Model checking freeze LTL over one-counter automata*

Stéphane Demri[1], Ranko Lazić[3], and Arnaud Sangnier[1,2]

[1]LSV, ENS Cachan, CNRS, INRIA & [2]EDF R&D
[3]Department of Computer Science, University of Warwick, UK

Abstract. We study complexity issues related to the model-checking problem for LTL with registers (a.k.a. freeze LTL) over one-counter automata. We consider several classes of one-counter automata (mainly deterministic vs. nondeterministic) and several syntactic fragments (restriction on the number of registers and on the use of propositional variables for control locations). The logic has the ability to store a counter value and to test it later against the current counter value. By introducing a non-trivial abstraction on counter values, we show that model checking LTL with registers over deterministic one-counter automata is PSPACE-complete with infinite accepting runs. By constrast, we prove that model checking LTL with registers over nondeterministic one-counter automata is Σ_1^1-complete [resp. Σ_1^0-complete] in the infinitary [resp. finitary] case even if only one register is used and with no propositional variable. This makes a difference with the facts that several verification problems for one-counter automata are known to be decidable with relatively low complexity, and that finitary satisfiability for LTL with a unique register is decidable. Our results pave the way for model-checking LTL with registers over other classes of operational models, such as reversal-bounded counter machines and deterministic pushdown systems.

1 Introduction

Logics for data words and trees. Data words are sequences in which each position is labelled by a letter from a finite alphabet and by another letter from an infinite alphabet (the datum). This fundamental and simple model captures the timed words accepted by timed automata [1], and its extension to trees is useful to model XML documents with values, see e.g. [4,15]. In order to really speak about data, known logical formalisms for data words/trees contain a mechanism that stores a value and tests it later against other values, see e.g. [5,9]. This is a powerful feature shared by other memoryful temporal logics [18,16]. However, the satisfiability problem for these logics becomes easily undecidable even when stored data can be tested only for equality. For instance, first-order logic for data words restricted to three individual variables is undecidable [5], whereas LTL with registers (also known as freeze LTL) restricted to a single register is undecidable over infinite data words [9]. By contrast, decidable fragments of the satisfiability problems have been found in [5,10,19] either by imposing syntactic restrictions (bound the number of registers, constrain the polarity of temporal formulae, etc.) or by considering subclasses of data words (finiteness for example). Similar phenomena occur with metric temporal logics and timed words [22,23]. A key point for all these

* Partially supported by project AVERISS (ANR).

logical formalisms is the ability to store a value from an infinite alphabet, which is a feature also present in models of register automata, see e.g [7,21,25]. However, the storing mechanism has a long tradition (apart from its ubiquity in programming languages) since it appeared for instance in real-time logics [2] (the data are time values) and in so-called hybrid logics (the data are node addresses), see an early undecidability result with reference pointers in [13]. Meaningful restrictions for hybrid logics can also lead to decidable fragments, see e.g. [24].

Our motivations. In this paper, our main motivation is to analyze the effects of adding a binding mechanism with registers to specify runs of operational models such as pushdown systems and counter automata. The registers are simple means to compare data values at different points of the execution. Indeed, runs can be naturally viewed as data words: for example, the finite alphabet is the set of locations and the infinite alphabet is the set of data values (natural numbers, stacks, etc.). To do so, we enrich an ubiquitous logical formalism for model-checking techniques, namely linear-time temporal logic LTL, with registers. Even though this was the initial motivation to introduce LTL with registers in [10], most decision problems considered in [10,19,9] are essentially oriented towards satisfiability. In this paper, we focus on the following type of model-checking problem: given a set of runs generated by an operational model, more precisely by a one-counter automaton, and a formula from LTL with registers, is there a run satisfying the given formula? In our context, it will become clear that the extension with two counters is undecidable. It is not difficult to show that this model-checking problem differs from those considered in [19,10] and are of a different nature from those for hybrid logics investigated in [12,27]. However, since two consecutive counter values in a run are ruled by the set of transitions, constraints on data that are helpful to get fine-tuned undecidability proofs for satisfiability problems in [10,9] may not be allowed on runs. This is precisely what we want to understand in this work. Like in [6], LTL with registers makes sense to specify and reason about configurations of operational models, precisely counter systems.

Our contribution. We study complexity issues related to the model-checking problem for LTL with registers over one-counter automata that are simple operational models but our undecidability results can be obviously lifted to pushdown systems when registers store the stack value. Moreover, in order to determine borderlines for decidability, we also present results for deterministic one-counter models that are less powerful but remain interesting when they are viewed as a means to specify an infinite path on which model checking is performed, see analogous issues in [20].

We consider several classes of one-counter automata (deterministic and nondeterministic) and several fragments by restricting the use of registers or the use of letters from the finite alphabet. Moreover, we distinguish finite accepting runs from infinite ones as data words. Unlike several results from [22,23,9,19], the decidability status of the model checking does not depend on the fact that we consider finite data words instead of infinite ones. In this paper, we present the following results.

– Model checking LTL with registers over deterministic one-counter automata is PSPACE-complete (see Sect. 3.2). PSPACE-hardness is established by reducing

QBF and it also holds when no letters from the finite alphabet are used in formulae. When the number of registers is bounded, the problem can be solved in polynomial time. In order to get these complexity upper bounds, we introduce an abstraction on counter values even though the counter values may not be bounded along the unique run of the deterministic automata. This makes a substantial difference with [20] in which no data values are considered, but still our problem amounts to model checking a path specified by a deterministic one-counter automaton.

- Model checking LTL with registers over nondeterministic one-counter automata restricted to a unique register and without alphabet is Σ_1^1-complete in the infinitary case by reducing the recurrence problem for Minsky machines (see Sect. 4). In the finitary case, the problem is shown Σ_1^0-complete by reducing the halting problem for Minsky machines. These results are quite surprising since several verification problems for one-counter automata are known to be decidable with relatively low complexity [14,26]. Moreover, finitary satisfiability for LTL with one register is decidable [9] even though with nonprimitive complexity.

Because of lack of space, omitted proofs can be found in [11].

2 Preliminaries

2.1 One-counter automaton

Let us recall standard definitions and notations about our operational models. A one-counter automaton is a tuple $\mathcal{A} = \langle Q, q_I, \delta, F \rangle$ where Q is a finite set of locations, $q_I \in Q$ is the initial location, $F \subseteq Q$ is the set of accepting locations and $\delta \subseteq Q \times L \times Q$ is the transition relation over the instruction set $L = \{\mathtt{inc}, \mathtt{dec}, \mathtt{ifzero}\}$. A counter valuation v is an element of $\mathbb{N}$ and a configuration of $\mathcal{A}$ is a pair in $Q \times \mathbb{N}$. The initial configuration is the pair $\langle q_I, 0 \rangle$. As usual, a one-counter automaton $\mathcal{A}$ induces a (possibly infinite) transition system $\langle Q \times \mathbb{N}, \rightarrow \rangle$ such that $\langle q, n \rangle \rightarrow \langle q', n' \rangle$ iff one of the conditions below holds true: (1) $\langle q, \mathtt{inc}, q' \rangle \in \delta$ and $n' = n+1$, (2) $\langle q, \mathtt{dec}, q' \rangle \in \delta$ and $n' = n - 1$ (and $n' \in \mathbb{N}$), (3) $\langle q, \mathtt{ifzero}, q' \rangle \in \delta$ and $n = n' = 0$. A finite [resp. infinite] *run* ρ is a finite [resp. infinite] sequence $\rho = \langle q_0, n_0 \rangle \rightarrow \langle q_1, n_1 \rangle \rightarrow \cdots$ where $\langle q_0, n_0 \rangle$ is the initial configuration. A finite run is *accepting* iff it ends with an accepting location. An infinite run ρ is accepting iff it contains an accepting location infinitely often (Büchi acceptance condition).

A one-counter automaton $\mathcal{A}$ is *deterministic* whenever it corresponds to a deterministic one-counter Minsky machine: for every location q, either $\mathcal{A}$ has a unique transition from q incrementing the counter, or $\mathcal{A}$ has exactly two transitions from q, one with instruction $\mathtt{ifzero}$ and the other one with instruction $\mathtt{dec}$, or $\mathcal{A}$ has no transition from q (not present in original deterministic Minsky machines). In the transition system induced by any deterministic one-counter automaton, each configuration has at most one successor. One-counter automata in full generality are understood as *nondeterministic* one-counter automata.

2.2 LTL over data words

Formulae of the logic $\mathrm{LTL}^{\downarrow,\Sigma}$ where Σ is a finite alphabet are defined as follows:

$$\phi ::= a \mid \uparrow_r \mid \neg\phi \mid \phi \wedge \phi \mid \phi \mathtt{U} \phi \mid \mathtt{X}\phi \mid \downarrow_r \phi$$

where $a \in \Sigma$ and r ranges over $\mathbb{N} \setminus \{0\}$. We write $\mathrm{LTL}^{\downarrow}$ to denote LTL with registers for some unspecified finite alphabet. An occurrence of $\uparrow_r$ within the scope of some freeze quantifier $\downarrow_r$ is bound by it; otherwise it is free. A sentence is a formula with no free occurrence of any $\uparrow_r$. Given a natural number $n > 0$, we write $\mathrm{LTL}_n^{\downarrow,\Sigma}$ to denote the restriction of $\mathrm{LTL}^{\downarrow,\Sigma}$ to registers in $\{1, \ldots, n\}$. Models of $\mathrm{LTL}^{\downarrow,\Sigma}$ are *data words*. A data word σ over a finite alphabet Σ is a non-empty word in $\Sigma^{<\omega}$ or Σ^{ω}, together with an equivalence relation $\sim^{\sigma}$ on word indices. We write $|\sigma|$ for the length of the data word, $\sigma(i)$ for its letters where $0 \leq i < |\sigma|$.

A *register valuation* v for a data word σ is a finite partial map from $\mathbb{N} \setminus \{0\}$ to the indices of σ. Whenever $v(r)$ is undefined, the formula $\uparrow_r$ is interpreted as false. The satisfaction relation $\models$ is defined as follows (Boolean clauses are omitted).

$$\begin{aligned}
\sigma, i \models_v a &\overset{\text{def}}{\Leftrightarrow} \sigma(i) = a \\
\sigma, i \models_v \uparrow_r &\overset{\text{def}}{\Leftrightarrow} r \in \mathrm{dom}(v) \text{ and } v(r) \sim^{\sigma} i \\
\sigma, i \models_v \mathtt{X}\phi &\overset{\text{def}}{\Leftrightarrow} i + 1 < |\sigma| \text{ and } \sigma, i+1 \models_v \phi \\
\sigma, i \models_v \phi_1 \mathtt{U} \phi_2 &\overset{\text{def}}{\Leftrightarrow} \text{ for some } j \geq i,\ \sigma, j \models_v \phi_2 \text{ and for all } i \leq j' < j,\ \sigma, j' \models_v \phi_2 \\
\sigma, i \models_v \downarrow_r \phi &\overset{\text{def}}{\Leftrightarrow} \sigma, i \models_{v[r \mapsto i]} \phi
\end{aligned}$$

$v[r \mapsto i]$ denotes the register valuation equal to v except that the register r is mapped to the position i. In the sequel, we omit the subscript "v" in $\models_v$ when sentences are involved. We use the standard abbreviations for the temporal operators ($\mathtt{G}$, $\mathtt{F}$, ...) and for the Boolean operators and constants ($\vee$,$\Rightarrow$, $\top$, $\perp$, ...). The infinitary [resp. finitary] satisfiability problem for LTL with registers, noted ω-SAT-$\mathrm{LTL}^{\downarrow}$ [resp. f-SAT-$\mathrm{LTL}^{\downarrow}$], is defined as follows: given a finite alphabet Σ and a formula ϕ in $\mathrm{LTL}^{\downarrow,\Sigma}$, is there an infinite [resp. a finite] data word σ such that $\sigma, 0 \models \phi$?

Theorem 1. *[9] ω-SAT-$\mathrm{LTL}^{\downarrow}$ restricted to one register is Π_1^0-complete and f-SAT-$\mathrm{LTL}^{\downarrow}$ restricted to one register is decidable with non-primitive recursive complexity.*

Given a one-counter automaton $\mathcal{A} = \langle Q, q_I, \delta, F \rangle$, finite [resp. infinite] accepting runs of $\mathcal{A}$ can be viewed as finite [resp. infinite] data words over the alphabet Q. Indeed, given a run ρ, the equivalence relation $\sim^{\rho}$ is defined as follows: $i \sim^{\rho} j$ iff the counter value at the ith position of ρ is equal to the counter value at the jth position of ρ. In order to ease the presentation, in the sequel we store in registers counter values, which is an equivalent way to proceed by slightly adapting the semantics for $\uparrow_r$ and $\downarrow_r$, and the values stored in registers (data).

The finitary [resp. infinitary] (existential) model-checking problem over one-counter automata for LTL with registers, noted $\mathrm{MC}^{<\omega}$ [resp. MC^{ω}] is defined as follows: given a one-counter automaton $\mathcal{A} = \langle Q, q_I, \delta, F \rangle$ and a sentence ϕ in $\mathrm{LTL}^{\downarrow,Q}$, is there a finite [resp. infinite] accepting run ρ of $\mathcal{A}$ such that $\rho, 0 \models \phi$? If the answer is "yes", we write

$\mathcal{A} \models^{<\omega} \phi$ [resp. $\mathcal{A} \models^{\omega} \phi$]. In this existential version of model checking, this problem can be viewed as a variant of satisfiability in which satisfaction of a formula can be only witnessed within a specific class of data words, namely the accepting runs of the automata. Results for the universal version of model checking will follow easily from those for the existential version.

We write MC_n^α to denote the restriction of MC^α to formulae with at most n registers. Very often, it makes sense that only counter values are known but not the current location of a configuration, which can be understood as an internal information about the system. We write PureMC_n^α to denote the restriction of MC_n^α (its "pure" version) to formulae with atomic formulae only of the form $\uparrow_r$.

Example 1. Here are properties that can be stated in $\mathrm{LTL}_2^{\downarrow,Q}$ along a run.

- "There is a suffix such that all the counter values are different": $\mathtt{FG}(\downarrow_1 \mathtt{XG}\neg \uparrow_1)$.
- "Whenever location q is reached with current counter value n and next current counter value m, if there is a next occurrence of q, the two consecutive counter values are also n and m": $\mathtt{G}(q \Rightarrow \downarrow_1 \mathtt{X} \downarrow_2 \mathtt{XG}(q \Rightarrow \uparrow_1 \wedge \mathtt{X} \uparrow_2))$.

We show how to get rid of propositional variables by reducing the model-checking problem over one-counter automata to its pure version.

Lemma 2 **(Purification).** *Given a one-counter automaton $\mathcal{A}$ and a sentence ϕ in $\mathrm{LTL}_n^{\downarrow,Q}$, one can compute in logarithmic space in $|\mathcal{A}| + |\phi|$ a one-counter automaton $\mathcal{A}_P$ and ϕ_P in $\mathrm{LTL}_{max(n,1)}^{\downarrow,\emptyset}$ such that $\mathcal{A} \models^{<\omega} \phi$ [resp. $\mathcal{A} \models^{\omega} \phi$] iff $\mathcal{A}_P \models^{<\omega} \phi_P$ [resp. $\mathcal{A}_P \models^{\omega} \phi_P$]. Moreover, $\mathcal{A}$ is deterministic iff $\mathcal{A}_P$ is deterministic.*

The idea of the proof is simply to identify locations with patterns about the changes of the unique counter that can be expressed in $\mathrm{LTL}^{\downarrow,\emptyset}$.

Proof. Let $\mathcal{A} = \langle Q, q_I, \delta, F\rangle$ with $Q = \{q_1, ..., q_n\}$ and ϕ in $\mathrm{LTL}^{\downarrow,Q}$. In order to define $\mathcal{A}_P$, we identify locations with patterns about the changes of the unique counter. For each location q_i in Q we associate the new sequence of transitions described in Fig. 1 and $q_i \xrightarrow{\mathtt{a}} q_j \in \delta$ iff $q_i^F \xrightarrow{\mathtt{a}} q_j \in \delta'$. In the sequence of picks numbered from 0 to $n+1$, the only pick of height 2 is one numbered i. In order to identify the beginning of the first pick of height 3 we introduce formulae in $\mathrm{LTL}_1^{\downarrow,\emptyset}$: $\varphi_{\neg\frac{3}{7}}$ expresses that "among the 7 next counter values (including the current counter value), there are no 3 equal values" and $\varphi_{0\sim 6}$ expresses that "the current counter value is equal to the counter value at the 6th next position". We write LOC to denote the formula $\varphi_{\neg\frac{3}{7}} \wedge \varphi_{0\sim 6}$. By a simple case analysis, one can check that for $k \geq 0$, in the run of $\mathcal{A}_P$, LOC holds true iff the current location is in Q. We pose $\phi_i = \mathtt{X}^{6+2(i-1)} \downarrow_1 \mathtt{X}^2\neg \uparrow_1$ for $1 \leq i \leq n$. One can check that for $k \geq 0$, in the run of $\mathcal{A}_P$ LOC $\wedge\, \phi_i$ holds true iff the current location is q_i. ϕ_P is equal to $\mathrm{T}(\phi)$ with the map T that is homomorphic for Boolean operators and $\downarrow_r$, and its restriction to $\uparrow_r$ is identity. The rest of the inductive definition is as follows.

$$\mathrm{T}(q_i) = \phi_i;\ \mathrm{T}(\mathtt{X}\phi) = \mathtt{X}^{10+2(n+1)+1}\mathrm{T}(\phi);\ \mathrm{T}(\phi\mathtt{U}\phi') = \big(\mathrm{LOC} \Rightarrow \mathrm{T}(\phi)\big)\mathtt{U}\big(\mathrm{LOC} \wedge \mathrm{T}(\phi')\big)$$

We remark that ϕ and ϕ_P have the same amount of registers unless ϕ has no register. $\square$

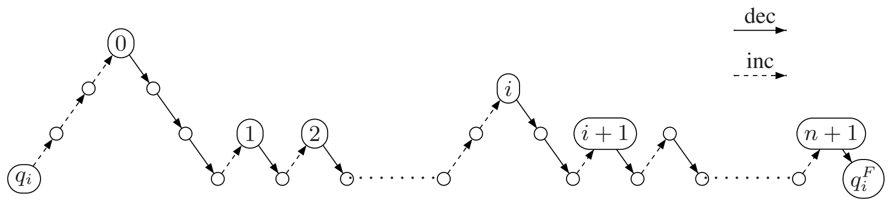


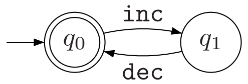
Fig. 1. Encoding q_i by a pattern made of $n + 2$ increasing picks of length $10 + 2(n + 1)$

3 Model checking deterministic one-counter automata

In this section, we show that MC^{ω} restricted to deterministic one-counter automata is PSPACE-complete and the same restriction for $\mathrm{MC}^{<\omega}$ is in EXPSPACE. First, we show PSPACE-hardness.

Proposition 3. *$\mathrm{PureMC}^{<\omega}$ and PureMC^{ω} restricted to deterministic one-counter automata are PSPACE-hard problems.*

Proof. Consider a QBF instance $\phi = \forall \mathrm{p}_1 \exists \mathrm{p}_2 \cdots \forall \mathrm{p}_{2N-1} \exists \mathrm{p}_{2N}\ \Psi(\mathrm{p}_1, ..., \mathrm{p}_{2N})$ where p_1,...,p_{2N} are propositional variables and $\Psi(\mathrm{p}_1, \ldots, \mathrm{p}_{2N})$ is a quantifier-free propositional formula built over $\mathrm{p}_1, \ldots, \mathrm{p}_{2N}$. The fixed deterministic one-counter automaton $\mathcal{A}$ below generates the sequence of counter values $(01)^{\omega}$.



Let ψ be the formula in $\mathrm{LTL}^{\downarrow,\emptyset}$ defined from the family $\psi_1, \ldots, \psi_{2N+1}$ of formulae with $\psi =\downarrow_{2N+1} \psi_1$: $\psi_{2N+1} = \Psi[\mathrm{p}_i \leftarrow (\uparrow_i \Leftrightarrow \uparrow_{2N+1})]$ and for $i \in \{1, ..., N\}$, $\psi_{2i} = \mathtt{F}(\downarrow_{2i} \psi_{2i+1})$ and $\psi_{2i-1} = \mathtt{G}(\downarrow_{2i-1} \psi_{2i})$. One can show that ϕ is satisfiable iff $\mathcal{A}_\phi \models_\omega \psi$. For $\mathrm{PureMC}^{<\omega}$, one can enforce the sequence of counter values from the accepting run to be $(01)^{2N}0$ and then use $\mathtt{X}$ to define the ψ_is. $\square$

Observe that in the reduction, we use an unbounded number of registers (see Theorem 12) but a fixed deterministic one-counter automaton.

3.1 Properties on runs for deterministic automata

Any deterministic one-counter automaton $\mathcal{A}$ has at most one infinite run, possibly with an infinite amount of counter values. If this run is not accepting, i.e. no accepting location is repeated infinitely, then for no formula ϕ, we have $\mathcal{A} \models^{\omega} \phi$. We show below that we can decide in polynomial-time whether $\mathcal{A}$ has accepting runs either finite or infinite. Moreover, we shall show that the infinite unique run has some regularity.

Let $\rho^{\omega}_{\mathcal{A}}$ be the unique run (if it exists) of the deterministic one-counter automaton $\mathcal{A}$ represented by the following sequence of configurations $\langle q_0, n_0\rangle\ \langle q_1, n_1\rangle\ \langle q_2, n_2\rangle \ldots$

Lemma 4. *Let $\mathcal{A}$ be a deterministic one-counter automaton with an infinite run. There are K_1, K_2, K_3 such that $K_1 + K_2 \leq |Q|^3$, $K_3 \leq |Q|$ and for every $i \geq K_1$, $\langle q_{i+K_2}, n_{i+K_2}\rangle = \langle q_i, n_i + K_3\rangle$.*

Hence, the run $\rho^\omega_{\mathcal{A}}$ can be encoded by its first K_1+K_2 configurations. $\rho^\omega_{\mathcal{A}}$ has a simple structure: it is composed of a polynomial-size prefix $\langle q_0, n_0\rangle \cdots \langle q_{K_1-1}, n_{K_1-1}\rangle$ followed by the polynomial-size loop $\langle q_{K_1}, n_{K_1}\rangle \cdots \langle q_{K_1+K_2-1}, n_{K_1+K_2-1}\rangle$ repeated infinitely often. The effect of applying the loop consists in adding K_3 to every counter value. Testing whether $\mathcal{A}$ has an infinite run or $\rho^\omega_{\mathcal{A}}$ is accepting amounts to check whether there is an accepting location in the loop, which can be done in cubic time in $|Q|$. In the rest of this section, we assume that $\rho^\omega_{\mathcal{A}}$ is accepting. Similarly, testing whether $\mathcal{A}$ has a finite accepting run amounts to check whether an accepting location occurs in the prefix or in the loop.

When $K_3 = 0$ and $\mathcal{A}$ has an infinite run, $\rho^\omega_{\mathcal{A}}$ is precisely

$$\langle q_0, n_0\rangle \cdots \langle q_{K_1-1}, n_{K_1-1}\rangle(\langle q_{K_1}, n_{K_1}\rangle \cdots \langle q_{K_1+K_2-1}, n_{K_1+K2-1}\rangle)^\omega.$$

It is then possible to apply a polynomial-space labelling algorithm à la CTL for model checking $\mathrm{LTL}^{\downarrow,Q}$ formulae on $\mathcal{A}$. However, one needs to take care of register valuations, which explains why unlike the polynomial-time algorithm for model checking ultimately periodic models on LTL formulae (see e.g., [20]), model checking restricted to deterministic automata with $K_3 = 0$ is still PSPACE-hard.

3.2 A PSPACE symbolic model-checking algorithm

In this section, we provide decision procedures for solving $\mathrm{MC}^{<\omega}$ and MC^{ω} restricted to deterministic one-counter automata. Let us introduce some notations. Let $\rho^\omega_{\mathcal{A}} = \langle q_0, n_0\rangle \langle q_1, n_1\rangle \langle q_2, n_2\rangle \ldots$ be the unique run of the deterministic one-counter automaton $\mathcal{A}$ and ϕ be a sentence with $N \geq 1$ registers. Let $i \geq 0$ be a position in $\rho^\omega_{\mathcal{A}}$ and m be a register value in $\mathbb{N}$. We write $\mathrm{pos}_{\mathcal{A}}(i, m)$ to denote the following (possibly infinite) set of offsets: $\mathrm{pos}_{\mathcal{A}}(i, m) = \{j \in \mathbb{N} : m = n_{i+j}\}$. The values m should be understood as register values when evaluation of subformulae is done at position i. In general, the set $\{\mathrm{pos}_{\mathcal{A}}(i, m) \subseteq \mathbb{N} : i, m \in \mathbb{N}\}$ can be infinite but if we restrict ourselves to m in $\{n_0, \ldots, n_i\}$ then it is not anymore the case. After all, this is a reasonable assumption when m is intended to be a value stored in a register. Before showing this property, we establish that whenever $K_3 > 0$, two positions with identical counter values are separated by a distance that is bounded by a polynomial in $|Q|$.

Lemma 5. *Suppose $K_3 > 0$. For all $i \leq j$, (I) $n_i = n_j$ and $i < K_1$ imply $(j - i) \leq K_1 + K_1K_2$, (II) $n_i = n_j$ and $i \geq K_1$ imply $(j - i) \leq K_2^2$.*

Lemma 6. *$\{\mathrm{pos}_{\mathcal{A}}(i, m) : i \in \mathbb{N},\ m \in \{n_0, \ldots, n_i\}\}$ is finite and its cardinality is polynomial in $|Q|$.*

We write $\mathrm{REGVALUES}_{\mathcal{A}}$ to denote the above finite set with polynomial cardinality. Observe that even though the set of counter values occurring in $\rho^\omega_{\mathcal{A}}$ may be infinite (exactly when $K_3 > 0$) we can represent symbolically each register value $v(r)$ at a position i by a concise representation for $\mathrm{pos}_{\mathcal{A}}(i, v(r))$. One consequence of the proof of Lemma 6 is that $|\mathrm{REGVALUES}_{\mathcal{A}}|$ is bounded by $(1+K_1+K_2^2)^2+K_2\times(1+K_1+K_2^2)$.

We define below the equivalence relation $\equiv$ between positions of $\rho^{\omega}_{\mathcal{A}}$: $i \equiv i'$ iff $q_i = q'_i$, and for all $\alpha, \beta \geq 0$, ($n_{i+\alpha} = n_{i+\beta}$ iff $n_{i'+\alpha} = n_{i'+\beta}$) and ($q_{i+\alpha} = q_{i+\beta}$ iff $q_{i'+\alpha} = q_{i'+\beta}$). Typically, i and i' are equivalent whenever the path starting at position i is isomorphic to the path starting at position i'. It is easy to see that $\equiv$ has at most $K_1 + K_2$ equivalence classes since $i \equiv_{K_2} i'$ and $i, i' \geq K_1$ imply $i \equiv i'$ (here $\equiv_{K_2}$ is the congruence relation). We extend $\equiv$ to pairs composed of positions and register valuations. Given positions $i, i' \in \mathbb{N}$ and register valuations v, v' such that $\mathrm{ran}(v) \subseteq \{n_0, \ldots, n_i\}$ and $\mathrm{ran}(v') \subseteq \{n_0, \ldots, n_{i'}\}$, $\langle i, v\rangle \equiv \langle i', v'\rangle$ iff (1) $i \equiv i'$ and (2) for all $\alpha \geq 0$ and registers $r \in \{1, \ldots, N\}$, $n_{i+\alpha} = v(r)$ iff $n_{i'+\alpha} = v'(r)$. Again, $\equiv$ is an equivalence relation. A pair $\langle i, v\rangle$ is called a *context*.

Condition (2) on the definition of $\equiv$ is equivalent to: for every register $r \in \{1, \ldots, N\}$, $\mathrm{pos}_{\mathcal{A}}(i, v(r)) = \mathrm{pos}_{\mathcal{A}}(i', v'(r))$. Consequently,

Lemma 7. *There are polynomials P_1 and P_2 such that the number of equivalence classes for $\equiv$ on contexts $\langle i, v\rangle$ is bounded by $P_1(|Q|) \times [P_2(|Q|)]^N$ (N is the number of registers).*

The bound is exactly $(K_1 + K_2) \times [(1 + K_1 + K_2^2)^2 + K_2 \times (K_2^2 + K_1 + 1)]^N$, where $(1 + K_1 + K_2^2)^2 + K_2 \times (K_2^2 + K_1 + 1)$ is the cardinal of $\mathrm{REGVALUES}_{\mathcal{A}}$ and $K_1 + K_2$ is the number of equivalent positions w.r.t. to $\equiv$.

Lemma 8. *If $\langle i, v\rangle \equiv \langle i', v'\rangle$, then (I) for all $j > 0$, $\langle i + j, v\rangle \equiv \langle i' + j, v'\rangle$ and (II) for every formula $\psi \in \mathrm{LTL}^{\downarrow, Q}_N$, $\rho^{\omega}_{\mathcal{A}}, i \models_v \psi$ iff $\rho^{\omega}_{\mathcal{A}}, i' \models_{v'} \psi$.*

Lemma 8(I) is by an easy verification (recurrence on j) whereas Lemma 8(II) is by structural induction on ψ.

3.3 Abstraction and complexity issues

We have seen that the equivalence relation $\equiv$ on contexts has finite index. We present below a means to represent symbolically an equivalence class. In the case $K_3 = 0$, a *symbolic context* is a pair $\langle i, pos\rangle$ where $i \in \{0, \ldots, K_1 + K_2 - 1\}$ and *pos* is a symbolic register valuation of the form $\{1, \ldots, N\} \to \{n_0, \ldots, n_{K_1+K_2-1}\}$. A context $\langle i, v\rangle$ is represented by the symbolic context $\langle i', pos\rangle$ where

- $i < K_1$ implies $i' = i$ otherwise i' is the unique element of $\{K_1, \ldots, K_1 + K_2 - 1\}$ such that $i \equiv_{K_2} i'$,
- for $r \in \{1, \ldots, N\}$, $pos(r) = v(r)$. Observe that $v(r) \in \{n_0, \ldots, n_{K_1+K_2-1}\}$ and $pos(r)$ can be encoded with $\mathcal{O}(log(|Q|))$ bits.

When $K_3 > 0$, the definition of a symbolic context is modified for the second component only since the set of counter values along the run is infinite. A *symbolic context* remains a pair $\langle i, pos\rangle$ but $i \in \{0, \ldots, K_1 + K_2 - 1\}$ and *pos* is a symbolic register valuation of the form $\{1, \ldots, N\} \to \mathcal{P}(\{0, \ldots, K_1 + K_1 K_2\}) \cup \mathcal{P}(\{0, \ldots, K_2^2\})$. Moreover, when $i < K_1$, $pos(r) \subseteq \{0, \ldots, K_1 + K_1 K_2\}$, otherwise $pos(r) \subseteq \{0, \ldots, K_2^2\}$. Indeed, from Lemma 5, whenever $K_3 > 0$, for all $i \in \mathbb{N}$ and $m \in \{n_0, \ldots, n_i\}$, if $i < K_1$, then $\mathrm{pos}_{\mathcal{A}}(i,m) \subseteq \{0,\ldots,K_1{+}K_1K_2\}$ otherwise $\mathrm{pos}_{\mathcal{A}}(i,m) \subseteq \{0,\ldots,K_2^2\}$. A context $\langle i,v\rangle$ is represented by the symbolic context $\langle i', pos\rangle$ where i' is defined as above and for $r \in \{1,\ldots,N\}$, $pos(r) = \mathrm{pos}_{\mathcal{A}}(i,v(r))$.

Each value $pos(r)$ can be encoded with a polynomial amount of bits in $|Q|$. One can compute in polynomial time in $|Q|$ the range of any symbolic register valuation (whether $K_3 = 0$ or not) thanks to Lemma 6. When we bound the number of registers, the number of symbolic contexts occurring in $\rho_{\mathcal{A}}^{\omega}$ is polynomial in $|Q|$ and they can be computed in polynomial time.

Given a context $\langle i,v\rangle$, we write $[\langle i,v\rangle]$ to denote its corresponding symbolic context (w.r.t. $\mathcal{A}$). Symbolic contexts correspond to the equivalence classes of $\equiv$:

Lemma 9. *Let $\langle i,v\rangle$ and $\langle i',v'\rangle$ be contexts. Then $[\langle i,v\rangle] = [\langle i',v'\rangle]$ iff $\langle i,v\rangle \equiv \langle i',v'\rangle$.*

Let us define a map $next$ that takes as argument a symbolic context $\langle i, pos\rangle$ and returns the symbolic context obtained at the next step. This is a well-defined function because taking two contexts that are $\equiv$-equivalent, moving one step forward leads to two new contexts that are also $\equiv$-equivalent (see Lemma 10 below). The map $next$ is defined as follows : $next(\langle i,pos\rangle) = \langle i', pos'\rangle$ where

- if $i < K_1 + K_2 - 1$ then $i' = i+1$, otherwise $i' = K_1$.
- if $K_3 > 0$, then for $r \in \{1,\ldots,N\}$, $pos'(r) = \{\alpha - 1 : \alpha \in pos(r), \alpha > 0\}$,
- if $K_3 = 0$, then $pos' = pos$.

Lemma 10. *Let $\langle i,v\rangle$ be a context with $\mathrm{ran}(v) \subseteq \{n_0,\ldots,n_i\}$. Then $next([\langle i,v\rangle]) = [\langle i+1,v\rangle]$.*

Below, we solve the model-checking problem by following an automata-based approach [30]. We consider alternating word automata with Büchi acceptance condition on ω-words, see e.g. [29]: every infinite branch of accepting runs has an accepting state repeated infinitely often. Let ϕ be a formula in NNF built over disjunction $\vee$ and the release operator $\mathtt{R}$ (dual of $\mathtt{U}$). Observe that $\mathtt{X}$ and $\downarrow_r$ are self-dual. We build an alternating automaton $\mathcal{A}_\phi$ that can be viewed as the product between the run of $\mathcal{A}$ and the automaton for ϕ. The synchronization mode between these two components takes into account the presence of registers. When $K_3 > 0$ and $\mathcal{A}$ has an accepting run (which can be checked in PTIME), let $\mathcal{A}_\phi = \langle \Sigma, S, s_0, \delta, F\rangle$ be defined as follows:

- $\Sigma = \{a\}$ and S is the set of states of the form $\langle\langle i,pos\rangle,\psi\rangle$ where $\langle i,pos\rangle$ is a symbolic context and ψ is a subformula of ϕ.
- the initial state is $s_0 = \langle\langle 0, pos_0\rangle, \phi\rangle$ where pos_0 is the symbolic register valuation representing the zero register valuation and F is the set of accepting states whose outermost connective of the second component is not until.
- Here is the transition function δ (obvious dual clauses are omitted):
  - $\delta(\langle\langle i,pos\rangle,q\rangle,a) = \top$ if $q = q_i$, otherwise $\delta(\langle\langle i,pos\rangle,q\rangle,a) = \perp$,
  - $\delta(\langle\langle i,pos\rangle,\neg\uparrow_r\rangle,a) = \perp$ if $0 \in pos(r)$, otherwise $\delta(\langle\langle i,pos\rangle,\neg\uparrow_r\rangle,a) = \top$,
  - $\delta(\langle\langle i,pos\rangle,\psi\wedge\psi'\rangle,a) = \delta(\langle\langle i,pos\rangle,\psi\rangle,a) \wedge \delta(\langle\langle i,pos\rangle,\psi'\rangle,a)$,
  - $\delta(\langle\langle i,pos\rangle,\mathtt{X}\psi\rangle,a) = \langle next(\langle i,pos\rangle),\psi\rangle$,
  - $\delta(\langle\langle i,pos\rangle,\downarrow_r \psi\rangle,a) = \delta(\langle\langle i,pos[r \leftarrow \mathrm{pos}_{\mathcal{A}}(i,n_i)]\rangle,\psi\rangle,a)$,

- $\delta(\langle\langle i, pos\rangle, \psi \mathtt{U} \psi'\rangle, a) =$
  $\delta(\langle\langle i, pos\rangle, \psi'\rangle, a) \vee (\delta(\langle\langle i, pos\rangle, \psi\rangle, a) \wedge \langle next(\langle i, pos\rangle), \psi \mathtt{U} \psi'\rangle)$.

When $K_3 = 0$, the clauses for $\downarrow_r$ and $\uparrow_r$ can be easily adapted. We write $\mathcal{A}_\phi^{\langle\langle i,pos\rangle,\psi\rangle}$ to denote the automaton defined from $\mathcal{A}_\phi$ with initial location $\langle\langle i, pos\rangle, \psi\rangle$.

Lemma 11. *Let $i \in \mathbb{N}$ and v be a register valuation with range $\{n_0, \ldots, n_i\}$. For every subformula ψ of ϕ, $\rho^\omega_\mathcal{A}, i \models_v \psi$ iff $\mathcal{A}_\phi^{\langle[\langle i,v\rangle],\psi\rangle}$ accepts an infinite run.*

The proof (by structural induction) is a variant of the one for LTL and uses Lemma 8 and 10. This will allow us to characterize precisely the complexity of model checking.

Theorem 12. MC^ω *restricted to deterministic one-counter automata is* PSPACE*-complete and its restriction to $n \geq 1$ registers is in* PTIME.

Proof. $\mathcal{A}_\phi$ is an hesitant alternating word automata over a 1-letter alphabet with each set S_j of the partition being a set of states with identical subformulae. By [17, Theorem 5.6], the nonemptiness problem for hesitant alternating word automata over a 1-letter alphabet can be solved in space $\mathcal{O}(m\ log^2 n)$ where n is the number of states and m is the number of elements in the partition of the set of states. In order to obtain the PSPACE upper bound, it is sufficient to check that the on-the-fly version of the algorithm given in the proof of [17, Theorem 5.6] can be performed (computation of the transition function on demand). This is possible partly because in $\mathcal{A}_\phi$, m is linear in $|\phi|$, n is exponential in $|\phi|$, for each state s, $\delta(s, a)$ can be built in polynomial-time in $|\phi|$ and testing if a state is accepting can be done in linear time in $|\phi|$. Moreover, each state in $\mathcal{A}_\phi$ can be encoded in polynomial space in $|\mathcal{A}| + |\phi|$.

When the number of registers is fixed, $\mathcal{A}_\phi$ has a polynomial number of states and since the nonemptiness problem for weak alternating word automata over a 1-letter alphabet can be solved in linear time [3], we get the PTIME upper bound. □

For the finitary case, we cannot invoke the result in [3] because the length of the word is a distinguishing factor.

Corollary 13. $\mathrm{MC}^{<\omega}$ *restricted to deterministic one-counter automata is in* EXPSPACE.

The proof consists in designing an alternating word automata on ω-words with a two-letter alphabet on the lines of the previous construction. However, the second letter marks the end of the word so that all the branches detect the end of the word in a synchronous way. The recognized ω-words are among $a^* \cdot b \cdot a^\omega$. Then, we invoke the quadratic space upper bound for the nonemptiness of alternating automata [28], which provides the EXPSPACE upper bound since $\mathcal{A}_\phi$ is of exponential size in $|\phi|$ and $\mathcal{A}_\phi$ can be built in polynomial space in $|\phi|$.

4 Model checking nondeterministic one-counter automata

In this section, we show that several model-checking problems over nondeterministic one-counter automata are undecidable by reducing decision problems for Minsky machines. Undecidability is preserved even in presence of a unique register. This is quite surprising since f-SAT-LTL$^\downarrow$ restricted to one register is decidable [9].

In order to illustrate the significance of the following results, it is worth recalling that the halting problem for Minsky machines with incrementing errors is reducible to finitary satisfiability for LTL with one register [9]. We show below that, if we have existential model checking of one-counter automata instead of satisfiability, then we can use one-counter automata to refine the reduction in [9] so that runs with incrementing errors are excluded. More precisely, in the reduction in [9], we were not able to exclude incrementing errors because the logic is too weak to express that, for every decrement, the datum labelling it was seen before (remember that we have no past operators). Now, the one-counter automata are used to ensure that such faulty decrements cannot occur.

Theorem 14. $\mathrm{MC}_1^{<\omega}$ *is* Σ_1^0*-complete.*

Proof. The Σ_1^0 upper bound is by an easy verification since the existence of a finite run (encoded in $\mathbb{N}$) verifying an $\mathrm{LTL}_1^{\downarrow,Q}$ formula (encoded in first-order arithmetic) can be encoded by a Σ_1^0 formula. So, let us reduce the halting problem for two-counter automata to $\mathrm{MC}_1^{<\omega}$. Let $\mathcal{A} = \langle Q, q_I, \delta, F\rangle$ be a two-counter automaton: the set of instructions L is $\{\texttt{inc}, \texttt{dec}, \texttt{ifzero}\} \times \{1, 2\}$. We build a one-counter automaton $\mathcal{B} = \langle Q', q'_I, \delta', F'\rangle$ and a sentence ϕ in $\mathrm{LTL}_1^{\downarrow,Q'}$ such that $\mathcal{A}$ reaches an accepting location iff $\mathcal{B} \models^{<\omega} \phi$.

For each run in $\mathcal{A}$ $\begin{pmatrix} q_I \\ c_1^0 = 0 \\ c_2^0 = 0 \end{pmatrix} \xrightarrow{\texttt{inst}^0} \begin{pmatrix} q^1 \\ c_1^1 \\ c_2^1 \end{pmatrix} \xrightarrow{\texttt{inst}^1} \cdots \begin{pmatrix} q^N \\ c_1^N \\ c_2^N \end{pmatrix}$ where $\texttt{inst}^\texttt{i}$s are instructions, we associate the run in $\mathcal{B}$ below

$$\begin{pmatrix} q_I \\ 0 \end{pmatrix} \xrightarrow{\star} \begin{pmatrix} \langle q_I, \texttt{inst}^0, q^1\rangle \\ n^1 \end{pmatrix} \xrightarrow{\star} \begin{pmatrix} \langle q^1, \texttt{inst}^1, q^2\rangle \\ n^2 \end{pmatrix} \cdots \begin{pmatrix} \langle q^{N-1}, \texttt{inst}^{\texttt{N-1}}, q^N\rangle \\ n^N \end{pmatrix}$$

where $\xrightarrow{\star}$ hides steps for updating the counter according to the constraints described below. During these steps, auxiliary locations are used and there are of two types: locations that increment or decrement the counter in order to reach an adequate data value ($\mathrm{busyup}_{t,t'}$ and $\mathrm{busydown}_{t,t'}$ where t, t' are transitions) and intermediate locations to perform ϵ-transitions. The data values in the run of $\mathcal{B}$ are governed by the rules below:

(ii) after any configuration labelled by $\langle q, \texttt{inc}, c, q'\rangle$ (incrementation of the counter c), there is no configuration labelled by some $\langle q_1, \texttt{inc}, c', q'_1\rangle$ with the same counter value,

(iii) after any configuration labelled by $\langle q, \texttt{inc}, c, q'\rangle$, there is at most one configuration labelled by some $\langle q_1, \texttt{dec}, c, q'_1\rangle$ with the same counter value (there are more incrementations than decrementations),

(iv) after any configuration labelled by $\langle q, \texttt{inc}, c, q'\rangle$, there is no configuration labelled by some $\langle q_1, \texttt{dec}, c', q'_1\rangle$ with the same counter value and $c \neq c'$,

(v) after any configuration labelled by $\langle q, \texttt{inc}, c, q'\rangle$, there is no configuration labelled by $\langle q_1, \texttt{ifzero}, c, q'_1\rangle$ followed by a configuration labelled by some $\langle q_1, \texttt{dec}, c, q'_1\rangle$ with the same counter value as $\langle q, \texttt{inc}, c, q'\rangle$,

(vi) after any configuration labelled by $\langle q, \texttt{inc}, c, q'\rangle$ for which there is no subsequent configuration labelled by $\langle q_1, \texttt{dec}, c, q'_1\rangle$ with the same counter value, there is also no $\langle q_2, \texttt{ifzero}, c, q'_2\rangle$, Now, let us define $\mathcal{B}$. We shall partly encode in its control graph the satisfaction of these conditions. For instance, two successive incrementation transitions in $\mathcal{A}$, leads to an incrementation in $\mathcal{B}$ since we enforce that the counter value is fresh in $\mathcal{B}$ iff its letter is some $\langle -, \mathtt{inc}, -, - \rangle$ (incrementation instructions). When we write $q \xrightarrow{\top} q'$ we mean $q \xrightarrow{\mathtt{inc}} auxi_{q,q'} \xrightarrow{\mathtt{dec}} q'$ for an auxiliary location $auxi_{q,q'}$.

- Q' is equal to $\delta \uplus (\{q_I\} \cup \{\text{busydown}_{t,t'}, \text{busyup}_{t,t'} \ : \ t, t' \in \delta\})$ plus some unspecified auxiliary locations,
- $F' = \{\langle q, l, c, q' \rangle \in \delta : q \in F\} \cup (\{q_I\} \cap F)$ and $q'_I = q_I$,
- The relation δ' contains the following transitions:
  - For $\langle q_I, \mathtt{inc}, c, q \rangle \in \delta$, add $q'_I \xrightarrow{\mathtt{inc}} \langle q_I, \mathtt{inc}, c, q \rangle$ to δ';
  - For $t = \langle q_I, \mathtt{ifzero}, c, q \rangle \in \delta$, add $q'_I \xrightarrow{\top} t$ to δ';
  - For every transition $t = \langle q, \mathtt{inc}, c, q' \rangle \in \delta$,
    1. if $t' = \langle q', \mathtt{inc}, c', q'' \rangle \in \delta$, then add $t \xrightarrow{\mathtt{inc}} t'$ to δ',
    2. if $t' = \langle q', \mathtt{ifzero}, c', q'' \rangle \in \delta$ with $c' \neq c$ or $t' = \langle q', \mathtt{dec}, c, q'' \rangle \in \delta$, then add $t \xrightarrow{\top} t'$ to δ',
    3. if $t' = \langle q', \mathtt{dec}, c', q'' \rangle \in \delta$ with $c' \neq c$, then add $t \xrightarrow{\top} \text{busydown}_{t,t'}$, $\text{busydown}_{t,t'} \xrightarrow{\mathtt{dec}} \text{busydown}_{t,t'}$, and $\text{busydown}_{t,t'} \xrightarrow{\mathtt{dec}} t'$ to δ' (decrement the counter until it reaches a value for a previous incrementation),
  - For every transition $t = \langle q, l, c, q' \rangle \in \delta$ with $l \in \{\mathtt{dec}, \mathtt{ifzero}\}$,
    1. if $t' = \langle q', \mathtt{inc}, c', q'' \rangle \in \delta$, then add $t \xrightarrow{\top} \text{busyup}_{t,t'}$, $\text{busyup}_{t,t'} \xrightarrow{\mathtt{inc}} \text{busyup}_{t,t'}$, and $\text{busyup}_{t,t'} \xrightarrow{\mathtt{inc}} t'$ to δ' (increment the counter until it reaches a new value),
    2. if $t' = \langle q', \mathtt{ifzero}, c', q'' \rangle \in \delta$ then add $t \xrightarrow{\top} t'$ to δ',
    3. if $t' = \langle q', \mathtt{dec}, c', q" \rangle \in \delta$, then add to δ' the transitions from Figure 2. Observe that this is the only case for which we do not know whether the counter increases or not.

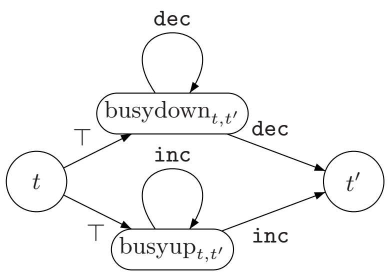


Fig. 2. Transitions in δ'

In runs of $\mathcal{B}$, we are only interested in positions with letters in δ. The control graph of $\mathcal{B}$ guarantees that the succession of transitions in $\mathcal{A}$ is valid assuming that we ignore the intermediate (auxiliary or busy) configurations

The formula ϕ is the conjunction of the following requirements: (ii)-(vi) plus

(i) some configuration in F' is visited,

(vii) after any configuration labelled by $t = \langle q, \mathtt{inc}, c, q' \rangle$, there is no configuration labelled by some $\mathrm{busyup}_{t,t'}$ with the same counter value and such that the next configuration has the same label unless there is some configuration labelled by some $\langle q_1, \mathtt{inc}, c, q_1' \rangle$ in between,

$$\mathtt{G}(t \Rightarrow \downarrow_1 \neg(\neg(\bigvee_{\langle -,\mathtt{inc},-,- \rangle} \langle -, \mathtt{inc}, -, - \rangle)\mathtt{U} \bigvee_{t'} (\mathrm{busyup}_{t,t'} \wedge \uparrow_1 \wedge \mathtt{X}\ \mathrm{busyup}_{t,t'})))$$

(viii) after any configuration labelled by $t = \langle q, \mathtt{inc}, c, q' \rangle$, there is no configuration labelled by some $\langle q_1, \mathtt{dec}, c, q_1' \rangle$ with a different counter value unless there is some configuration labelled by some $\langle q_2, \mathtt{inc}, c, q_2' \rangle$ in between,

$$\mathtt{G}(t \Rightarrow \downarrow_1 \neg(\neg(\bigvee_{\langle -,\mathtt{inc},-,- \rangle} \langle -, \mathtt{inc}, -, - \rangle)\mathtt{U}(\bigvee_{\langle -,\mathtt{dec},c,- \rangle} (\langle -, \mathtt{dec}, c, - \rangle \wedge \neg \uparrow_1))))$$

It is easy to check that each condition in (i)-(viii) can be expressed in $\mathrm{LTL}_1^{\downarrow,Q'}$ (some examples are indeed provided above). Now consider any run of $\mathcal{B}$ which satisfies (ii)-(viii). The key achievement of the definitions of $\mathcal{B}$ and ϕ is that, for every position in the run, the counter value is fresh iff either its letter is some $\langle q, \mathtt{inc}, c, q' \rangle$ or the letter is not in $\delta \cup \{q_I\}$. For any counter $c \in \{1, 2\}$, we can define its value as the number of $\langle q, \mathtt{inc}, c, q' \rangle$ letters for which a latter letter $\langle q_1, \mathtt{dec}, c, q_1' \rangle$ with the same value of the counter $\mathcal{B}$ has not yet occurred. Observe that the conditions (vii), (viii) and the control graph of $\mathcal{B}$ induce a stack discipline for the counter values of configurations with labels of the form either $\langle -, \mathtt{inc}, c, - \rangle$ and $\langle -, \mathtt{dec}, c, - \rangle$. This guarantees that no configuration labelled by $\langle -, \mathtt{dec}, c, - \rangle$ has a new counter value.

For any run of $\mathcal{B}$ which satisfies (ii)-(viii), we can thus extract a valid run of $\mathcal{A}$. Conversely, any valid run of $\mathcal{A}$ can be encoded in the same way as a run of $\mathcal{B}$ which satisfies (ii)-(viii). The latter is done by inserting auxiliary letters as required to reach appropriate values of the counter of $\mathcal{B}$. □

Theorem 15. MC_1^{ω} *is* Σ_1^1*-complete.*

The proof is similar to the proof of Theorem 14 except that instead of reducing the halting problem for Minsky machines, we reduce the recurrence problem for nondeterministic Minsky machines that is known to be Σ_1^1-hard [2]. The Σ_1^1 upper bound is by an easy verification since an accepting run can be viewed as a function $f : \mathbb{N} \to \mathbb{N}$ and then checking that it satisfies an $\mathrm{LTL}_1^{\downarrow,Q}$ formula can be expressed in first-order arithmetic. Another consequence of the Purification Lemma is the result below.

Theorem 16. $\mathrm{PureMC}_1^{<\omega}$ *is* Σ_1^0*-complete and* $\mathrm{PureMC}_1^{\omega}$ *is* Σ_1^1*-complete.*

The above-mentioned undecidability holds true even if we restrict ourselves to one-counter automata for which there are no transitions with identical instructions going from the same location. A one-counter automaton $\mathcal{A}$ is *weakly deterministic* whenever for every location q, if $\langle q, l, q' \rangle, \langle q, l', q'' \rangle \in \delta$, we have $l = l'$ implies $q' = q''$. The transition systems induced by these automata are not necessarily deterministic.

Theorem 17. $\mathrm{PureMC}_1^{<\omega}$ *[resp.* $\mathrm{PureMC}_1^{\omega}$*] restricted to weakly deterministic one-counter automata is* Σ_1^0*-complete [resp.* Σ_1^1*-complete].*

The proof uses the Purification Lemma and provides reductions from the model-checking problems to their restrictions to weakly deterministic automata.

5 Conclusion

We have shown that model checking $\mathrm{LTL}^{\downarrow}$ over one-counter automata is undecidable, which contrasts with the decidability of many verification problems for one-counter automata [14,26]. For instance, we have shown that model checking nondeterministic one-counter automata over $\mathrm{LTL}^{\downarrow}$ restricted to a unique register and without alphabet is already Σ_1^1-complete in the infinitary case. On the decidability side, a suitable abstraction has been introduced to establish the PSPACE upper bound for model checking $\mathrm{LTL}^{\downarrow}$ over deterministic one-counter automata in the infinitary case.

Viewing runs as data words is an idea that can be pushed further. For instance, the decidability status of model checking $\mathrm{LTL}^{\downarrow}$ over the class of reversal-bounded counter automata [8] remains open. Hence, our results pave the way for model checking $\mathrm{LTL}^{\downarrow}$ over other classes of operational models that are known to admit powerful techniques for solving verification tasks. Finally, among the specific problems left open by this paper, we wish to mention the complexity of model-checking deterministic one-counter automata with $\mathrm{LTL}^{\downarrow}$ in the finitary case (the complexity is however known in the infinitary case). Finitary nonemptiness problem for 1-letter hesitant alternating word automata also faces the difficulty to determine the end of the word (synchronization is needed), see e.g. [17].

Acknowledgement: We would like to thank Philippe Schnoebelen for suggesting simplifications in the proofs of Lemma 2 and Proposition 3.

References

1. R. Alur and D. Dill. A theory of timed automata. *TCS*, 126:183–235, 1994.
2. R. Alur and T. Henzinger. A really temporal logic. In *FOCS'89*, pages 164–169. IEEE, 1989.
3. O. Bernholtz, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In *CAV'94*, volume 818 of *LNCS*, pages 142–155. Springer, 1994.
4. M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. In *PODS'06*, pages 10–19, 2006.
5. M. Bojańczyk, A. Muscholl, T. Schwentick, L. Segoufin, and C. David. Two-variable logic on words with data. In *LICS'06*, pages 7–16. IEEE, 2006.
6. A. Bouajjani, Y. Jurski, and M. Sighireanu. A generic framework for reasoning about dynamic networks of infinite-state processes. In *TACAS'07*, volume 4424 of *LNCS*, pages 690–705. Springer, 2007.

7. P. Bouyer, A. Petit, and D. Thérien. An algebraic approach to data languages and timed languages. *I & C*, 182(2):137–162, 2003.
8. Z. Dang, O. Ibarra, and P. S. Pietro. Liveness verification of reversal-bounded multicounter machines with a free counter. In *FST&TCS'01*, volume 2245 of *LNCS*, pages 132–143. Springer, 2001.
9. S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. In *LICS'06*, pages 17–26. IEEE, 2006.
10. S. Demri, R. Lazić, and D. Nowak. On the freeze quantifier in constraint LTL: decidability and complexity. *I & C*, 205(1):2–24, 2007.
11. S. Demri, R. Lazić, and A. Sangnier. Model checking freeze LTL over one-counter automata. Research report, Laboratoire Spécification et Vérification, ENS Cachan, 2008.
12. M. Franceschet and M. de Rijke. Model checking hybrid logics (with an application to semistructured data). *Journal of Applied Logic*, 4(3):279–304, 2006.
13. V. Goranko. Hierarchies of modal and temporal logics with references pointers. *Journal of Logic, Language, and Information*, 5:1–24, 1996.
14. P. Jančar, A. Kučera, F. Moller, and Z. Sawa. DP lower bounds for equivalence-checking and model-checking of one-counter automata. *I & C*, 188(1):1–19, 2004.
15. M. Jurdziński and R. Lazić. Alternation-free modal mu-calculus for data trees. In *LICS'07*, pages 131–140, 2007.
16. O. Kupferman and M. Vardi. Memoryful Branching-Time Logic. In *LICS'06*, pages 265–274. IEEE, 2006.
17. O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *JACM*, 47(2):312–360, 2000.
18. F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal logic with forgettable past. In *LICS'02*, pages 383–392. IEEE, 2002.
19. R. Lazić. Safely freezing LTL. In *FST&TCS'06*, volume 4337, pages 381–392. LNCS, 2006.
20. N. Markey and P. Schnoebelen. Model checking a path. In *CONCUR'03*, volume 2761 of *LNCS*, pages 251–261. Springer, 2003.
21. F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *TOCL*, 5(3):403–435, 2004.
22. J. Ouaknine and J. Worrell. On Metric Temporal Logic and faulty Turing machines. In *FOSSACS'06*, volume 3921 of *LNCS*, pages 217–230. Springer, 2006.
23. J. Ouaknine and J. Worrell. On the decidability and complexity of metric temporal logic over finite words. *Logical Methods in Computer Science*, 3(1:8):1–27, 2007.
24. T. Schwentick and V. Weber. Bounded-variable fragments of hybrid logics. In *STACS'07*, volume 4393 of *LNCS*, pages 561–572. Springer, 2007.
25. L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL'06*, volume 4207 of *LNCS*, pages 41–57. Springer, 2006.
26. O. Serre. Parity games played on transition graphs of one-counter processes. In *FOSSACS'06*, volume 3921 of *LNCS*, pages 337–351. Springer, 2006.
27. B. ten Cate and M. Franceschet. On the complexity of hybrid logics with binders. In *CSL'05*, volume 3634 of *LNCS*, pages 339–354. Springer, 2005.
28. M. Vardi. Alternating automata and program verification. In *CS Today – Recent Trends and Developments*, volume 1000 of *LNCS*, pages 471–485. Springer, 1996.
29. M. Vardi. Alternating automata: unifying truth and validity checking for temporal logics. In *CADE-14*, volume 1249 of *LNCS*, pages 191–206. Springer, 1997.
30. M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *JCSS*, 32:183–221, 1986.