Verification for Timed Automata Extended with Unbounded Discrete Data Structures

Karin Quaas[⋆]

Universität Leipzig, Germany

Abstract. We study decidability of verification problems for timed automata extended with unbounded discrete data structures. More detailed, we extend timed automata with a pushdown stack. In this way, we obtain a strong model that may for instance be used to model real-time programs with procedure calls. It is long known that the reachability problem for this model is decidable. The goal of this paper is to identify subclasses of timed pushdown automata for which the language inclusion problem and related problems are decidable.

1 Introduction

Timed automata were introduced by Alur and Dill [4], and have since then become a popular standard formalism to model real-time systems. An undeniable reason for the success of timed automata is the PSPACE decidability of the *language emptiness problem* [4]. A major drawback of timed automata is the undecidability [4] of the *language inclusion problem*: given two timed automata $\mathcal{A}$ and $\mathcal{B}$, does $L(\mathcal{A}) \subseteq L(\mathcal{B})$ hold? The undecidability of this problem prohibits the usage of automated verification algorithms for analysing timed automata, where $\mathcal{B}$ can be seen as the specification that is supposed to be satisfied by the system modelled by $\mathcal{A}$. However, if $\mathcal{B}$ is restricted to have at most one clock, then the language inclusion problem over finite timed words is decidable (albeit with non-primitive recursive complexity) [31]. Another milestone in the success story of timed automata is the decidability of the *model checking problem* for timed automata and Metric Temporal Logic (MTL, for short) over finite timed words [32].

Timed automata can express many interesting time-related properties, and even with the restriction to a single clock, they allow one to model a large class of systems, including, for example, the internet protocol TCP [30]. If we want to reason about real-time programs with procedure calls, or about the number of events occurring in computations of real-time systems, we have to extend the model of timed automata with some unbounded discrete data structure. In 1994, Bouajjani et al. [7] extended timed automata with discrete counters and a pushdown stack and proved that the satisfiability of reachability properties for several subclasses of this model is decidable. Nine years later, it was shown that

[⋆] The author is supported by *DFG*, project QU 316/1-1.

the binary reachability relation for *timed pushdown systems* is decidable [14]. Decidability of the reachability problem was also proved for several classes of *timed counter systems* [8], mainly by simple extensions of the classical region-graph construction [4]. The language inclusion problem, however, is to the best of our knowledge only considered in [18] for the class of timed pushdown systems. In [18] it is stated that the language inclusion problem is decidable if $\mathcal{A}$ is a timed pushdown automaton, and $\mathcal{B}$ is a one-clock timed automaton. The proof is based on an extension of the proof for the decidability of the language inclusion problem for the case that $\mathcal{A}$ is a timed automaton without pushdown stack [31]. Unfortunately, and as is well known, the proof in [18] is not correct.

In this paper, we prove that different to what is claimed in [18], the language inclusion problem for the case that $\mathcal{A}$ is a pushdown timed automaton and $\mathcal{B}$ is a one-clock timed automaton is undecidable. This is even the case if $\mathcal{A}$ is a deterministic instance of a very restricted subclass of timed pushdown automata called *timed visibly one-counter nets*. On the other hand, we prove that the language inclusion problem is decidable if $\mathcal{A}$ is a timed automaton and $\mathcal{B}$ is a timed automaton extended with a finite set of counters that can be incremented and decremented, and which we call *timed counter nets*. As a special case, we obtain the decidability of the *universality problem* for timed counter nets: given a timed automaton $\mathcal{A}$ with input alphabet Σ, does $L(\mathcal{A})$ accept the set of all timed words over Σ? Finally, we give the precise decidability border for the universality problem by proving that the universality problem is undecidable for the class of *timed visibly one-counter automata*. We remark that all results apply to extensions of timed automata over *finite* timed words.

2 Extensions of Timed Automata with Discrete Data Structure

We use $\mathbb{Z}$, $\mathbb{N}$ and $\mathbb{R}_{\geq 0}$ to denote the integers, the non-negative integers and the non-negative reals, respectively.

We use Σ to denote a finite alphabet. A *timed word* over Σ is a non-empty finite sequence $(a_1, t_1) \ldots (a_k, t_n) \in (\Sigma \times \mathbb{R}_{\geq 0})^+$ such that the sequence $t_1, \ldots, t_n$ of timestamps is non-decreasing. We say that a timed word is *strictly monotonic* if $t_{i-1} < t_i$ for every $i \in \{2, \ldots, n\}$. We use $T\Sigma^+$ to denote the set of finite timed words over Σ. A set $L \subseteq T\Sigma^+$ is called a *timed language*.

Let $\mathcal{X}$ be a finite set of *clock variables* ranging over $\mathbb{R}_{\geq 0}$. We define *clock constraints* ϕ over $\mathcal{X}$ to be conjunctions of formulas of the form $x \sim c$, where $x \in \mathcal{X}$, $c \in \mathbb{N}$, and $\sim \in \{<, \leq, =, \geq, >\}$. We use $\Phi(\mathcal{X})$ to denote the set of all clock constraints over $\mathcal{X}$. A *clock valuation* is a mapping from $\mathcal{X}$ to $\mathbb{R}_{\geq 0}$. A clock valuation ν satisfies a clock constraint ϕ, written $\nu \models \phi$, if ϕ evaluates to true according to the values given by ν. For $\delta \in \mathbb{R}_{\geq 0}$ and $\lambda \subseteq \mathcal{X}$, we define $\nu + \delta$ to be $(\nu + \delta)(x) = \nu(x) + \delta$ for each $x \in \mathcal{X}$, and we define $\nu[\lambda := 0]$ by $(\nu[\lambda := 0])(x) = 0$ if $x \in \lambda$, and $(\nu[\lambda := 0])(x) = \nu(x)$ otherwise.

Let Γ be a finite stack alphabet. We use Γ^* to denote the set of finite words over Γ, including the empty word denoted by ε. We define a finite set Op of *stack operations* by $\mathsf{Op} := \{pop(a), push(a) \mid a \in \Gamma\} \cup \{noop, empty?\}$.

A *timed pushdown automaton* is a tuple $\mathcal{A} = (\Sigma, \Gamma, \mathcal{L}, \mathcal{L}_0, \mathcal{L}_f, \mathcal{X}, E)$, where

- $\mathcal{L}$ is a finite set of *locations*,
- $\mathcal{L}_0 \subseteq \mathcal{L}$ is the set of initial locations,
- $\mathcal{L}_f \subseteq \mathcal{L}$ is the set of accepting locations,
- $E \subseteq \mathcal{L} \times \Sigma \times \Phi(\mathcal{X}) \times \mathsf{Op} \times 2^{\mathcal{X}} \times \mathcal{L}$ is a finite set of edges.

A *state* of $\mathcal{A}$ is a triple (l, ν, u), where $l \in \mathcal{L}$ is the current location, the clock valuation ν represents the current values of the clocks, and $u \in \Gamma^*$ represents the current stack content, where the top-most symbol of the stack is the left-most symbol in the word u, and the empty word ε represents the empty stack. We use $\mathcal{G}^{\mathcal{A}}$ to denote the set of all states of $\mathcal{A}$. A timed pushdown automaton $\mathcal{A}$ induces a transition relation $\Rightarrow_{\mathcal{A}}$ on $(\mathcal{G}^{\mathcal{A}} \times \mathbb{R}_{\geq 0} \times \Sigma \times \mathcal{G}^{\mathcal{A}})$ as follows: $\langle (l, \nu, u), \delta, a, (l', \nu', u') \rangle \in \Rightarrow_{\mathcal{A}}$, if, and only if, there exists some edge $(l, a, \phi, \mathsf{op}, \lambda, l') \in E$ such that $(\nu + \delta) \models \phi$, $\nu' = (\nu + \delta)[\lambda := 0]$, and (i) if $\mathsf{op} = \mathit{pop}(a)$ for some $a \in \Gamma$, then $u = a \cdot u'$; (ii) if $\mathsf{op} = \mathit{push}(a)$ for some $a \in \Gamma$, then $u' = a \cdot u$; (iii) if $\mathsf{op} = \mathit{empty?}$, then $u = u' = \varepsilon$; (iv) if $\mathsf{op} = \mathit{noop}$, then $u' = u$. A *run* of $\mathcal{A}$ is a finite sequence $\prod_{1 \leq i \leq n} \langle (l_{i-1}, \nu_{i-1}, u_{i-1}), \delta_i, a_i, (l_i, \nu_i, u_i) \rangle$ such that $\langle (l_{i-1}, \nu_{i-1}, u_{i-1}), \delta_i, a_i, (l_i, \nu_i, u_i) \rangle \in \Rightarrow_{\mathcal{A}}$ for every $i \in \{1, \ldots, n\}$. A run is called *successful* if $l_0 \in \mathcal{L}_0$, $\nu_0(x) = 0$ for every $x \in \mathcal{X}$, $u_0 = \varepsilon$, and $l_n \in \mathcal{L}_f$. With a run we associate the timed word $(a_1, \delta_1)(a_2, \delta_1 + \delta_2) \ldots (a_n, \Sigma_{1 \leq i \leq n} \delta_i)$. The language accepted by a timed automaton, denoted by $L(\mathcal{A})$, is defined to be the set of timed words $w \in T\Sigma^+$ for which there exists a successful run of $\mathcal{A}$ that w is associated with.

Next we define some subclasses of timed pushdown automata; see Fig. 1 for a graphical overview. We start with timed extensions of *one-counter automata* [16,28] and *one-counter nets* [24,1]. A *timed one-counter automaton* is a timed pushdown automaton where the stack alphabet is a singleton. By writing *push* and *pop* we mean that we increment and decrement the counter, respectively, whereas *empty*? corresponds to a zero test. A *timed one-counter net* is a timed one-counter automaton without zero tests, *i.e.*, the *empty*? operation is not allowed. We remark that for both classes, the execution of an edge of the form $(l, a, \phi, \mathit{pop}, \lambda, l')$ is *blocked* if the stack is empty. Next, we consider the timed extension of an interesting subclass of pushdown automata called *visibly pushdown automata* [5]. A *timed visibly pushdown automaton* is a timed pushdown automaton for which the input alphabet Σ can be partitioned into three pairwise disjoint sets $\Sigma = \Sigma_{\mathsf{int}} \cup \Sigma_{\mathsf{call}} \cup \Sigma_{\mathsf{ret}}$ of *internal*, *call*, and *return* input symbols, respectively, and such that for every edge $(l, a, \phi, \mathsf{op}, \lambda, l')$ the following conditions are satisfied:

- $a \in \Sigma_{\mathsf{int}}$ if, and only if, $\mathsf{op} = \mathit{noop}$,
- $a \in \Sigma_{\mathsf{call}}$, if, and only if, $\mathsf{op} = \mathit{push}(b)$ for some $b \in \Gamma$,
- $a \in \Sigma_{\mathsf{ret}}$ if, and only if, $\mathsf{op} = \mathit{empty?}$ or $\mathsf{op} = \mathit{pop}(b)$ for some $b \in \Gamma$.

A *timed visibly one-counter automaton* (timed visibly one-counter net, respectively) is a timed one-counter automaton (timed one-counter net, respectively) that is also a timed visibly pushdown automaton. We say that a timed visibly one-counter net with no clocks is *deterministic* if for all $e = (l, a, \texttt{true}, \mathsf{op}', \emptyset, l')$, $e' = (l, a, \texttt{true}, \mathsf{op}'', \emptyset, l'') \in E$ with $e \neq e'$ we have either $\mathsf{op}' = \mathit{pop}$ and $\mathsf{op}'' = \mathit{empty?}$, or $\mathsf{op}' = \mathit{empty?}$ and $\mathsf{op}'' = \mathit{pop}$.

Finally, we define the class of *timed counter nets*, which generalizes timed one-counter nets, but is not a subclass of timed pushdown automata. A timed counter net of dimension n is a tuple $\mathcal{A} = (\Sigma, n, \mathcal{L}, \mathcal{L}_0, \mathcal{L}_f, \mathcal{X}, E)$, where $\mathcal{L}, \mathcal{L}_0, \mathcal{L}_f$ are the sets of locations, initial locations and accepting locations, respectively, and $E \subseteq \mathcal{L} \times \Sigma \times \Phi(\mathcal{X}) \times \{0, 1, -1\}^n \times 2^{\mathcal{X}} \times \mathcal{L}$ is a finite set of edges. A state of a timed counter net is a triple $(l, \nu, \boldsymbol{v})$, where $l \in \mathcal{L}$, ν is a clock valuation, and $\boldsymbol{v} \in \mathbb{N}^n$ is a vector representing the current values of the counters. We define $\langle (l, \nu, \boldsymbol{v}), \delta, a, (l', \nu', \boldsymbol{v'}) \rangle \in \Rightarrow_{\mathcal{A}}$ if, and only if, there exists some edge $(l, a, \phi, \boldsymbol{c}, \lambda, l') \in E$ such that $(\nu + \delta) \models \phi$, $\nu' = (\nu + \delta)[\lambda := 0]$, and $\boldsymbol{v'} = \boldsymbol{v} + \boldsymbol{c}$, where vector addition is defined pointwise. Note that, similar to pop operations on an empty stack, transitions which result in the negative value of one of the counters are *blocked*. The notions of *runs*, *successful runs*, *associated timed words* and *the language accepted by* $\mathcal{A}$, are defined analogously to the corresponding definitions for timed pushdown automata.

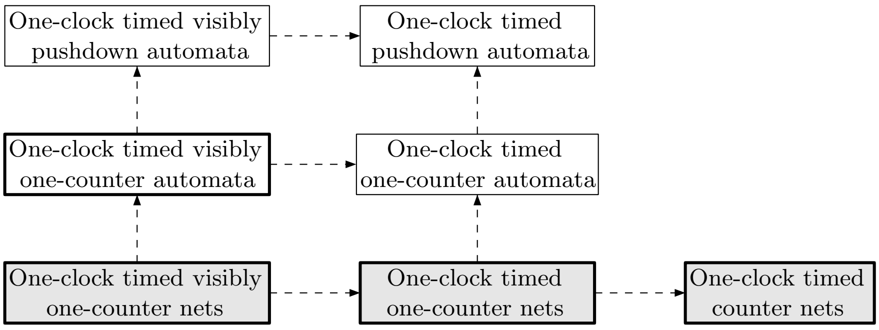
One-clock timed visibly pushdown automata
One-clock timed pushdown automata
One-clock timed visibly one-counter automata
One-clock timed one-counter automata
One-clock timed visibly one-counter nets
One-clock timed one-counter nets
One-clock timed counter nets

Fig. 1. Extensions of one-clock timed automata with discrete data structures. The subclass relation is represented by dashed arrows. The language emptiness problem is decidable for all classes. The classes in grey boxes have a decidable universality problem, the classes in white boxes have an undecidable universality problem, where the corresponding results for classes in boxes with bold line are new and presented in this paper.

3 Main Results

In this section, we present the main results of the paper. We are interested in the language inclusion problem $L(\mathcal{A}) \subseteq L(\mathcal{B})$, where $\mathcal{A}$ and $\mathcal{B}$ are extensions of timed automata with discrete data structure. Recall that according to standard notation in the field of verification, in this problem formulation $\mathcal{B}$ is seen as the *specification*, and $\mathcal{A}$ is the system that should satisfy this specification, *i.e.*, $\mathcal{A}$ should be a *model* of $\mathcal{B}$. As a special case of this problem, we consider the universality problem, *i.e.*, the question whether $L(\mathcal{A}) = T\Sigma^+$ for a given automaton

$\mathcal{A}$. In general, the two problems are undecidable for timed pushdown automata. This follows on the one hand from the undecidability of the universality problem for timed automata [4], and on the other hand from the undecidability of the universality problem for pushdown automata. In fact, it is long known that the universality problem is undecidable already for non-deterministic one-counter automata [22,26].

However, there are interesting decidability results for subclasses of timed pushdown automata: The language inclusion problem is decidable if $\mathcal{A}$ is a timed automaton, and $\mathcal{B}$ is a timed automaton with at most one clock [31]. As a special case, the universality problem for timed automata is decidable if only one clock is used. The language inclusion problem is also decidable if $\mathcal{A}$ is a one-counter net and $\mathcal{B}$ is a finite automaton, and if $\mathcal{A}$ is a finite automaton and $\mathcal{B}$ is a one-counter net [27]. The universality problem for non-deterministic one-counter nets has recently been proved to have non-primitive recursive complexity [25]. Further we know that the universality and language inclusion problems are decidable if $\mathcal{A}$ and $\mathcal{B}$ are visibly pushdown automata [6].

Hence it is interesting to consider the two problems for the corresponding subclasses of timed pushdown automata. It turns out that the decidability status changes depending on whether the *model* uses a stack (or, more detailed: a counter) or not. As a first main result, we have:

Theorem 1. *The language inclusion problem is undecidable if $\mathcal{A}$ is a timed visibly one-counter net and $\mathcal{B}$ is a timed automaton, even if $\mathcal{A}$ is deterministic and has no clocks, and $\mathcal{B}$ uses at most one clock.*

We remark that this result corrects a claim concerning the decidability of the language inclusion problem if $\mathcal{A}$ is a timed pushdown automaton and $\mathcal{B}$ is a one-clock timed automaton, stated in Theorem 2 in [18]. In contrast to Theorem 1, we have decidability for the following classes:

Theorem 2. *The language inclusion problem is decidable with non-primitive recursive complexity if $\mathcal{A}$ is a timed automaton and $\mathcal{B}$ is a one-clock timed counter net.*

As a special case of this result (and with the lower bound implied by the corresponding result for one-clock timed automata [2]), we obtain:

Corollary 1. *The universality problem for one-clock timed counter nets is decidable with non-primitive recursive complexity.*

The next two sections are devoted to the proofs of Theorems 1 and 2. We will also give some interesting consequences of these results respectively of their proofs. Amongst others, we prove the undecidability of model checking problem for timed visibly one-counter nets and MTL over finite timed words. After this, in Sect. 5, we will prove the following theorem:

Theorem 3. *The universality problem for one-clock timed visibly one-counter automata is undecidable.*

This is in contrast to the decidability of the universality problem for the two underlying models of one-clock timed automata [31] and visibly one-counter automata, which form a subclass of visibly pushdown automata [6]. We also want to point out that this result is stronger than a previous result on the undecidability of the universality problem for one-clock timed visibly pushdown automata (Theorem 3 in [18]), and our proof closes a gap in the proof of Theorem 3 in [18]. Further, we can infer from Corollary 1 and Theorem 3 the exact decidability border for the universality problem of timed pushdown automata, which lies between timed visibly one-counter nets and timed visibly one-counter automata.

4 Undecidability Results

In this section, we prove Theorem 1. The proof is a reduction of an undecidable problem for *channel machines.*

4.1 Channel Machines

Let A be a finite alphabet. We define the order $\leq$ over the set of finite words over A by $a_1a_2\ldots a_m \leq b_1b_2\ldots b_n$ if there exists a strictly increasing function $f : \{1,\ldots,m\} \to \{1,\ldots,n\}$ such that $a_i = b_{f(i)}$ for every $i \in \{1,\ldots,m\}$.

A *channel machine* consists of a finite-state automaton acting on an unbounded fifo channel. Formally, a channel machine is a tuple $\mathcal{C} = (S, s_I, M, \Delta)$, where

- S is a finite set of *control states*,
- $s_I \in S$ is the initial control state,
- M is a finite set of *messages*,
- $\Delta \subseteq S \times L \times S$ is the transition relation over the label set $L = \{!m, ?m \mid m \in M\} \cup \{\mathit{empty}?\}$.

Here, $!m$ corresponds to a *send* operation, $?m$ corresponds to a *read* operation, and *empty*? is a test which returns `true` if and only if the channel is empty. Without loss of generality, we assume that s_I does not have any incoming transitions, *i.e.*, $(s, l, s') \in \Delta$ implies $s' \neq s_I$. Further, we assume that $(s_I, l, s') \in \Delta$ implies $l = \mathit{empty}?$. A *configuration* of $\mathcal{C}$ is a pair (s, x), where $s \in S$ is the control state and $x \in M^*$ represents the contents of the channel. We use $\mathcal{H}^{\mathcal{C}}$ to denote the set of all configurations of $\mathcal{C}$. The rules in Δ induce a transition relation $\to_{\mathcal{C}}$ on $(\mathcal{H}^{\mathcal{C}} \times L \times \mathcal{H}^{\mathcal{C}})$ as follows:

- $\langle (s, x), !m, (s', x') \rangle \in \to_{\mathcal{C}}$ if, and only if, there exists some transition $(s, !m, s') \in \Delta$ and $x' = x \cdot m$, *i.e.*, m is added to the tail of the channel.
- $\langle (s, x), ?m, (s', x') \rangle \in \to_{\mathcal{C}}$ if, and only if, there exists some transition $(s, ?m, s') \in \Delta$ and $x = m \cdot x'$, *i.e.*, m is removed from the head of the channel.
- $\langle (s, x), \mathit{empty}?, (s', x') \rangle \in \to_{\mathcal{C}}$ if, and only if, there exists some transition $(s, \varepsilon, s') \in \Delta$ and $x = \varepsilon$, *i.e.*, the channel is empty, and $x' = x$.

Next, we define a second transition relation $\rightsquigarrow_{\mathcal{C}}$ on $(\mathcal{H}^{\mathcal{C}} \times L \times \mathcal{H}^{\mathcal{C}})$. The relation $\rightsquigarrow_{\mathcal{C}}$ is a superset of $\rightarrow_{\mathcal{C}}$. It contains some additional transitions which result from *insertion errors*. We define $\langle (s, x_1), l, (s, x_1') \rangle \in \rightsquigarrow_{\mathcal{C}}$, if, and only if, there exist $x, x' \in M^*$ such that $\langle (s, x), l, (s', x') \rangle \in \rightarrow_{\mathcal{C}}$, $x_1 \leq x$, and $x' \leq x_1'$. A *computation* of $\mathcal{C}$ is a finite sequence $\prod_{1 \leq i \leq k} \langle (s_{i-1}, x_{i-1}), l_i, (s_i, x_i) \rangle$ such that $\langle (s_{i-1}, x_{i-1}), l_i, (s_i, x_i) \rangle \in \rightsquigarrow_{\mathcal{C}}$ for every $i \in \{1, \ldots, k\}$. We say that a computation is *error-free* if for all $i \in \{1, \ldots, k\}$ we have $\langle (s_{i-1}, x_{i-1}), l_i, (s_i, x_i) \rangle \in \rightarrow_{\mathcal{C}}$. Otherwise, we say that the computation is *faulty*.

The proof of Theorem 1 is a reduction from the following undecidable [12,2] control state reachability problem: given a channel machine $\mathcal{C}$ with control states S and $s_F \in S$, does there exist an error-free computation of $\mathcal{C}$ from (s_I, ε) to (s_F, x) for some $x \in M^*$? We remark that the analogous problem for faulty computations is decidable [33]. The idea of our reduction is as follows: Given a channel machine $\mathcal{C}$, we define a timed language $L(\mathcal{C})$ consisting of all timed words that encode potentially faulty computations of $\mathcal{C}$ that start in (s_I, ε) and end in (s_F, x) for some $x \in M^*$. Then we define a timed visibly one-counter net $\mathcal{A}$ such that $L(\mathcal{A}) \cap L(\mathcal{C})$ contains exactly *error-free* encodings of such computations. In other words, we use $\mathcal{A}$ to exclude the encodings of faulty computations from $L(\mathcal{C})$, obtaining undecidability of the non-emptiness problem for $L(\mathcal{A}) \cap L(\mathcal{C})$. Finally, we define a one-clock timed automaton $\mathcal{B}$ that accepts the complement of $L(\mathcal{C})$; hence the problem of deciding whether $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$ is undecidable.

4.2 Encoding Faulty Computations

For the remainder of Section 4, let $\mathcal{C} = (S, s_I, M, \Delta)$ be a channel machine and let $s_F \in S$. Define $\Sigma_{\mathsf{int}} := (S \backslash \{s_I\}) \cup M \cup L \cup \{\#\}$, $\Sigma_{\mathsf{call}} := \{s_I, +\}$, and $\Sigma_{\mathsf{ret}} := \{-, \star\}$, where $+, -, \#$ and $\star$ are fresh symbols that do not occur in $S \cup M \cup L$. We define a timed language $L(\mathcal{C})$ over $\Sigma = \Sigma_{\mathsf{int}} \cup \Sigma_{\mathsf{call}} \cup \Sigma_{\mathsf{ret}}$ that consists of all timed words that encode computations of $\mathcal{C}$ from (s_I, ε) to (s_F, x) for some $x \in M^*$. The definition of $L(\mathcal{C})$ follows the ideas presented in [32].

Let $\gamma = \prod_{1 \leq i \leq k} \langle (s_{i-1}, x_{i-1}), l_i, (s_i, x_i) \rangle$ be a computation of $\mathcal{C}$ with $s_0 = s_I$, $x_0 = \varepsilon$, and $s_k = s_F$. For each $i \in \{0, \ldots, k\}$, the configuration (s_i, x_i) is encoded by a timed word of duration one. This timed word starts with the symbol s_i at some time t_i. If the content of the channel x_i is of the form $m_1 m_2 \ldots m_j$, then s_i is followed by the symbols $m_1, m_2, \ldots, m_j$ in this order. The timestamps of these symbols must be in the interval $(t_i, t_i + 1)$. Due to the denseness of the time domain, one can indeed store the channel content in one time unit without any upper bound on j. For encoding the computation, we glue together the encodings of the single configurations as follows: Every control state symbol s_{i-1} is followed by l_i after exactly one time unit, and by s_i after exactly two time units. For every message symbol m between s_{i-1} and l_i, there is a copy of m after two time units, unless it is removed from the channel by a read operation. There are no symbols between l_i and s_i, and the encoding of the last configuration ends with $\star$ one time unit after s_k. Note: we *do not* require that for every message symbol m between s_{i-1} and l_i there is a copy of m two time units *before*. It is the absence of exactly this condition that causes $L(\mathcal{C})$ to contain encodings of *faulty* computations.

For our reduction to work, we change the idea from [32] in some details. As mentioned above, we will later define a timed visibly one-counter net $\mathcal{A}$ to exclude faulty computations from $L(\mathcal{C})$. The idea is to let $\mathcal{A}$ *guess* the maximum number n of messages occurring between control state symbols and the following label symbol. While reading the encoding of the first configuration of a computation, it increments the counter n times. The automaton does not do any operations on the counter until it reads the encoding of the last configuration, where it decrements the counter whenever it reads symbols occurring between the control state and the label symbol. Since it can decrement the counter at most n times, it can only accept encodings of error-free computations. We define a timed language

$$\text{(a)}\quad \begin{array}{|c|c|c|}\hline + & + & + \\ \hline 1.2 & 1.7 & 1.8 \\ \hline\end{array} \overset{!m}{\rightarrow}_{\mathcal{C}} \begin{array}{|c|c|c|}\hline m & \# & \# \\ \hline 3.2 & 3.7 & 3.8 \\ \hline\end{array} \overset{!m'}{\rightarrow}_{\mathcal{C}} \begin{array}{|c|c|c|}\hline m & m' & \# \\ \hline 5.2 & 5.7 & 5.8 \\ \hline\end{array} \overset{?m}{\rightarrow}_{\mathcal{C}} \begin{array}{|c|c|c|}\hline m' & \# & \# \\ \hline 7.2 & 7.7 & 7.8 \\ \hline\end{array} \overset{?m}{\rightsquigarrow}_{\mathcal{C}} \begin{array}{|c|c|c|c|}\hline m' & \# & \# & \# \\ \hline 9.2 & 9.7 & 9.8 & 9.85 \\ \hline\end{array}$$

$$\text{(b)}\quad \begin{array}{|c|c|c|c|}\hline m & m' & m' & m \\ \hline 15.2 & 15.7 & 15.8 & 15.85 \\ \hline\end{array} \overset{!m}{\rightsquigarrow}_{\mathcal{C}} \begin{array}{|c|c|c|c|c|}\hline m & m' & m' & m & m \\ \hline 17.2 & 17.7 & 17.8 & 17.85 & 17.9 \\ \hline\end{array} \overset{?m}{\rightarrow}_{\mathcal{C}} \begin{array}{|c|c|c|c|c|}\hline - & - & - & - & - \\ \hline 19.2 & 19.7 & 19.8 & 19.85 & 19.9 \\ \hline\end{array}$$

Fig. 2. Examples for the encoding of the channel content for $n = 3$

$L(\mathcal{C}, n)$ for every $n \in \mathbb{N}$. For illustrating the definition, we use the examples in Fig. 2. Since *push*, *pop* and *noop* require symbols from the call, internal, and return input alphabet, respectively, we use three fresh extra symbols $+, \#$, and $-$. In the encoding of the initial configuration, we use n occurrences of the call symbol $+$ as placeholder for message symbols, see the first timed word in Fig. 2(a). In the encoding of all following configurations except for the last one we use the internal symbol $\#$ as placeholder. In the encoding of the last configuration, we use the return symbol $-$ as placeholder, see the last timed word in Fig. 2(b). Every time some $!m$ operation occurs, all symbols in the encoding of the current configuration have a copy after exactly two time units, except for the first free placeholder symbol, which is replaced by m in the encoding of the next configuration, see for instance the first two transitions in Fig. 2(a). Every time an error-free $?m$ operation occurs, the first message symbol in the encoding of the current configuration (which should be m) is replaced by a new placeholder symbol at the end of the encoding of the next configuration, and the timestamps of the other symbols are shifted one position, see the third transition in Fig. 2(a). A faulty read operation due to an insertion error is encoded by the insertion of a new placeholder symbol at the end of the encoding of the next configuration, see the last transition in Fig. 2(a). We also insert a new symbol if a faulty send operation due to a too small choice of n is happening (first transition in Fig. 2(b)). Note, however, that this transition is not faulty due to an insertion error of the channel machine.

Let $w \in L(\mathcal{C}, n)$ for some $n \in \mathbb{N}$. We use $\max(w)$ to denote the maximum number of symbols in $M \cup \{\#, +, -\}$ that occur in w between a control state symbol and a symbol in $L \cup \{\star\}$. Let $\gamma = \prod_{1 \leq i \leq k} \langle (s_{i-1}, x_{i-1}), l_i, (s_i, x_i) \rangle$ be a computation of $\mathcal{C}$. We use $\max(\gamma)$ to denote the maximum length of the channel content occurring in γ, formally: $\max(\gamma) := \max\{|x_i| \mid 0 \leq x_i \leq k\}$.

Lemma 1. *For each error-free computation γ of $\mathcal{C}$ from (s_I, ε) to (s_F, x) for some $x \in M^*$, there exists some timed word $w \in L(\mathcal{C}, \max(\gamma))$ such that* $\max(w) = \max(\gamma)$*.*

Lemma 2. *For each $n \in \mathbb{N}$ and $w \in L(\mathcal{C}, n)$ with* $\max(w) = n$*, there exists some error-free computation γ of $\mathcal{C}$ from (s_I, ε) to (s_F, x) for some $x \in M^*$ with* $\max(\gamma) \leq n$*.*

4.3 Excluding Faulty Computations

We define a timed visibly one-counter net $\mathcal{A}$ over Σ such that for every $n \in \mathbb{N}$ the intersection $L(\mathcal{A}) \cap L(\mathcal{C}, n)$ consists of all timed words that encode *error-free* computations of $\mathcal{C}$ from (s_I, ε) to (s_F, x) for some $x \in M^*$. The timed visibly one-counter net $\mathcal{A}$ is shown in Fig. 3. It non-deterministically guesses a number $n \in \mathbb{N}$ of symbols $+$ and increments the counter each time it reads the symbol $+$. When $\mathcal{A}$ leaves l_1, the value of the counter is $n+1$. After that, the counter value is not changed until the state symbol s_F is read. Then, while reading symbols in $\{-, \star\}$, the counter value is decremented. Note that $\mathcal{A}$ can reach the final location l_4 only if the number of the occurrences of symbol $-$ between s_F and $\star$ is *at most* n. Note that $\mathcal{A}$ does not use any clock, and it is deterministic.

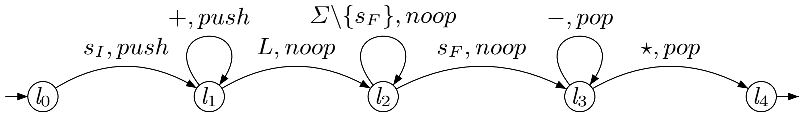

Fig. 3. The deterministic timed visibly one-counter net $\mathcal{A}$ for excluding insertion errors

Lemma 3. *$\mathcal{C}$ has an error-free computation from (s_I, ε) to (s_F, x) for some $x \in M^*$, if, and only if, there exists $n \in \mathbb{N}$ such that $L(\mathcal{C}, n) \cap L(\mathcal{A}) \neq \emptyset$.*

We finally define $L(\mathcal{C}) := \bigcup_{n \in \mathbb{N}} L(\mathcal{C}, n)$.

Corollary 2. *There exists some error-free computation of $\mathcal{C}$ from (s_I, ε) to (s_F, x) for some $x \in M^*$ if, and only if, $L(\mathcal{A}) \cap L(\mathcal{C}) \neq \emptyset$.*

4.4 The Reduction

Finally, we define a one-clock timed automaton $\mathcal{B}$ such that $L(\mathcal{B}) = T\Sigma^+ \backslash L(\mathcal{C})$. The construction of $\mathcal{B}$ follows the same ideas as, *eg.*, in [3]: $\mathcal{B}$ is the union of several one-clock timed automata, each of them violating one of the conditions of the definition of $L(\mathcal{C})$. For instance, the timed automaton in Fig. 4 accepts the set of timed words over Σ violating the condition that for every control state

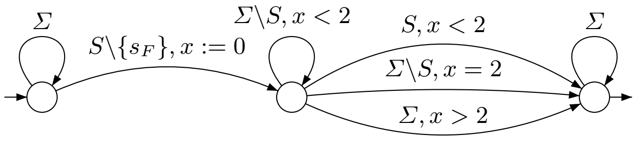
Σ
$S\backslash\{s_F\}, x := 0$
$\Sigma\backslash S, x < 2$
$S, x < 2$
$\Sigma\backslash S, x = 2$
$\Sigma, x > 2$
Σ

Fig. 4. A timed automaton violating condition 2 of $L(\mathcal{C})$

symbol different from s_F there must be some control state symbol after two time units.

By Corollary 2, there exists some error-free computation of $\mathcal{C}$ from (s_I, ε) to (s_F, x) for some $x \in M^*$ if, and only if, $L(\mathcal{A}) \cap L(\mathcal{C}) \neq \emptyset$. The latter is equivalent to $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$. Hence, the language inclusion problem is undecidable. $\square$

4.5 Undecidability of the Model Checking Problem for MTL

The proof idea of Theorem 1 can be used to show the undecidability of the following model checking problem: given a timed visibly one-counter net $\mathcal{A}$, and an MTL formula φ, is every $w \in L(\mathcal{A})$ a model of φ? Recall that this problem is decidable for the class of timed automata [32]. We prove that adding a visibly counter without zero test already makes the problem undecidable. Recall that MTL only allows to express restrictions on time, and it does not allow for any restrictions on the values of the counters. In fact, it is known that as soon as we add to MTL the capability for expressing restrictions on the values of a counter that can be incremented and decremented, model checking is undecidable [35]. The proof of the following theorem is based on the fact that - like one-clock timed automata - MTL can encode computations of channel machines with insertion errors [32].

Theorem 4. *The model checking problem for timed visibly one-counter nets and MTL is undecidable, even if the timed visibly one-counter net does not use any clocks and is deterministic.*

We would like to remark that the proof of Theorem 4 shares some similarities with the proof of the undecidability of model checking one-counter machines (*i.e.*, one-counter automata without input alphabet) and Freeze LTL with one register ($\mathrm{LTL}_1^{\downarrow}$, for short) [16]. In [15], it is proved that $\mathrm{LTL}_1^{\downarrow}$ can encode computations of *counter automata with incrementing errors*. Similar to the situation for MTL and channel machines, $\mathsf{LTL}_1^{\downarrow}$ can however not encode *error-free* computations of counter automata. In [16], a one-counter machine is used to repair this incapability, resulting in the undecidability of the model checking problem. The one-counter machine in [16] does not use zero tests; however, we point out that in contrast to our visibly timed one-counter net the one-counter machine in [16] is *non-deterministic*. Indeed, model checking *deterministic* one-counter machines and $\mathrm{LTL}_1^{\downarrow}$ is decidable [16].

4.6 Energy Problems on Timed Automata with Discrete Weights

Next we will consider an interesting extension of *lower-bound energy problems on weighted timed automata*, introduced in [10], which gained attention in the last years, see, *eg.*, [11,34,9]. In lower-bound energy problems, one is interested whether in a given automaton with some weight variable whose value can be increased and decreased, there exists a successful run in which all accumulated weight values are never below zero. Similar problems have also been considered for untimed settings, *eg.*, [29,19,20,13].

A *timed automaton with discrete weights* (dWTA, for short) is syntactically the same as a timed one-counter net. In the semantical graph induced by a dWTA, however, we allow the value of the counter (or, the *weight variable*) to become negative. Hence the value of the weight variable does not influence the behaviour of the dWTA, because, different to timed one-counter nets, transitions that result in negative values are not blocked. We remark that for the simple reasons that the value of the weight variable does not influence the behaviour of dWTA and MTL does not restrict the values of the weight variable, the model checking problem for dWTA and MTL is decidable, using the same algorithm as for timed automata [32]. We define the *energy model checking problem* for dWTA and MTL as follows: given a dWTA $\mathcal{A}$ and an MTL formula φ, does there exist some accepting run ρ of $\mathcal{A}$ such that the value of the weight variable is always non-negative, and the timed word w associated with ρ satisfies φ? For the special case $\varphi = \texttt{true}$, the problem is decidable in polynomial time for one-clock dWTA [10].

Theorem 5. *The energy problem for dWTA and MTL is undecidable, even if the dWTA uses no clocks.*

5 Decidability Result

In this section, we shortly explain the proof idea for the decidability of the language inclusion problem if $\mathcal{A}$ is a timed automaton, and $\mathcal{B}$ is a one-clock timed counter net. The proof is a generalization of the proof for the case where both $\mathcal{A}$ and $\mathcal{B}$ are timed automata [31]. The idea is to reduce the language inclusion problem to a reachability problem on an infinite graph constructed from the joint state space of $\mathcal{A}$ and $\mathcal{B}$. The decidability of the reachability problem on our infinite graph is implied by the fact that the graph is a downward-compatible well-structured transition system [21]. For taking into account the additional information on the values of the counters, we have to define a new well-quasi-order on the state space of the graph. This new well-quasi-order is based on the product of the equality order $=$ on a finite alphabet and the pointwise order $\leq^n$ on $\mathbb{N}^n$ (where n is the number of counters of $\mathcal{B}$), which is a well-quasi-order by Dickson's Lemma [17]. We use several applications of Higman's Lemma [23] to prove that our quasi-order is a well-quasi-order.

6 The Universality Problem for Visibly One-Counter Automata

We prove that allowing the counter in a one-clock timed visibly one-counter net to be tested for zero, results in the undecidability of the universality problem. The undecidability of the universality problem for the more general class of one-clock visibly pushdown automata was already stated in Theorem 3 in [18]. The proof in [18] is a reduction of the halting problem for two-counter machines. Given a two-counter machine $\mathcal{M}$, one can define a timed language $L(\mathcal{M})$ that consists of all timed words encoding a halting computation of $\mathcal{M}$. Then a timed visibly pushdown automaton $\mathcal{A}$ is defined that accepts the complement of $L(\mathcal{M})$. Altogether, $L(\mathcal{A}) = T\Sigma^+$ if, and only if, $\mathcal{M}$ does not have a halting computation. The definition of $L(\mathcal{M})$ is similar to the definition of $L(\mathcal{C})$ in the proof of Theorem 1. Recall that in the definition of $L(\mathcal{C})$ we did not include a condition that requires every symbol to have a matching symbol two time units *before*, and, as we mentioned, this is the reason for $L(\mathcal{C})$ to contain timed words encoding *faulty* computations of $\mathcal{C}$. However, in the definition of $L(\mathcal{M})$ in [18], such a "backward-looking" condition is used. In the proof in [18], it is unfortunately not clear how the one-clock timed visibly pushdown automaton $\mathcal{A}$ can detect violations of this condition[1].

Here, we give a complete proof for the subclass of timed visibly one-counter automata. Like the proof of Theorem 1, the proof is a reduction of the control state reachability problem for channel machines. We however remark that one can similarly use a reduction of the halting problem for two-counter machines.

Proof of Theorem 3. Let $\mathcal{C} = (S, s_I, M, \Delta)$ be a channel machine, and let $s_F \in S$. Define Σ in the same way as in the proof of Theorem 1. For every $n \in \mathbb{N}$, we define a timed language $L_{\mathsf{ef}}(\mathcal{C}, n)$ that consists of all timed words over Σ that encode *error-free* computations of $\mathcal{C}$ from (s_I, ε) to (s_F, x) for some $x \in M^*$. Formally, $L_{\mathsf{ef}}(\mathcal{C}, n)$ is defined using the same conditions as the ones for $L(\mathcal{C}, n)$ in the proof of Theorem 1 plus an additional condition that requires every symbol in the encoding of a configuration to have a matching symbol two time units before. As we have mentioned in the proof of Theorem 1, this excludes encodings of faulty computations. We thus have for $L_{\mathsf{ef}}(\mathcal{C}) = \bigcup_{n \geq 1} L_{\mathsf{ef}}(\mathcal{C}, n)$:

Lemma 4. *There exists some error-free computation of $\mathcal{C}$ from (s_I, ε) to (s_F, x) for some $x \in M^*$, if, and only if, $L_{\mathsf{ef}}(\mathcal{C}) \neq \emptyset$.*

Next, we define a timed visibly one-counter automaton with a single clock such that $L(\mathcal{A}) = T\Sigma^+ \setminus L_{\mathsf{ef}}(\mathcal{C})$. Hence, by the preceding lemma, $L(\mathcal{A}) \neq T\Sigma^+$ if, and only if, there exists some error-free computation of $\mathcal{C}$ from (s_I, ε) to (s_F, x) for some $x \in M^*$.

[1] More detailed, it is not clear how to construct one-clock timed automata $N_{\neg f_r \leftarrow f_c}$ and $N_{\neg g_r \leftarrow g_c}$ mentioned on p. 10 in [18]. Recall that in the proof for undecidability of the universality problem for timed automata with two or more clocks, it is exactly this backward-looking condition that requires *two* clocks [3].

$\mathcal{A}$ is the union of several one-clock timed automata and one timed visibly one-counter automaton with no clocks. We already know from the proof of Theorem 1, that violations of conditions of $L(\mathcal{C}, n)$ can be detected by one-clock timed automata. For detecting violations of the additional condition, we use the visibly one-counter automaton shown in Fig. 5. The automaton non-deterministically guesses the maximum number n of occurrences of the symbol $+$. When leaving l_1, the value of the counter is $n + 1$. The final location l_4, however, can only be reached while reading $-$ or $\star$ if the value of the counter is zero. This means that there must be some symbol for which there is no matching symbol two time units before. $\square$

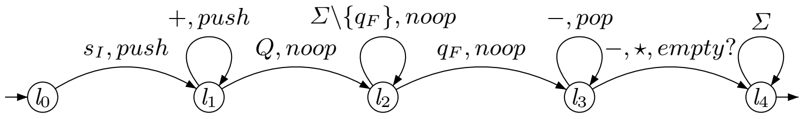

Fig. 5. The timed visibly one-counter automaton for recognizing timed words violating the additional "backwards-looking" condition of $L_{\mathsf{ef}}(\mathcal{C})$

7 Conclusion and Open Problems

The main conclusion of this paper is that even for very weak extensions of timed automata with counters it is impossible to automatically verify whether a given specification is satisfied. On the other hand, we may use one-clock timed counter nets as specifications to verify timed automata. This increases so far known possibilities for the verification of timed automata: For instance, the timed language $L = \{(a^m b^n, \bar{\tau}) \mid m \geq n\}$ can be accepted by a timed one-counter net without any clocks, but not by a timed automaton.

An interesting problem is to figure out a (decidable) extension of LTL that is capable of expressing properties referring to both time and discrete data structures.

We remark that all our results hold for automata defined over *finite* timed words. We cannot expect the decidability of, *eg.*, the universality problem for one-clock timed counter nets over infinite timed words, as the same problem is already undecidable for the subclass of one-clock timed automata [2].

Acknowledgements. I would like to thank Michael Emmi and Rupak Majumdar for helpful discussions on their work on timed pushdown automata. I further would like to thank James Worrell very much for pointing me to MTL's capability of encoding faulty computations of channel machines.

References

1. Abdulla, P.A., Cerans, K.: Simulation is decidable for one-counter nets (extended abstract). In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 253–268. Springer, Heidelberg (1998)

2. Abdulla, P.A., Deneux, J., Ouaknine, J., Quaas, K., Worrell, J.: Universality analysis for one-clock timed automata. Fundam. Inform. 89(4), 419–450 (2008)
3. Adams, S., Ouaknine, J., Worrell, J.B.: Undecidability of universality for timed automata with minimal resources. In: Raskin, J.-F., Thiagarajan, P.S. (eds.) FORMATS 2007. LNCS, vol. 4763, pp. 25–37. Springer, Heidelberg (2007)
4. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
5. Alur, R., Madhusudan, P.: Decision problems for timed automata: A survey. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. LNCS, vol. 3185, pp. 1–24. Springer, Heidelberg (2004)
6. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Babai, L. (ed.) STOC, pp. 202–211. ACM (2004)
7. Bouajjani, A., Echahed, R., Robbana, R.: On the automatic verification of systems with continuous variables and unbounded discrete data structures. In: Antsaklis, P., Kohn, W., Nerode, A., Sastry, S. (eds.) Hybrid Systems II. LNCS, vol. 999, pp. 64–85. Springer, Heidelberg (1995)
8. Bouchy, F., Finkel, A., Sangnier, A.: Reachability in timed counter systems. Electr. Notes Theor. Comput. Sci. 239, 167–178 (2009)
9. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N.: Timed automata with observers under energy constraints. In: Johansson, K.H., Yi, W. (eds.) HSCC, pp. 61–70. ACM (2010)
10. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 33–47. Springer, Heidelberg (2008)
11. Bouyer, P., Larsen, K.G., Markey, N.: Lower-bound-constrained runs in weighted timed automata. Perform. Eval. 73, 91–109 (2014)
12. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM 30(2), 323–342 (1983)
13. Brázdil, T., Jančar, P., Kučera, A.: Reachability games on extended vector addition systems with states. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 478–489. Springer, Heidelberg (2010)
14. Dang, Z.: Pushdown timed automata: a binary reachability characterization and safety verification. Theor. Comput. Sci. 302(1-3), 93–121 (2003)
15. Demri, S., Lazić, R.: LTL with the freeze quantifier and register automata. ACM Trans. Comput. Log. 10(3) (2009)
16. Demri, S., Lazić, R.S., Sangnier, A.: Model checking Freeze LTL over one-counter automata. In: Amadio, R.M. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 490–504. Springer, Heidelberg (2008)
17. Dickson, L.E.: Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. Amer. J. Math. 35, 413–422 (1913)
18. Emmi, M., Majumdar, R.: Decision problems for the verification of real-time software. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 200–211. Springer, Heidelberg (2006)
19. Ésik, Z., Fahrenberg, U., Legay, A., Quaas, K.: Kleene algebras and semimodules for energy problems. In: Van Hung, D., Ogawa, M. (eds.) ATVA 2013. LNCS, vol. 8172, pp. 102–117. Springer, Heidelberg (2013)
20. Fahrenberg, U., Juhl, L., Larsen, K.G., Srba, J.: Energy games in multiweighted automata. In: Cerone, A., Pihlajasaari, P. (eds.) ICTAC 2011. LNCS, vol. 6916, pp. 95–115. Springer, Heidelberg (2011)

21. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theor. Comput. Sci. 256(1-2), 63–92 (2001)
22. Greibach, S.A.: An infinite hierarchy of context-free languages. J. ACM 16(1), 91–106 (1969)
23. Higman, G.: Ordering by divisibility in abstract algebras. Proceedings of the London Mathematical Society 2, 236–366 (1952)
24. Hofman, P., Lasota, S., Mayr, R., Totzke, P.: Simulation over one-counter nets is PSPACE-complete. In: Seth, A., Vishnoi, N.K. (eds.) FSTTCS. LIPIcs, vol. 24, pp. 515–526. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2013)
25. Hofman, P., Totzke, P.: Trace inclusion for one-counter nets revisited. CoRR, abs/1404.5157 (2014)
26. Ibarra, O.H.: Restricted one-counter machines with undecidable universe problems. Mathematical Systems Theory 13, 181–186 (1979)
27. Jančar, P., Esparza, J., Moller, F.: Petri nets and regular processes. J. Comput. Syst. Sci. 59(3), 476–503 (1999)
28. Jančar, P., Kucera, A., Moller, F., Sawa, Z.: DP lower bounds for equivalence-checking and model-checking of one-counter automata. Inf. Comput. 188(1), 1–19 (2004)
29. Juhl, L., Guldstrand Larsen, K., Raskin, J.-F.: Optimal bounds for multiweighted and parametrised energy games. In: Liu, Z., Woodcock, J., Zhu, H. (eds.) He Festschrift. LNCS, vol. 8051, pp. 244–255. Springer, Heidelberg (2013)
30. Information Sciences Institute of the University of Southern California. Transmission Control Protocoll (DARPA Internet Program Protocol Specification) (1981), `http://www.faqs.org/rfcs/rfc793.html`
31. Ouaknine, J., Worrell, J.: On the language inclusion problem for timed automata: Closing a decidability gap. In: LICS, pp. 54–63. IEEE Computer Society (2004)
32. Ouaknine, J., Worrell, J.: On the decidability of metric temporal logic. In: LICS, pp. 188–197. IEEE Computer Society (2005)
33. Ouaknine, J., Worrell, J.B.: On metric temporal logic and faulty turing machines. In: Aceto, L., Ingólfsdóttir, A. (eds.) FOSSACS 2006. LNCS, vol. 3921, pp. 217–230. Springer, Heidelberg (2006)
34. Quaas, K.: On the interval-bound problem for weighted timed automata. In: Dediu, A.-H., Inenaga, S., Martín-Vide, C. (eds.) LATA 2011. LNCS, vol. 6638, pp. 452–464. Springer, Heidelberg (2011)
35. Quaas, K.: Model checking metric temporal logic over automata with one counter. In: Dediu, A.-H., Martín-Vide, C., Truthe, B. (eds.) LATA 2013. LNCS, vol. 7810, pp. 468–479. Springer, Heidelberg (2013)