On the Relationship between Reachability Problems in Timed and Counter Automata

Christoph Haase, Joël Ouaknine, and James Worrell

Department of Computer Science, University of Oxford, UK

Abstract. This paper establishes 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 naturally logarithmic-space interreducible 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 provides new insight into two problems whose precise computational complexity have independently been identified as open.

1 Introduction

Timed automata [1] and counter automata [9] are prominent infinite-state formalisms for modelling and reasoning about quantitative behaviour 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 naturals that can be incremented, decremented or tested for zero along a transition. Reachability asks for two given configurations of a timed automaton, respectively a counter automaton, whether there is a path connecting the two configurations in the corresponding induced transition system.

From a theoretical and practical perspective, the computational complexity of deciding reachability is of great interest. Reachability in timed automata was shown to be decidable and PSpace-complete in the seminal paper [1]. This result was later refined in [4], where PSpace-hardness was established for timed automata with three clocks. The cases with fewer than three clocks were considered in [7], where reachability for one-clock timed automata was shown to be NL-complete, and NP-hard in the presence of two clocks. However, no matching upper bound for the latter problem was given in [7], and this gap is still open. Aspects of this problem have been studied in [10] without leading to an improvement of the PSpace upper bound. For counter automata, the earliest result is that reachability is undecidable in the presence of at least two counters [9]. For that reason, restrictions on the resources of counter automata that lead to decidable reachability problems have been widely studied in the literature. Examples include the restriction to one counter, disallowing zero-tests, reversal-boundedness or flatness, all of which lead to a decidable reachability problem. In this paper, we introduce bounded counter automata in which the counters range over values from bounded intervals. Due to the finite state space, reachability is trivially decidable and in PSPACE. Bounded counter automata can be viewed as a class of strongly-bounded vector addition systems with states (VASS) [8]. A main difference from general VASS is that they allow for testing whether a counter is smaller than a given constant. The complexity of reachability for bounded counter automata with only one counter was investigated in [3] in the context of weighted timed automata, where the problem was shown to be NP-hard and in PSPACE.

Our contribution. We exhibit a novel natural connection between reachability problems in timed automata and bounded counter automata which shows that, in terms of resources available, both classes behave very similarly with respect to the complexity of reachability. We show that reachability for timed automata with at least three clocks can be reduced in logarithmic space to reachability in bounded two-counter automata. The most interesting insight comes from showing the inter-reducibility between reachability in two-clock timed automata and bounded one-counter automata, since both problems have independently been studied in the literature [7,10,3] without observing that they are essentially equivalent with respect to the complexity of reachability.

Related work. Apart from the literature referenced above, work related to ours has been conducted by Figueira *et al.*, which relates decision problems for timed automata to register automata [5]. Though the latter class of automata is incomparable to ours, their work also shows a relationship between resources in both systems and the complexity of standard decision problems. Furthermore, in [2] a relationship between reachability in parametric two-clock timed automata and a rather non-standard class of parametric one-counter automata is shown. Further related work is our work [6] on the complexity of reachability in unbounded one-counter automata, which shows that this problem is NP-complete, though the techniques developed therein do not promise to improve the PSPACE upper bounded for reachability in bounded one-counter automata.

2 Preliminaries

In this section, we give some of the definitions that we use in the remainder of this paper.

General Notation. 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 $\{m + r : m \in M\}$. 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 $[1, i]$. Given $n \in \mathbb{N}$, we define $\lg n$ as $\min\{i \in \mathbb{N} : 2^i \geq n\}$. Throughout this paper, we assume *binary* encoding of integers, *i.e.*, the size of an integer z is $\lg |z|$.

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 $s, s' \in S$, we write $s \rightarrow s'$ whenever $(s, s') \in S$ and denote by $\rightarrow^*$ the reflexive transitive closure of

$\to$. Given $s, s' \in S$, *reachability* is to decide the existence of an s-s' path in T, *i.e.*, whether $s \to^* s'$.

Timed Automata. Let X be a finite set of *clock variables*. A *clock valuation* is a mapping $\vartheta : X \to \mathbb{R}_{\geq 0}$, and 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 $\vartheta + r \overset{\text{def}}{=} x \mapsto x + r$ 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 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 $\xi : \Delta \to CC(X) \times 2^X$ is the *transition labelling function*. 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 $Q \times CV(X)$. The *size* of a timed automaton is $|\mathcal{A}| \overset{\text{def}}{=} |Q| + |\Delta| + \max\{\lg n : n \in C_x, x \in X\}$.

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

(i) $q = q'$ and there exists $r \in \mathbb{R}_{\geq 0}$ such that $\vartheta' = \vartheta + r$ (*delay transitions*);
(ii) $(q, q') \in \Delta$, $\xi(q, q') = (\phi, X')$, $\vartheta \models \phi$ and ϑ' is such that $\vartheta'(x') = 0$ for every $x' \in X'$ and $\vartheta'(x) = \vartheta(x)$ for every $x \in X \setminus X'$ (*discrete transitions*).

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

Bounded Counter Automata. Let $k \in \mathbb{N}$ and $Op \overset{\text{def}}{=} \{add_i(z) : i \in [k], z \in \mathbb{Z}\}$. A *bounded k-counter automaton* is a tuple $\mathcal{A} = (Q, \Delta, \boldsymbol{b}, \xi)$, where Q is a finite set of *control locations*, $\Delta \subseteq Q \times Q$ is the *transition relation*, $\boldsymbol{b} = (b_1, \ldots, b_k) \in \mathbb{N}^k$ is a vector of *bounds* and $\xi : \Delta \to \mathsf{Op}$ is the *transition labelling function*, where the absolute value of each add_i is at most the maximum value of the components of $\boldsymbol{b}$. The set $C(\mathcal{A})$ of *configurations* of $\mathcal{A}$ is $Q \times [0, b_1] \times \ldots \times [0, b_k]$. We call b_i the *bound* of counter i. The *size* of a bounded k-counter automaton is $|\mathcal{A}| \overset{\text{def}}{=} |Q| + |\Delta| + \max\{\lg b_i : i \in [k]\}$.

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 $(q, n_1, \ldots, n_k) \to (q', n_1', \ldots, n_k')$ if $(q, q') \in \Delta$, $\xi(q, q') = add_i(z)$, $n_i' = n_i + z$ and $n_j' = n_j$ for all $j \neq i$. *Reachability* for bounded k-counter automata is to decide $C \to_{\mathcal{A}}^* C'$ for given configurations $C, C' \in C(\mathcal{A})$.

For technical convenience, we may assume that counters range over bounded intervals from $(1/n)\mathbb{Z}, n \in \mathbb{N}$, and that there are additional operations $counter_i \sim q, \sim \in \{<, \leq, =, \geq, >\}, q \in (1/n)\mathbb{Z}$ labelling transitions that allow for comparing a counter with a certain number. It is easy to see that reachability in this enriched formalism is logspace-reducible to reachability for bounded counter automata as defined above.

3 The General Case

In this section, we show the logspace inter-reducibility between reachability problems in timed automata with at least three clocks and bounded counter automata with at least two counters. We show that (i) reachability in bounded k-counter automata with $k > 2$ can be reduced to reachability in bounded two-counter automata. Next, we show that (ii) reachability in bounded two-counter automata can be reduced to reachability in three-clock timed automata. Finally, we show that (iii) reachability in k-clock timed automata with $k \geq 3$ can be reduced to reachability in bounded $(2k+2)$-counter, which by (i) implies that this problem is reducible to reachability in bounded two-counter automata.

Reduction (i). Let $\mathcal{A} = (Q, \Delta, \boldsymbol{b}, \xi)$ be a bounded k-counter automaton with $k > 2$ and $\boldsymbol{b} = (b_1, \ldots, b_k)$. It is easily seen that we may assume all bounds of $\boldsymbol{b}$ to be uniform, *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 $\mathcal{A}' = (Q', \Delta', \hat{\boldsymbol{b}}, \xi')$, where $\hat{\boldsymbol{b}} = (\hat{b}, \ldots, \hat{b})$.

Lemma 1. *Let $\mathcal{A}$ be a bounded k-counter automaton with $k > 2$. One can compute in logarithmic space a bounded two-counter automaton $\mathcal{A}'$ such that for all $(q, \boldsymbol{n}), (q', \boldsymbol{n'}) \in C(\mathcal{A})$ there exist logspace-computable $\boldsymbol{m}, \boldsymbol{m'} \in \mathbb{N}^2$ such that $(q, \boldsymbol{n}) \rightarrow^*_{\mathcal{A}} (q', \boldsymbol{n'})$ iff $(q, \boldsymbol{m}) \rightarrow^*_{\mathcal{A}'} (q', \boldsymbol{m'})$.*

Proof. Let $b = 2^g - 1$ be the uniform bound of $\mathcal{A}$, hence $r \overset{\text{def}}{=} g - 1$ bits are sufficient to represent a counter value. The idea behind our reduction is to simulate counters three up to k of $\mathcal{A}$ in the most significant bits of the second counter of $\mathcal{A}'$, and to use the most significant bits of the first 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 $2^{r(k-1)+1} - 1$. In order to make our intuition about the relationship between configurations of $\mathcal{A}$ and $\mathcal{A}'$ formal, we define a mapping h as $h : C(\mathcal{A}) \to C(\mathcal{A}') \overset{\text{def}}{=} (q, (n_1, \ldots, n_k)) \mapsto (q, (n_1, \sum_{i \in [2,k]} 2^{(i-2)r} n_i))$. Our aim is to construct $\mathcal{A}'$ such that $(q, \boldsymbol{n}) \rightarrow^*_{\mathcal{A}} (q', \boldsymbol{n'})$ iff $h(q, \boldsymbol{n}) \rightarrow^*_{\mathcal{A}'} h(q', \boldsymbol{n'})$. To this end, any transition (q, q') of $\mathcal{A}$ that adds a positive integer to the first counter, *i.e.*, is of the form $add_1(n), n \in [0, b]$, gets replaced in $\mathcal{A}'$ by two consecutive transitions that first add n to the first counter of $\mathcal{A}'$ and then check that the value of this counter is less than or equal to b. Any transition of $\mathcal{A}$ adding a negative number to the first counter is duplicated in $\mathcal{A}'$. Simulating the addition of integers to a counter different from the first counter requires some more effort. Informally speaking, we have to make sure that we do not underflow or overflow. Formally, any transition (q, q') labeled with $add_i(z), i \geq 2, z \in \mathbb{Z}$ in $\mathcal{A}$ gets replaced in $\mathcal{A}'$ with a gadget that performs the following sequence of actions on the first and second counter of $\mathcal{A}'$ in this order:

(i) move the bits $(i-1)r+1$ up to $(k-1)r$ from the second to the first counter;
(ii) add $2^{(i-2)r} z$ to the second counter;

(iii) test that the value of the second counter is less than $2^{(i-1)r+1}$;
(iv) move the bits $(i-1)r$ up to $(k-1)r$ from the first to the second counter;
(v) and switch to control location q'.

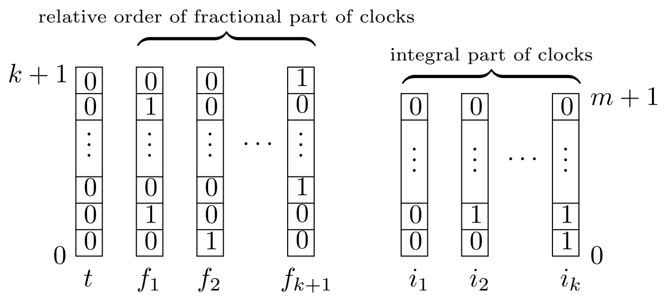
A sketch showing how to construct a gadget moving bits between counters is given in the appendix. It is not difficult to verify that $(q, \boldsymbol{n}) \to_{\mathcal{A}} (q', \boldsymbol{n'})$ iff there is a path in $T(\mathcal{A}')$ traversing locations of the gadget starting in $h(q, \boldsymbol{n})$ and ending in $h(q', \boldsymbol{n'})$, which concludes the proof of the lemma. $\square$

Reduction (ii). We now show that reachability in bounded two-counter automata can be reduced to reachability in three-clock timed automata. By the observation made in Reduction (i), we may assume that $\mathcal{A}$ has a uniform bound b. We encode counter values as follows: for any clock valuation ϑ, whenever $\vartheta(x) = b$ 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 [2] in order to show undecidability of reachability in parametric three-clock timed automata, and due to space constraints we defer the proof of the next lemma to the appendix.

Lemma 2. *Let $\mathcal{A}$ be a bounded two-counter automaton and $(q, \boldsymbol{n}), (q', \boldsymbol{n'}) \in C(\mathcal{A})$. One can compute in logarithmic space a three-clock timed automaton $\mathcal{A}'$ and clock valuations ϑ, ϑ' such that $(q, \boldsymbol{n}) \to^*_{\mathcal{A}} (q', \boldsymbol{n'})$ iff $(q, \vartheta) \to^*_{\mathcal{A}'} (q', \vartheta')$.*

Reduction (iii). The only reduction that remains to be shown is the reduction from 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 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. As defined in [1], recall that the region abstraction makes two configurations equivalent if (a) their control locations are the same; (b) the integral parts of the value of each clock with a value below the maximum constant appearing in $\mathcal{A}$ are the same; (c) the relative orders of the fractional parts of the values of the clocks are the same; and (d) the clocks with fractional part 0 are the same.

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 $\mathcal{A}'$. We aim for encoding (a)-(d) into configurations of $\mathcal{A}'$. The main difficulty is that any of (b)-(d) allows for an exponential number of possibilities in $|\mathcal{A}|$ and is therefore unsuitable to be encoded into control locations of $\mathcal{A}'$. Instead, we use the $2k+2$ counters for their encoding. Let $m \in \mathbb{N}$ be chosen such that m bits are sufficient to represent one plus the maximum integer constant appearing in $\mathcal{A}$. $\mathcal{A}'$ 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 $2^{k+1} - 1$ and $2^{m+1} - 1$ for the counters $i_1, \ldots, i_k$. The bit representation of the counters is illustrated in the figure below, where the least significant bit of each counter is at the bottom and the most significant bit on top:



The counter t will serve as temporary storage space. In order to represent a configuration (q, ϑ) of $\mathcal{A}$, $f_1, \dots, f_{k+1}$ will be used to encode the order of the clocks with respect to their fractional parts, induced by ϑ. The counter f_1 additionally encodes those clocks that have fractional part 0. Finally, the counters $i_1, \dots, i_k$ are used to store the integral part of the clocks induced by ϑ. For 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 $\vartheta(x_k) = 3.8$. Let $l < l' \in [k]$, whenever the j-th bit of the counter f_l is set and the j'-th bit of the counter $f_{l'}$ is 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'. Combining the example with the figure above, we see that the second bit of f_1 is set and the first bit of f_2 as expected. In addition, f_1 indicates which clocks have fractional part 0, which is why the second and the $(k-1)$-th bit of f_1 are set. Moreover, clock x_3 and x_k "reside" on the same counter f_{k+1} as their fractional part is equivalent in our example. The counters $i_1, \dots, i_k$ are used to store the integral part of the clocks up to $2^{m+1} - 1$. In our example, this means that the value of i_1 is 4, the value of i_2 is 2, *etc.* Delay transitions can be simulated as follows: first, the value of the counter f_{k+1} is moved to the counter t and the value of f_{k+1} is set to zero. Then, the value of the counter f_k is moved to the counter f_{k+1} until eventually we move the value of f_1 to f_2. We can then copy the value of t to f_1. All clocks that "resided" in f_{k+1} have now a fractional part zero and their integral part needs to be incremented by one. This can be simulated by incrementing the respective counter i_j, 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 elapses time as described above. It remains to describe how to discrete transitions of $\mathcal{A}$. To this end, checking the truth value of the guard of the transition against the currently abstracted clock valuation and resetting of clocks needs to be simulated. Again, we illustrate the reduction with the help of an example. Suppose the guard is $(x_1 < 6 \land x_2 = 4, \{x_1\})$. The constraint $x_1 < 6$ can be checked in $\mathcal{A}'$ with an edge that is labeled with $\mathit{counter}_{i_1} < 6$, checking $x_2 = 4$ can also be simulated with an edge $\mathit{counter}_{i_2} = 4$, but we additionally need to check that the second bit of f_1 is set. Simulating a reset of x_1 is also relatively straightforward: we nondeterministically 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.

In summary, in order to check $(q, \vartheta) \to_{\mathcal{A}}^* (q', \vartheta')$, we construct $\mathcal{A}'$ in logarithmic space, compute counter values $\boldsymbol{n}, \boldsymbol{n}' \in \mathbb{N}^{2k+2}$ that represent the abstraction of the clock valuations ϑ, ϑ' and check $(q, \boldsymbol{n}) \to_{\mathcal{A}'}^* (q', \boldsymbol{n}')$. The converse direction follows straight-forwardly by defining a bijection between configurations $(q, \boldsymbol{n})$ and the region abstraction of $\mathcal{A}$, we omit further details. We have thus proven the following lemma.

Lemma 3. *Let $\mathcal{A}$ be a k-clock timed automaton and $(q, \vartheta), (q', \vartheta') \in C(\mathcal{A})$. One can compute in logarithmic space a bounded $(2k+2)$-counter automaton $\mathcal{A}'$ and $\boldsymbol{n}, \boldsymbol{n}' \in \mathbb{N}^{2k+2}$ such that $(q, \vartheta) \to_{\mathcal{A}}^* (q', \vartheta')$ iff $(q, \boldsymbol{n}) \to_{\mathcal{A}'}^* (q', \boldsymbol{n}')$.*

The following theorem summarises the results of this section. It also yields as a byproduct that reachability in k-counter automata is PSpace-complete.

Theorem 1. *Reachability in k-clock timed automata with $k \geq 3$ is logarithmic-space inter-reducible with reachability in bounded two-counter automata.*

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 logspace inter-reducible with reachability in bounded *one*-counter automata. The reduction from reachability in bounded one-counter automata to reachability in two-clock timed automata is a rather trivial adaption of the two-counter case presented in the previous section and will be left out for brevity.

For our reduction, we require a gadget that allows for adding numbers in an interval to the counter. The proof of the next lemma is deferred to the appendix.

Lemma 4. *Let $a < b \in \mathbb{N}$. One can compute in logarithmic space a 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 $n' - n \in [a, b]$.*

Let $\mathcal{A} = (Q, X, \Delta, \xi)$ be a fixed two-clock timed automaton such that $X = \{x, y\}$. In the following, we construct in logarithmic space a bounded one-counter automaton $\mathcal{A}' = (Q', \Delta', b_l, b_u, \xi')$ corresponding to $\mathcal{A}$. For technical convenience, $\mathcal{A}'$ has a lower and an upper bound $b_l, b_u \in 0.5\mathbb{Z}$, *c.f.* Section 2. The set of control locations Q' of $\mathcal{A}'$ contains 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

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

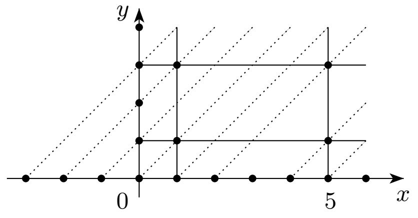


Fig. 1. Example of the regions and the clock difference zones of a two-clock timed automaton with $C_x = \{0, 1, 5\}$ and $C_y = \{0, 1, 3\}$.

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, *e.g.*, $\vartheta(x_i, y_j, x_i, y_j) \stackrel{\text{def}}{=} \{\vartheta : \vartheta(x) = x_i, \vartheta(y) = y_j\}$ and $\vartheta(x_i, y_j, x_{i+1}, y_j) \stackrel{\text{def}}{=} \{\vartheta : x_i < \vartheta(x) < x_{i+1}, \vartheta(y) = y_j\}$, *etc.*, and hence R partitions the set of all clock valuations. Moreover, any two clock valuations of a region r cannot be distinguished by $\mathcal{A}$. Figure 1 presents an example of the regions of a two-clock timed automaton $\mathcal{A}$ 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 are going to use builds upon the set of *clock differences D of $\mathcal{A}$*, which is defined as $D \stackrel{\text{def}}{=} \{c_x - c_y : c_x \in C_x, c_y \in C_y\}$. 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

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

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, we associate a set of clock valuations $\vartheta(z) \stackrel{\text{def}}{=} \{\vartheta : \vartheta(x) - \vartheta(y) \in z\}$, which gives us an abstraction. The set of clock difference zones Z also partitions the set of all clock valuations. Figure 1 illustrates the partitioning of the clock valuations by clock difference regions where each dashed line and the space between them in the first quadrant is a partition.

We can now define a subset of the control locations of $\mathcal{A}'$. Our overall goal is to represent the set of configurations of $\mathcal{A}$ as a finite quotient encoded as configurations of $\mathcal{A}'$ and then discretely simulate transitions in $T(\mathcal{A})$ as transitions in $T(\mathcal{A}')$. In order to obtain the control locations Q' of $\mathcal{A}'$, we pair each $q \in Q$ with a region and a clock difference zone and thus have $Q \times \{(r, z) \in R \times Z : \vartheta(r) \cap \vartheta(z) \neq \emptyset\} \subseteq Q'$. 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 each configuration (q, ϑ) a control location $(q, \vartheta)^\dagger$ of q' as $(q, \vartheta)^\dagger \stackrel{\text{def}}{=} (q, (r, z))$, where r, z are uniquely chosen such that $\vartheta \in \vartheta(r) \cap \vartheta(z)$.

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, elapsing of time can be simulated by moving from region to region along the dashed lines in Figure 1. Let us first consider the case $z = [d, d]$ and suppose in the following that $x_{i+1} \neq \infty$ and $y_{j+1} \neq \infty$, we *e.g.* define

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

The definition of *succ* can straightforwardly be extended for the remaining cases. Now if $z = (d_k, d_{k+1})$, we only sketch the definition of $succ(r, z)$. Again, suppose in the following that $x_{i+1} \neq \infty$ and $y_{j+1} \neq \infty$, we *e.g.* define

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

Again, the remaining cases are defined analogously and it is not difficult to check that $succ(r, z)$ can be computed in logarithmic space. In order to simulate time delay steps, $\mathcal{A}'$ contains transitions from each $(q, (r, z))$ to $(q, (succ(r, z), z))$ and to itself, which perform no action on the counter. Note that we can only simulate delay steps between regions but not within regions. Elapse of time inside regions only needs to be considered when resetting clocks and is going to be handled there. In order to handle clock resets, we are going to 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 $\mathbb{Z} \cup 0.5\mathbb{Z}$ and define the set of counter values as $V \overset{\text{def}}{=} \{d_1 - 0.5, d_1, \ldots, d_c, d_c + 0.5\}$. We use the counter to partition the set of clock valuations. For $n \in V$, we define

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

We will use this definition to map configurations of $\mathcal{A}$ to configurations of $\mathcal{A}'$. For any clock valuation ϑ, let ϑ^+ denote the unique $n \in V$ such that $\vartheta \in \vartheta(n)$. We define $(q, \vartheta)^+ \overset{\text{def}}{=} ((q, \vartheta)^\dagger, \vartheta^+)$. 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 "interesting" integers. While simulating $\mathcal{A}$ through $\mathcal{A}'$, we are going to ensure as an invariant that if we are in a configuration $((q, (r, z)), n)$ of $\mathcal{A}'$ then n is consistent with z, *i.e.*, $n \in z$. In fact, it is easy to construct a gadget that, informally speaking, non-deterministically guesses the clock difference zone the counter is currently in without destroying the counter value.

We now give some the technical details on 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,ϑ) 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 are going to use to initiate the simulation of clock resets:

$$Q \times \{(r,z) \in R \times Z : \vartheta(r) \cap \vartheta(z) \neq \emptyset\} \times \{\mathit{reset}_x, \mathit{reset}_y, \mathit{reset}_{x,y}\} \subseteq Q'.$$

If $(q,q') \in \Delta$, $\xi(q,q') = (\phi, X')$ 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 X', Δ' contains a transition from $(q,(r,z))$ to $(q',(r,z),\mathit{reset}_x)$, $(q',(r,z),\mathit{reset}_y)$ or $(q',(r,z),\mathit{reset}_{x,y})$, which perform no action on the counter. If no clock is required to be reset, *i.e.*, $\vartheta' = \emptyset$, 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.

The simplest case is when we want to simulate a reset of both clocks x, y. This can be done by setting the counter to 0, changing r to $(0,0,0,0)$ and z to $[0,0]$. If we only want to reset *one* clock, things become slightly more complicated. In the following, we are going to consider three representative cases that show how to simulate clock resets. The remaining cases follow a similar pattern.

First, suppose $r = (x_i, y_j, x_{i+1}, y_{j+1})$, $z = [d,d]$ and that we wish to reset the clock y of a clock valuation $\vartheta \in \vartheta(r)\cap\vartheta(z)$. Let us illustrate this case with the help of Figure 1, for example with $z = [0,0]$ and $r = (1,1,5,3)$. In this example, if we consider a clock valuation ϑ infinitesimally close to $(1,1)$, if we let time elapse while staying inside r and then reset clock y, we obtain a new clock valuation ϑ' such that $\vartheta'(x) \in (1,3)$ and hence $(q,\vartheta')^+ = ((q,(r',z')),n')$, where $r' = (1,0,5,0)$, $z' \in \{(1,2),[2,2],(2,3)\}$ and $n' \in [1.5, 2.5]$ such that z' and n' are consistent. Thus simulating a reset of clock y boils down to setting the counter to some value in the interval $[1.5, 2.5]$. This observation generalises to the following procedure: we pre-compute the left and right boundaries x_l, x_r on the x-axis of $\vartheta(r)\cap\vartheta(z)$, in our example 1 and 3 respectively, and connect $(q,(r,z),\mathit{reset}_y)$ to a gadget that non-deterministically repeatedly adds 0.5 to the counter, then performs a check that the counter value is strictly between x_l and x_r and finally non-deterministically performs a transition to the correct $(q,((x_i,0,x_{i+1},0),z'))$ for the new clock difference zone $z' = [d_k,d_k]$ or $z' = (d_k, d_{k+1})$ (recall that we can verify that we are in the correct clock difference zone). The case of resetting clock x can be handled analogously.

Next, we consider the case $r = (x_i, y_j, x_{i+1}, y_{j+1})$ and $z = (d_k, d_{k+1})$ where we wish to reset clock y. Again, we use Figure 1 to illustrate this case with the help of the region $r = (1,1,5,3)$. Our first observation is that this case yields four different sub-cases. First, if $d = (-1,0)$ then the boundaries of $\vartheta(r)\cap\vartheta(z)$ lie at the left and the top boundary of r. Second, if $d = (0,1)$ then the boundaries of $\vartheta(r)\cap\vartheta(z)$ lie at the bottom and the top boundary of r. Third, if $d = (2,4)$ then the boundaries of $\vartheta(r)\cap\vartheta(z)$ lie at the bottom and the right boundary of r. The fourth sub-case cannot be found in region $(1,1,5,3)$ but in region $(0,1,1,3)$, it is the case when the boundaries of $\vartheta(r)\cap\vartheta(z)$ lie at the left and the right boundary of r. Subsequently, we are going to consider the first and the second sub-case. The other sub-cases follow along similar lines.

Suppose $r = (x_i, y_j, x_{i+1}, y_{j+1})$, $z = (d_k, d_{k+1})$ and the boundaries of the intersection of $\vartheta(z)$ and $\vartheta(r)$ lie at (x_i, y_j, x_i, y_{j+1}) and $(x_i, y_{j+1}, x_{i+1}, y_{j+1})$, *e.g.*, $z = (-1, 0)$ in our example. Suppose $n \in V$ is the current counter value, since $\vartheta(y) < y_{j+1}$ for any $\vartheta \in \vartheta(r) \cap \vartheta(n)$, we have $\vartheta(x) < n + y_{i+1}$. This implies that when simulating a clock reset, the updated counter must not exceed $n + y_{i+1}$. On the other hand, the updated counter value must be above x_i. Thus, in this scenario, resetting clock y boils down to connecting $(q, (r, z), \mathit{reset}_y)$ to a gadget that adds y_{i+1} to the counter, non-deterministically subtracts 0.5 from the counter, checks whether the counter is strictly above x_i and then non-deterministically chooses the new z' that is consistent with the new counter value and switches to $(q, ((x_i, 0, x_{i+1}, 0), z'))$. If we were to reset clock x, we proceed analogously.

The last case we consider is $r = (x_i, y_j, x_{i+1}, y_{j+1})$, $z = (d_k, d_{k+1})$ and $\vartheta(z)$ intersects with $\vartheta(r)$ at (x_i, y_j, x_{i+1}, y_j) and $(x_i, y_{j+1}, x_{i+1}, y_{j+1})$, *e.g.*, $z = (0, 2)$ in our example. Let us first consider resetting clock y. Similar to the previous case, we observe that for any $n \in z$ and $\vartheta \in \vartheta(r) \cap \vartheta(n)$, $\vartheta(x) < n + y_{j+1}$. Moreover, the lower bound for $\vartheta(x)$ is determined by y_j: $\vartheta(x) > n + y_j$. Thus, simulating a clock reset on clock y boils down to adding some number from the interval $[y_j + 0.5, y_{j-1} - 0.5]$ to the counter, which can be realised with the gadget from Lemma 4. In summary, in this case a clock reset on the clock y starting a control location $(q, (r, z), \mathit{reset}_y)$ can be simulated by connecting this control location to a gadget that adds a number from $[y_j + 0.5, y_{j-1} - 0.5]$ to the counter, then non-deterministically chooses the correct new clock difference zone z' and performs a transition to $(q, ((x_i, 0, x_{i+1}, 0), z'))$. If we were to reset clock x, we observe that the value of clock y always lies in the interval (y_i, y_{i+1}). Thus, starting in $(q, (r, z), \mathit{reset}_x)$, the reset can be simulated by connecting to a gadget that non-deterministically subtracts 0.5 from the counter and then verifies that the counter is strictly between $-y_{i+1}$ and $-y_i$.

All remaining cases have a symmetric counterpart that we discussed before. It is not difficult to check that all constructions can be performed in logarithmic space. The following lemma provides a summary of the properties of the reduction we described in this section and allows us to reduce reachability in two-clock timed automata to reachability in bounded one-counter automata.

Lemma 5. *Let $\mathcal{A}$ be a two-clock timed automaton, let $\mathcal{A}'$ be its corresponding bounded one-counter automaton and let $C = ((q, (r, z)), n), C' = ((q', (r', z')), n') \in C(\mathcal{A}')$. There exist ϑ, ϑ' such that $(q, \vartheta)^+ = C$, $(q', \vartheta')^+ = C'$ and $(q, \vartheta) \to^*_{\mathcal{A}} (q', \vartheta')$ iff $C \to^*_{\mathcal{A}'} C'$.*

In order to reduce an arbitrary instance $(q, \vartheta), (q', \vartheta')$ 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 $\mathcal{A}'$. Applying the previous lemma, we obtain the main result of this section.

Theorem 2. *Reachability in two-clock timed automata logarithmic-space inter-reducible with reachability in bounded one-counter automata.*

5 Discussion

We have shown a relationship between reachability problems in timed automata and bounded counter automata with respect to the resources available. This relationship also extends to the case of one-clock timed automata, since [7] shows that reachability in this class reduces to reachability in finite-state machines, which can be viewed as bounded counter automata with no counter.

Besides these meta-level result, with regards to settling the complexity of reachability in two-clock timed automata, we believe that our reduction greatly simplifies this problem, since bounded one-counter automata are on a mathematical level cleaner and easier to define and to handle than two-clock timed automata.

References

1. R. Alur and D.L. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126:183–235, 1994.
2. R. Alur, Th.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. In *Proc. STOC*, pages 592–601. ACM Press, 1993.
3. P. Bouyer, U. Fahrenberg, K.G. Larsen, N. Markey, and J. Srba. Infinite runs in weighted timed automata with energy constraints. In *Proc. FORMATS*, volume 5215 of *LNCS*, pages 33–47. Springer, 2008.
4. C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Form. Method. Syst. Des.*, 1(4):385–415, 1992.
5. D. Figueira, P. Hofman, and S. Lasota. Relating timed and register automata. In *Proc. EXPRESS*, volume 41 of *EPTCS*, pages 61–75, 2010.
6. C. Haase, S. Kreutzer, J. Ouaknine, and J. Worrell. Reachability in succinct and parametric one-counter automata. In *Proc. CONCUR*, volume 5710 of *LNCS*, pages 369–383. Springer, 2009.
7. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking timed automata with one or two clocks. In *Proc. CONCUR*, volume 3170 of *LNCS*, pages 387–401. Springer, 2004.
8. G. Memmi and G. Roucairol. Linear algebra in net theory. In *Net Theory and Applications*, volume 84 of *LNCS*, pages 213–223. Springer, 1980.
9. M.L. Minsky. Recursive unsolvability of post's problem of "tag" and other topics in theory of turing machines. *Ann. Math.*, 74(3):437–455, 1961.
10. G. Naves. Accessibilité dans les automates temporisés à deux horloges. Rapport de Master, MPRI, Paris, France, 2006.

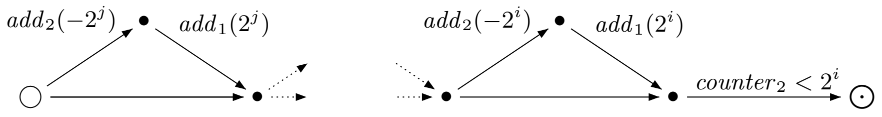


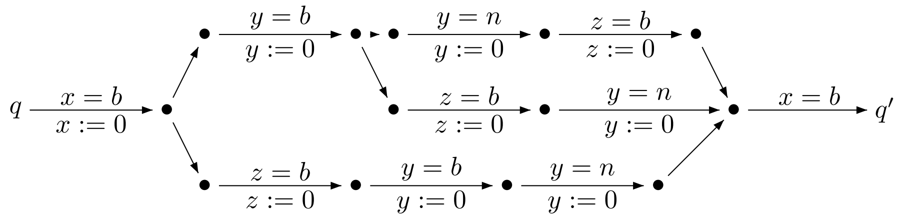
Fig. 2. Generic gadget $\mathcal{A}_{mov}(i, j)$ used for moving the bits i up to j of the second to the first counter.

A Missing Proofs from Section 3

Gadget for moving values between counters left out in the proof of Lemma 1. A gadget $\mathcal{A}_{mov}((i-1)r, (k-1)r)$ as sketched in Figure 2 can be used to move values between counters. The gadget non-deterministically subtracts relevant powers of two from the second counter and immediately adds them to the first counter. A test that the counter is less than $2^{((i-1)r)}$ at the end ensures that all bits have been moved. The same gadget can be modified to move the same bits back from the first to the second counter, as required in (iv).
Proof of Lemma 2. *Let $\mathcal{A}$ be a bounded two-counter automaton and $(q, \boldsymbol{n}), (q', \boldsymbol{n}') \in C(\mathcal{A})$. One can compute in logarithmic space a three-clock timed automaton $\mathcal{A}'$ and clock valuations ϑ, ϑ' such that $(q, \boldsymbol{n}) \to^*_{\mathcal{A}} (q', \boldsymbol{n}')$ iff $(q, \vartheta) \to^*_{\mathcal{A}'} (q', \vartheta')$.*

Proof. Let b be the uniform bound of $\mathcal{A}$. The clock valuations ϑ, ϑ' required in the lemma are defined as $\vartheta(x) = \vartheta'(x) \overset{\text{def}}{=} b, \vartheta(y) \overset{\text{def}}{=} b - n_1, \vartheta(z) \overset{\text{def}}{=} b - n_2, \vartheta'(y) \overset{\text{def}}{=} b - n'_1$ and $\vartheta'(z) \overset{\text{def}}{=} b - n'_2$.

We are now going to sketch how $\mathcal{A}'$ can be obtained from $\mathcal{A}$. The timed automaton $\mathcal{A}'$ contains all control locations of $\mathcal{A}$. However, the transitions from $\mathcal{A}$ are going to be replaced by gadgets that manipulate the clocks in a way that simulates the action of the replaced transition. As an invariant, we are going to 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 are going to replace this transition by the following gadget:



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. 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 first counter is less or equal to the second counter and downwards otherwise. We only discuss the first case here. The gadget waits until clock y has value b. It then aims at waiting for n time units. However, clock z could reach value b in the meantime. 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. Finally, the gadget waits until clock x reaches clock value b in order to establish our agreed invariant when it reaches q'. A similar gadget can be constructed for the simulation of incrementing the second counter and for decrementing the counters. $\square$

B Missing Proofs from Section 4

Proof of Lemma 4. *Let $a < b \in \mathbb{N}$. One can compute in logarithmic space a 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. We first consider the case $a = 0$ from which we are then going to derive the general case. For any $m \in \mathbb{N}$, let $k(m) \overset{\text{def}}{=} \max\{i : m \geq (2^i - 1)\}$. We define a sequence $m_1 \geq m_2 \geq \ldots$ 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 $k_{j+1} = 0$ for some $j \leq \lg b$ and hence $b = \sum_{i \in [j]}(2^{k_i} - 1)$. The one-counter automaton $\mathcal{A}$ consists of gadgets $\mathcal{A}_{k_i}, i \in [j]$ as shown here:

$$\begin{tikzcd}
 & & \bullet\, add_1(2^0) \arrow[dr] & & & & \bullet\, add_1(2^{i-1}) \arrow[dr] & & \\
\bigcirc \arrow[r] & q_i \arrow[ur] \arrow[rr] & & \bullet & \cdots & \bullet \arrow[ur] \arrow[rr] & & q'_i \arrow[r] & \odot
\end{tikzcd}$$

Each $\mathcal{A}_{k_i}$ connects to $\mathcal{A}_{k_{i+1}}$ for $i \in [j-1]$. For each $i \in \mathbb{N}$, on a run from $\bigcirc$ to $\odot$, $\mathcal{A}_i$ can non-deterministically add a number from the interval $[0, 2^i - 1]$ to the counter where we assume that $\mathcal{A}_0$ does not affect the counter at all. 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 $(q, n) \to^*_{\mathcal{A}} (q', n')$ iff $n' - n \in [0, b]$.

In the general case where a and b take 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. $\square$