. Despite their simplicity, the shortest run between two given configurations can have length exponential in the size of $\mathcal{A}$. For instance, for even $b \geq 4$ consider the family of 1-dim bounded VAS $\mathcal{A}_b$ with

- bound b; and
- transitions $\{b/2 + 1, -b/2\}$,

which have the property that any run starting in 0 and ending in $b/2$ visits all counter values from 0 up to b, *i.e.*, has length exponential in $|\mathcal{A}_b|$.

Of course, the PSPACE upper bound for reachability trivially carries over to 1-dim bounded VAS. Moreover, by a reduction from a variant of the subset sum problem, it is not difficult to show that reachability is NP-hard despite the lack of a control structure in 1-dim bounded VAS, *cf.* [1, Prop. 4.1.2]. However, the PSPACE lower bound for reachability in bounded one-counter automata from [5] does not obviously carry over to the setting of 1-dim bounded VAS. Moreover, when restricting to only two transitions, it is shown in [1, Lemma 4.3.2] that reachability can be decided in NP via the computation of the discrete volume of a certain polytope that can be associated with a reachability instance of a 1-dim bounded VAS. We thus have the following open problem:

Is the reachability problem for 1-dim bounded VAS NP*-complete?*

As a final remark, note that the reachability problem for 4-dim bounded VAS becomes PSPACE-complete, since the construction of Hopcroft and Pansiot [26, Lemma 2.1] can be applied in order to simulate control states with three additional bounded counters.

6. Conclusion

In this paper, we have established relationships between reachability problems in timed automata and counter automata. For reachability problems in timed automata with $k \geq 3$ clocks, we have provided a logarithmic-space reduction to reachability in bounded $(2k + 2)$-counter automata (whose reachability problem can in turn be reduced to reachability in two-counter automata). In the special case of two-clock timed automata we showed that the reachability problem can, in a more elaborate way, be reduced to reachability in bounded one-counter automata. We closed the circle by showing that reachability in bounded one- and two-counter automata reduces to reachability in two- and three-clock timed automata, respectively. Finally, we discussed reachability in 1-dim bounded VAS, for which the precise complexity remains an interesting open problem.

Acknowledgments. We would like to thank the anonymous referees for recommending various improvements in the presentation of this paper.

References

[1] Haase C. On the Complexity of Model Checking Counter Automata. University of Oxford, UK; 2012.

[2] Alur R, Dill DL. A theory of timed automata. Theoretical computer science. 1994;126(2):183–235. doi:10.1016/0304-3975(94)90010-8.

[3] Minsky ML. Computation: Finite and Infinite Machines. Prentice-Hall; 1967. ISBN:0-13-165563-9.

[4] Haase C, Ouaknine J, Worrell J. On the Relationship between Reachability Problems in Timed and Counter Automata. In: Finkel A, Leroux J, Potapov I, editors. Reachability Problems. vol. 7550 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2012. p. 54–65. doi:10.1007/978-3-642-33512-9-6.

[5] Fearnley J, Jurdziński M. Reachability in Two-Clock Timed Automata Is PSPACE-Complete. In: Fomin FV, Freivalds R, Kwiatkowska M, Peleg D, editors. Automata, Languages, and Programming. vol. 7966 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2013. p. 212–223. doi:10.1007/978-3-642-39212-2_21.

[6] Abdulla PA, Deneux J, Ouaknine J, Worrell J. Decidability and Complexity Results for Timed Automata via Channel Machines. In: Caires L, Italiano GF, Monteiro L, Palamidessi C, Yung M, editors. Automata, Languages and Programming. vol. 3580 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2005. p. 1089–1101. doi:10.1007/11523468_88.

[7] Bouyer P, Markey N, Reynier PA. Robust Analysis of Timed Automata Via Channel Machines. In: Amadio R, editor. Foundations of Software Science and Computational Structures. vol. 4962 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2008. p. 157–171. doi:10.1007/978-3-540-78499-9_12.

[8] Demri S, Gascon R. The Effects of Bounding Syntactic Resources on Presburger LTL. Journal of Logic and Computation. 2009;19(6):1541–1575. doi:10.1093/logcom/exp037.

[9] Courcoubetis C, Yannakakis M. Minimum and maximum delay problems in real-time systems. Formal Methods in System Design. 1992;1(4):385–415. doi:10.1007/BF00709157.

[10] Laroussinie F, Markey N, Schnoebelen Ph. Model Checking Timed Automata with One or Two Clocks. In: Gardner P, Yoshida N, editors. CONCUR 2004 - Concurrency Theory. vol. 3170 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2004. p. 387–401. doi:10.1007/978-3-540-28644-8_25.

[11] Naves G. Accessibilité dans les automates temporisés à deux horloges [Rapport de Master]. MPRI, Paris, France; 2006.

[12] Göller S, Lohrey M. Branching-Time Model Checking of One-Counter Processes and Timed Automata. SIAM Journal on Computing. 2013;42(3):884–923. doi:10.1137/120876435.

[13] Rosier LE, Yen HC. A multiparameter analysis of the boundedness problem for vector addition systems. Journal of Computer and System Sciences. 1986;32(1):105–135. doi:10.1016/0022-0000(86)90006-1.

[14] Haase C, Kreutzer S, Ouaknine J, Worrell J. Reachability in Succinct and Parametric One-Counter Automata. In: Bravetti M, Zavattaro G, editors. CONCUR 2009 - Concurrency Theory. vol. 5710 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2009. p. 369–383. doi:10.1007/978-3-642-04081-8_25.

[15] Mayr EW. An algorithm for the general Petri net reachability problem. In: ACM Symposium on Theory of Computing. New York, NY, USA: ACM; 1981. p. 238–246. doi:10.1145/800076.802477.

[16] Lambert JL. A structure to decide reachability in Petri nets. Theoretical Computer Science. 1992;99(1):79–104. doi:10.1016/0304-3975(92)90173-D.

[17] Reinhardt K. Reachability in Petri Nets with Inhibitor Arcs. Electronic Notes in Theoretical Computer Science. 2008;223(0):239–264. doi:10.1016/j.entcs.2008.12.042.

[18] Leroux J. Vector Addition Systems Reachability Problem (A Simpler Solution). In: Voronkov A, editor. Turing-100. The Alan Turing Centenary. vol. 10 of EPiC Series in Computer Science. EasyChair; 2012. p. 214–228. ISSN:2040-557X. Available from: EasyChair,http://www.easychair.org.

[19] Bonnet R. The Reachability Problem for Vector Addition System with One Zero-Test. In: Murlak F, Sankowski P, editors. Mathematical Foundations of Computer Science 2011. vol. 6907 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2011. p. 145–157. doi:10.1007/978-3-642-22993-0_16.

[20] Ibarra OH. Reversal-Bounded Multicounter Machines and Their Decision Problems. Journal of the ACM. 1978;25(1):116–133. doi:10.1145/322047.322058.

[21] Finkel A, Sangnier A. Reversal-Bounded Counter Machines Revisited. In: Ochmański E, Tyszkiewicz J, editors. Mathematical Foundations of Computer Science 2008. vol. 5162 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2008. p. 323–334. doi:10.1007/978-3-540-85238-4_26.

[22] Leroux J, Sutre G. Flat Counter Automata Almost Everywhere! In: Peled DA, Tsay YK, editors. Automated Technology for Verification and Analysis. vol. 3707 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2005. p. 489–503. doi:10.1007/11562948_36.

[23] Bouyer P, Fahrenberg U, Larsen K, Markey N, Srba J. Infinite Runs in Weighted Timed Automata with Energy Constraints. In: Cassez F, Jard C, editors. Formal Modeling and Analysis of Timed Systems. vol. 5215 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2008. p. 33–47. doi:10.1007/978-3-540-85778-5_4.

[24] Memmi G, Roucairol G. Linear algebra in net theory. In: Brauer W, editor. Net Theory and Applications. vol. 84 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 1980. p. 213–223. ISBN:3-540-10001-6.

[25] Alur R, Henzinger TA, Vardi MY. Parametric real-time reasoning. In: ACM Symposium on Theory of Computing. New York, NY, USA: ACM; 1993. p. 592–601. ISBN:0-89791-591-7.

[26] Hopcroft J, Pansiot JJ. On the reachability problem for 5-dimensional vector addition systems. Theoretical Computer Science. 1979;8(2):135–159. doi:10.1016/0304-3975(79)90041-0.