Model checking memoryful linear-time logics over one-counter automata[☆]

Stéphane Demri[a], Ranko Lazić[b], Arnaud Sangnier[c]

[a]*LSV, ENS Cachan, CNRS, INRIA Saclay IdF, France*
[b]*Department of Computer Science, University of Warwick, UK*
[c]*LSV, ENS Cachan, CNRS & EDF R&D, France*

Abstract

We study complexity of the model-checking problems for LTL with registers (also known as freeze LTL and written $\mathrm{LTL}^{\downarrow}$) and for first-order logic with data equality tests (written $\mathrm{FO}(\sim,<,+1)$) over one-counter automata. We consider several classes of one-counter automata (mainly deterministic vs. nondeterministic) and several logical fragments (restriction on the number of registers or variables and on the use of propositional variables for control states). The logics have the ability to store a counter value and to test it later against the current counter value. We show that model checking $\mathrm{LTL}^{\downarrow}$ and $\mathrm{FO}(\sim,<,+1)$ over deterministic one-counter automata is PSPACE-complete with infinite and finite accepting runs. By constrast, we prove that model checking $\mathrm{LTL}^{\downarrow}$ in which the until operator $\mathtt{U}$ is restricted to the eventually $\mathtt{F}$ over nondeterministic one-counter automata is Σ^1_1-complete [resp. Σ^0_1-complete] in the infinitary [resp. finitary] case even if only one register is used and with no propositional variable. As a corollary of our proof, this also holds for $\mathrm{FO}(\sim,<,+1)$ restricted to two variables (written $\mathrm{FO}_2(\sim,<,+1)$). 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 $\mathrm{LTL}^{\downarrow}$ and $\mathrm{FO}_2(\sim,<,+1)$ are decidable. Our results pave the way for model-checking memoryful (linear-time) logics over other classes of operational models, such as reversal-bounded counter machines.

Keywords: one-counter automaton, temporal logic, first-order logic, computational complexity

1. Introduction

Logics for data words. 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 arises in systems that are potentially unbounded in some way. Typical examples are runs of counter systems [1], timed words accepted by timed automata [2] and runs of systems with unboundedly many parallel components (data are component indices) [3]. The extension to trees makes also sense to model XML documents with values, see e.g. [4, 5, 6]. 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. [7, 8]. This is a powerful feature shared by other memoryful temporal logics [9, 10]. 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 [7] and LTL with registers (also known as freeze LTL) restricted to a single register is undecidable over infinite data words [8]. By contrast, decidable fragments of the satisfiability problems have been found in [11, 7, 12, 8, 13] 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 [14, 15]. A key point for all these logical formalisms is the ability to store a value from an infinite alphabet, which is a feature also present in models of register

[☆]Supported by the Agence Nationale de la Recherche, grant ANR-06-SETIN-001.

automata, see e.g. [16, 17, 18, 19]. However, the storing mechanism has a long tradition (apart from its ubiquity in programming languages) since it appeared for instance in real-time logics [20] (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 [21]. Meaningful restrictions for hybrid logics can also lead to decidable fragments, see e.g. [22].

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 control states 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 [12], most decision problems considered in [12, 13, 8] 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 [13, 12] and from those in [23, 24, 25] dealing with so-called hybrid logics. 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 [12, 8] may not be allowed on runs. This is precisely what we want to understand in this work. As a second main motivation, we would like to compare the results on LTL with registers with those for first-order logic with data equality tests. Indeed, LTL (with past-time operators) and first-order logic are equivalently expressive by Kamp's theorem, but such a correspondence in presence of data values is not known. Our investigation about the complexity of model-checking one-counter automata with memoryful logics include then first-order logic.

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 mean to specify an infinite path on which model checking is performed, see analogous issues in [26].

We consider several classes of one-counter automata (deterministic, weakly 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 results from [14, 15, 8, 13], 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 establish the following results.

- Model checking LTL with registers [resp. first-order logic with data equality test] over deterministic one-counter automata is PSPACE-complete (see Sect. 3.3). PSPACE-hardness is established by reducing QBF and it also holds when no letters from the finite alphabet are used in formulae. In order to get these complexity upper bounds, we translate our problems into model-checking first-order logic without data equality test over ultimately periodic words that can be solved in polynomial space thanks to [26].
- 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^0_1-complete by reducing the halting problem for Minsky machines. These results are quite surprising since several verification problems for one-counter automata are decidable with relatively low complexity [27, 28, 29]. Moreover, finitary satisfiability for LTL with one register is decidable [8] even though with non-primitive recursive complexity. These results can be also obtained for first-order logic with data equality test restricted to two variables by analysing the structure of formulae used in the undecidability proofs and by using [8].

Figure 1 contains a summary of the main results we obtained; notations are fully explained in Section 2. For instance, $\text{MC}(\text{LTL})^{\omega}_{1}[\mathtt{X}, \mathtt{F}]$ refers to the existential model-checking problem on infinite accepting runs from

PSpace-completeness for det. 1CA	Σ^0_1-completeness for 1CA	Σ^1_1-completeness for 1CA
$\mathrm{MC(LTL)}^\omega$, $\mathrm{MC(LTL)}^*$ $\mathrm{MC(LTL)}^\omega[\mathtt{F}]$, $\mathrm{MC(LTL)}^*[\mathtt{X},\mathtt{F}]$	$\mathrm{MC(LTL)}^*_1[\mathtt{X},\mathtt{F}]$ $\mathrm{PureMC(LTL)}^*_1[\mathtt{X},\mathtt{F}]$	$\mathrm{MC(LTL)}^\omega_1[\mathtt{X},\mathtt{F}]$ $\mathrm{PureMC(LTL)}^\omega_1[\mathtt{X},\mathtt{F}]$
$\mathrm{MC(FO)}^\omega$, $\mathrm{MC(FO)}^*$ $\mathrm{MC(FO)}^\omega[\sim,<]$	$\mathrm{MC(FO)}^*_2[\sim,<]$	$\mathrm{MC(FO)}^\omega_2[\sim,<]$

Figure 1: Summary of main results

one-counter automata with freeze LTL restricted to the temporal operators “next” and “sometimes”, and to a unique register. Similarly, $\mathrm{MC(FO)}^\omega_2[\sim,<]$ refers to the existential model-checking problem on finite accepting runs from one-counter automata with first-order logic on data words restricted to two individual variables.

Plan of the paper. In Sect. 2, we introduce the model-checking problem for LTL with registers over one-counter automata as well as the corresponding problem for first-order logic with data equality test. In Sect. 3, we consider decidability and complexity issues for model checking deterministic one-counter automata. In Sect. 4, several model-checking problems over nondeterministic one-counter automata are shown undecidable.

This paper is an extended version of [30] that also improves significantly the results about the PSpace upper bounds and the undecidability results, in particular by considering first-order language over data words.

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 states,
- $q_I \in Q$ is the initial state,
- $F \subseteq Q$ is the set of accepting states,
- $\delta \subseteq Q \times L \times Q$ 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 $\rho = \langle q_0, n_0\rangle \rightarrow \langle q_1, n_1\rangle \rightarrow \cdots \rightarrow \langle q_f, n_f\rangle$ is *accepting* iff q_f is an accepting state. An infinite run ρ is accepting iff it contains an accepting state infinitely often (Büchi acceptance condition). All these notations can be naturally adapted to multicounter automata.

A one-counter automaton $\mathcal{A}$ is *deterministic* whenever it corresponds to a deterministic one-counter Minsky machine: for every state 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 `ifzero` and the other with instruction `dec`,

- or $\mathcal{A}$ has no transition from q (not present in original deterministic Minsky machines [1]).

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 $\mathrm{LTL}^{\downarrow,\Sigma}$ [8] where Σ is a finite alphabet are defined as follows:

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

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 Σ^* 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 $0 \le i < |\sigma|$. Let $\Sigma^*(\sim)$ [resp. $\Sigma^\omega(\sim)$] denote the sets of all such finite [resp. infinite] data words. We denote by $\Sigma^\infty(\sim)$ the set $\Sigma^*(\sim)\cup\Sigma^\omega(\sim)$ of finite and infinite data words.

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. Let σ be a data word in $\Sigma^\infty(\sim)$ and $0 \le i < |\sigma|$, the satisfaction relation $\models$ is defined as follows (Boolean clauses are omitted).

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

$v[r \mapsto i]$ 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 ($\mathtt{G}$, $\mathtt{F}$, $\mathtt{G}^+$, $\mathtt{F}^+$, ...) and for the Boolean operators and constants ($\vee$, $\Rightarrow$, $\top$, $\bot$, ...). The finitary [resp. infinitary] satisfiability problem for LTL with registers, noted $*$-SAT-$\mathrm{LTL}^{\downarrow}$ [resp. ω-SAT-$\mathrm{LTL}^{\downarrow}$], is defined as follows:

Input: A finite alphabet Σ and a formula ϕ in $\mathrm{LTL}^{\downarrow,\Sigma}$;

Question: Is there a finite [resp. an infinite] data word σ such that $\sigma, 0 \models \phi$?

Theorem 1. *[8, Theorem 5.2] $*$-SAT-$\mathrm{LTL}^{\downarrow}$ restricted to one register is decidable with non-primitive recursive complexity and ω-SAT-$\mathrm{LTL}^{\downarrow}$ restricted to one register is Π^0_1-complete.*

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 sometimes store counter values in registers, 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 $\mathrm{MC(LTL)}^*$ [resp. $\mathrm{MC(LTL)}^\omega$], is defined as follows:

Input: A one-counter automaton $\mathcal{A} = \langle Q, q_I, \delta, F\rangle$ and a sentence ϕ in $\mathrm{LTL}^{\downarrow,Q}$;

Question: 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 $\mathcal{A} \models^{*} \phi$ [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 $\mathrm{MC(LTL)}^{\alpha}_{n}$ to denote the restriction of $\mathrm{MC(LTL)}^{\alpha}$ to formulae with at most n registers. Very often, it makes sense that only counter values are known but not the current state of a configuration, which can be understood as an internal information about the system. We write $\mathrm{PureMC(LTL)}^{\alpha}_{n}$ to denote the restriction of $\mathrm{MC(LTL)}^{\alpha}_{n}$ (its "pure data" version) to formulae with atomic formulae only of the form $\uparrow_r$. Given a set $\mathcal{O}$ of temporal operators, we write $\mathrm{MC(LTL)}^{\alpha}_{n}[\mathcal{O}]$ [resp. $\mathrm{PureMC(LTL)}^{\alpha}_{n}[\mathcal{O}]$] to denote the restriction of $\mathrm{MC(LTL)}^{\alpha}_{n}$ [resp. $\mathrm{PureMC(LTL)}^{\alpha}_{n}$] to formulae using only temporal operators in $\mathcal{O}$.

Example 1. *Here are some properties that can be stated in* $\mathrm{LTL}^{\downarrow,Q}_{2}$ *along a run.*

- *"There is a suffix such that all the counter values are different":*

$$\mathtt{FG}(\downarrow_1 \mathtt{G}^{+} \neg \uparrow_1).$$

- *"Whenever state 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":*

$$\mathtt{G}(q \Rightarrow \downarrow_1 \mathtt{X} \downarrow_2 \mathtt{XG}(q \Rightarrow \uparrow_1 \wedge \mathtt{X} \uparrow_2)).$$

Observe also that we have chosen as alphabet the set of states of the automata. Alternatively, it would have been possible to add finite alphabets to automata, to label each transition by a letter and then consider as data words generated from automata the recognized words augmented with the counter values. This choice does not change our main results but it improves the readability of some technical details.

2.3. First-order logic over data words

Let us introduce the second logical formalism considered in the paper. Formulae of $\mathrm{FO}^{\Sigma}(\sim, <, +1)$ [7] where Σ is a finite alphabet are defined as follows:

$$\phi ::= a(\mathtt{x}) \mid \mathtt{x} \sim \mathtt{y} \mid \mathtt{x} < \mathtt{y} \mid \mathtt{x} = \mathtt{y} + 1 \mid \neg\phi \mid \phi \wedge \phi \mid \exists\, \mathtt{x}\ \phi$$

where $a \in \Sigma$ and $\mathtt{x}$ ranges over a countably infinite set of variables. We write $\mathrm{FO}(\sim, <, +1)$ to denote $\mathrm{FO}^{\Sigma}(\sim, <, +1)$ for some unspecified finite alphabet and $\mathrm{FO}(<, +1)$ to denote the restriction of $\mathrm{FO}(\sim, <, +1)$ without atomic formulae of the form $\mathtt{x} \sim \mathtt{y}$. Given a natural number $n > 0$, we write $\mathrm{FO}^{\Sigma}_{n}(\sim, <, +1)$ to denote the restriction of $\mathrm{FO}^{\Sigma}(\sim, <, +1)$ to variables in $\{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$. A variable valuation u for a data word σ is a finite partial map from the set of variables to the indices of σ. Let σ be a data word in $\Sigma^{\infty}(\sim)$, the satisfaction relation $\models$ is defined as follows (Boolean clauses are again omitted):

$$\begin{array}{rcl}
\sigma \models_u a(\mathtt{x}) & \overset{\text{def}}{\Leftrightarrow} & u(\mathtt{x}) \text{ is defined and } \sigma(u(\mathtt{x})) = a \\
\sigma \models_u \mathtt{x} \sim \mathtt{y} & \overset{\text{def}}{\Leftrightarrow} & u(\mathtt{x}) \text{ and } u(\mathtt{y}) \text{ are defined and } u(\mathtt{x}) \sim^{\sigma} u(\mathtt{y}) \\
\sigma \models_u \mathtt{x} < \mathtt{y} & \overset{\text{def}}{\Leftrightarrow} & u(\mathtt{x}) \text{ and } u(\mathtt{y}) \text{ are defined and } u(\mathtt{x}) < u(\mathtt{y}) \\
\sigma \models_u \mathtt{x} = \mathtt{y} + 1 & \overset{\text{def}}{\Leftrightarrow} & u(\mathtt{x}) \text{ and } u(\mathtt{y}) \text{ are defined and } u(\mathtt{x}) = u(\mathtt{y}) + 1 \\
\sigma \models_u \exists\, \mathtt{x}\ \phi & \overset{\text{def}}{\Leftrightarrow} & \text{there is } i \in \mathbb{N} \text{ such that } 0 \leq i < |\sigma| \text{ and } \sigma \models_{u[\mathtt{x} \mapsto i]} \phi
\end{array}$$

$u[\mathtt{x} \mapsto i]$ denotes the variable valuation equal to u except that the variable $\mathtt{x}$ is mapped to the position i. In the sequel, we omit the subscript "u" in $\models_u$ when sentences are involved.

The finitary [resp. infinitary] (existential) model-checking problem over one-counter automata for the logic $\mathrm{FO}^{\Sigma}(\sim, <, +1)$, noted $\mathrm{MC(FO)}^{*}$ [resp. $\mathrm{MC(FO)}^{\omega}$] is defined as follows:

Input: A one-counter automaton $\mathcal{A}$ and a sentence ϕ in $\mathrm{FO}^{Q}(\sim, <, +1)$;

Question: Is there a finite [resp. infinite] accepting run ρ of $\mathcal{A}$ such that $\rho \models \phi$? If the answer is "yes", we write $\mathcal{A} \models^{*} \phi$ [resp. $\mathcal{A} \models^{\omega} \phi$].

We write $\mathrm{MC(FO)}^{\alpha}_{n}$ to denote the restriction of $\mathrm{MC(FO)}^{\alpha}$ to formulae with at most n variables. We write $\mathrm{PureMC(FO)}^{\alpha}_{n}$ to denote the restriction of $\mathrm{MC(FO)}^{\alpha}_{n}$ (its "pure data" version) to formulae with no atomic formulae of the form $a(\mathtt{x})$.

Extending the standard translation from LTL into first-order logic, we can easily establish the result below.

Lemma 2. *Given a sentence ϕ in $\mathrm{LTL}^{\downarrow,\Sigma}_{n}$, there is a first-order formula ϕ' in $\mathrm{FO}^{\Sigma}(\sim, <, +1)$ that can be computed in linear time in $|\phi|$ such that*

1. *ϕ' has at most $max(3, n+1)$ variables,*
2. *ϕ' has a unique free variable, say $\mathtt{y}_0$,*
3. *for all data words σ, register valuations v and $i \geq 0$, we have $\sigma, i \models_{v} \phi$ iff $\sigma \models_{u} \phi'$, where for $r \in \{1, \ldots, n\}$, $v(r) = u(\mathtt{x}_r)$ and $u(\mathtt{y}_0) = i$.*

Proof. We build a translation function T which takes as arguments a formula in $\mathrm{LTL}^{\downarrow,\Sigma}_{n}$ and a variable, and which returns the wanted formula in $\mathrm{FO}^{\Sigma}(\sim, <, +1)$. Intuitively the variable, which is given as argument, is used to represent the current position in the data word. Then, we use the variables $\mathtt{x}_1, \ldots, \mathtt{x}_r$ to characterize the registers. We add to this set of variables three variables $\mathtt{y}_0, \mathtt{y}_1$ and $\mathtt{y}_2$. In the sequel, we write $\mathtt{y}$ to represent indifferently $\mathtt{y}_0$ or $\mathtt{y}_1$ or $\mathtt{y}_2$. Furthermore the notation $\mathtt{y}_{i+1}$ stands for $\mathtt{y}_{(i+1)mod(3)}$ and $\mathtt{y}_{i+2}$ stands for $\mathtt{y}_{(i+2)mod(3)}$. The function T, which is homomorphic for the Boolean operators, is defined inductively as follows, for $i \in \{0, 1, 2\}$:

- $T(a, \mathtt{y}) = a(\mathtt{y})$,
- $T(\uparrow_r, \mathtt{y}) = \mathtt{y} \sim \mathtt{x}_r$,
- $T(\mathtt{X}\phi, \mathtt{y}_i) = \exists\ \mathtt{y}_{i+1}\ (\mathtt{y}_{i+1} = \mathtt{y}_i + 1 \wedge T(\phi, \mathtt{y}_{i+1}))$,
- $T(\phi \mathtt{U} \psi, \mathtt{y}_i) = \exists\ \mathtt{y}_{i+1}\ (\mathtt{y}_i \leq \mathtt{y}_{i+1} \wedge T(\psi, \mathtt{y}_{i+1}) \wedge \forall\ \mathtt{y}_{i+2}\ (\mathtt{y}_i \leq \mathtt{y}_{i+2} < \mathtt{y}_{i+1} \Rightarrow T(\phi, \mathtt{y}_{i+2}))$,
- $T(\downarrow_r \phi, \mathtt{y}) = \exists\ \mathtt{x}_r\ (\mathtt{x}_r = \mathtt{y} \wedge T(\phi, \mathtt{y}))$.

Then if ϕ is a formula in $\mathrm{LTL}^{\downarrow,\Sigma}_{n}$ and $\mathtt{y}_0$ is the variable chosen to characterize the current position in the word, the formula $T(\phi, \mathtt{y}_0)$ satisfies the three conditions given in the above lemma. In order to ensure the first condition, we use the fact that we can recycle the variables. More details about this technique can be found in [31]. ☐

The decidability borderline for $\mathrm{FO}(\sim, <, +1)$ is between two and three variables.

Theorem 3. *[7, Theorem 1, Propositions 19 & 20] Satisfiability for $\mathrm{FO}(\sim, <, +1)$ restricted to 3 variables is undecidable and satisfiability for $\mathrm{FO}_2(\sim, <, +1)$ is decidable (for both finitary and infinitary cases).*

In Section 3 we will use Theorem 4 below in an essential way.

Theorem 4. *[26, Proposition 4.2] Given two finite words $s, t \in \Sigma^{*}$ and a sentence ϕ in $\mathrm{FO}^{\Sigma}(<, +1)$, checking whether $s \cdot t^{\omega} \models \phi$ can be done in space $\mathcal{O}((|s| + |t|) \times |\phi|^2)$.*

2.4. Purification of the model-checking problem

We now show how to get rid of propositional variables by reducing the model-checking problem over one-counter automata to its pure version. This amounts to transform any MC(LTL) instance into a PureMC(LTL) instance.

Lemma 5 (Purification for $\mathrm{LTL}^{\downarrow}$). *Given a one-counter automaton $\mathcal{A}$ and a sentence ϕ in $\mathrm{LTL}^{\downarrow,Q}_{n}$, one can compute in logarithmic space in $|\mathcal{A}|+|\phi|$ a one-counter automaton $\mathcal{A}_P$ and a formula ϕ_P in $\mathrm{LTL}^{\downarrow,\emptyset}_{max(n,1)}$ such that $\mathcal{A} \models^* \phi$ [resp. $\mathcal{A} \models^\omega \phi$] iff $\mathcal{A}_P \models^* \phi_P$ [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 states 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 $Q = \{q_1, \ldots, q_m\}$ and ϕ be an $\mathrm{LTL}^{\downarrow,Q}$ formula. In order to define $\mathcal{A}_P$, we identify states with patterns about the changes of the unique counter. Let $\mathcal{A}_P$ be $\langle Q_P, q_I, \delta_P, F_P\rangle$ with $Q_P = Q \uplus Q'$ and Q' is defined below:

$$\begin{aligned} Q' = \ & \big\{q_i^1, q_i^2, q_i^3, q_i^4, q_i^5, q_{i,F} \mid i \in \{1,\ldots,m\}\big\} \\ & \cup\big\{q_{i,j}, q'_{i,j} \mid i \in \{1,\ldots,m\} \text{ and } j \in \{1,\ldots,m+1\} \text{ and } i \neq j\big\} \\ & \cup\big\{q^0_{i,i}, q_{i,i}, q^1_{i,i}, q^2_{i,i} \mid i \in \{1,\ldots,m\}\big\}. \end{aligned}$$

Figure 2 presents the set of transitions δ_P associated with each state q_i of Q (providing a pattern). Furthermore, for all $i, j \in \{1,\ldots,m\}$, $q_{i,F} \xrightarrow{\mathtt{a}} q_j \in \delta_P$ iff $q_i \xrightarrow{\mathtt{a}} q_j \in \delta$. The sequence of transitions associated to each $q_i \in Q$ is a sequence of $m+2$ picks and among these picks, the first pick is the only one of height 3, the i-th pick is the only one of height 2, and the height of all the other picks is 1. Observe that this sequence of transitions has a fixed length and it is composed of exactly $9 + 2(m+1)$ states.

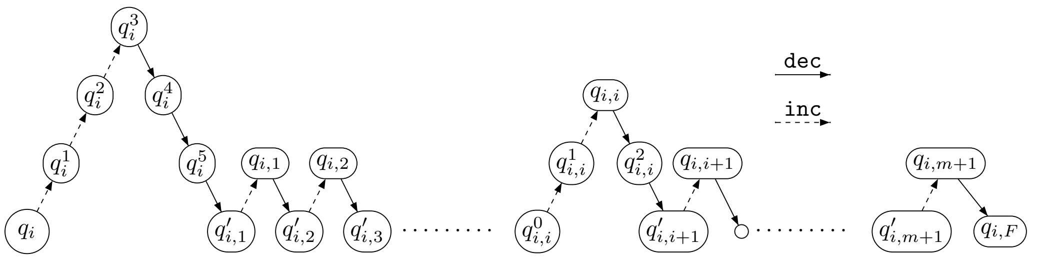


Figure 2: Encoding q_i by a pattern made of $m+2$ picks and of length $9+2(m+1)$

Finally, the set of accepting states of $\mathcal{A}_P$ is defined as the set $\{q_{i,F} \mid q_i \in F\}$. In order to detect the first pick of height 3 which characterizes the beginning of the sequence of transitions associated to each state belonging to Q, we build the two following formulae in $\mathrm{LTL}^{\downarrow,\emptyset}_{1}$:

- $\varphi_{\neg 3/7}$ which expresses that "among the 7 next counter values (including the current counter value), there are no 3 equal values",
- $\varphi_{0\sim 6}$ which expresses that "the current counter value is equal to the counter value at the 6th next position".

These two formulae can be written as follows:

$$\begin{aligned} \varphi_{\neg 3/7} \ = \ & \neg\big(\downarrow_1 \big(\textstyle\bigvee_{i\neq j\in\{1,\ldots,6\}}(\mathtt{X}^i \uparrow_1 \wedge \mathtt{X}^j \uparrow_1)\big) \\ & \vee\mathtt{X} \downarrow_1 \big(\textstyle\bigvee_{i\neq j\in\{1,\ldots,5\}}(\mathtt{X}^i \uparrow_1 \wedge \mathtt{X}^j \uparrow_1)\big) \\ & \vee\mathtt{X}^2 \downarrow_1 \big(\textstyle\bigvee_{i\neq j\in\{1,\ldots,4\}}(\mathtt{X}^i \uparrow_1 \wedge \mathtt{X}^j \uparrow_1)\big) \\ & \vee\mathtt{X}^3 \downarrow_1 \big(\textstyle\bigvee_{i\neq j\in\{1,2,3\}}(\mathtt{X}^i \uparrow_1 \wedge \mathtt{X}^j \uparrow_1)\big) \\ & \vee\mathtt{X}^4 \downarrow_1 \big(\textstyle\bigvee_{i\neq j\in\{1,2\}}(\mathtt{X}^i \uparrow_1 \wedge \mathtt{X}^j \uparrow_1)\big)\big) \end{aligned}$$

$$\varphi_{0\sim 6} \ = \ \downarrow_1 (\mathtt{X}^6 \uparrow_1)$$

We write STA to denote the formula $\varphi_{\neg 3/7} \wedge \varphi_{0\sim 6}$.

Let ρ be a run of $\mathcal{A}_P$ and j be such that $0 \leq j < |\rho|$. We show that (1) $\rho, j \models$ STA iff (2) ($\rho, j \models q$ for some $q \in Q$ and $j + 6 < |\rho|$). In the sequel, we assume that $j + 6 < |\rho|$ since otherwise it is clear that $\rho, j \not\models$ STA. By construction, it is clear that (2) implies (1). In order to prove that (1) implies (2), we show that if $\rho, j \models q$ for some $q \in Q_P \setminus Q$ and $j + 6 < |\rho|$, then $\rho, j \not\models$ STA. We perform a systematic case analysis according to the type of q (we group the cases that require similar arguments):

1. If q is of the form q_i^2 with $i \in \{2, \ldots, m\}$, then $\rho, j \not\models \varphi_{0\sim 6}$. When q is q_1^2, $\rho, j \not\models \varphi_{\neg 3/7}$.
2. If q is of the form q_i^3 with $i \in \{1, \ldots, m\}$, then $\rho, j \not\models \varphi_{0\sim 6}$.
3. If q is of the form q_i^4 with $i \in \{1, \ldots, m\} \setminus \{2\}$, then $\rho, j \not\models \varphi_{0\sim 6}$. When q is q_2^4, $\rho, j \not\models \varphi_{\neg 3/7}$.
4. If q is of the form $q_{i,i}$ with $i \in \{2, \ldots, m-1\}$, then $\rho, j \not\models \varphi_{0\sim 6}$. When q is $q_{m,m}$ and an incrementation is performed after $q_{m,F}$, we have $\rho, j \not\models \varphi_{\neg 3/7}$. If another action is performed, then we also have $\rho, j \not\models \varphi_{0\sim 6}$.
5. If q is of the form either q_i^1 or q_i^5 with $i \in \{1, \ldots, m\}$, then $\rho, j \not\models \varphi_{\neg 3/7}$.
6. If q is of the form either $q_{i,i}^0$ or $q_{i,i}^1$ with $i \in \{1, \ldots, m\}$, then $\rho, j \not\models \varphi_{\neg 3/7}$.
7. If q is of the form $q_{i,i}^2$ with $i \in \{1, \ldots, m\}$, then $\rho, j \not\models \varphi_{\neg 3/7}$ (the case $i = m$ requires a careful analysis).
8. If q is of the form $q_{i,k}$ for some $i \in \{1, \ldots, m\}$, $k \in \{1, \ldots, m-1\}$ such that either $|i-k| > 2$ or $k > i$, then $\rho, j \not\models \varphi_{\neg 3/7}$.
9. If q is of the form $q_{i,i-1}$ with $i \in \{2, \ldots, m\}$, then $\rho, j \not\models \varphi_{\neg 3/7}$.
10. If q is of the form $q_{i,i-2}$ with $i \in \{3, \ldots, m\}$, then $\rho, j \not\models \varphi_{\neg 3/7}$.
11. If q is of the form $q_{i,m}$ with $i \in \{1, \ldots, m-1\}$, then $\rho, j \not\models \varphi_{\neg 3/7}$.
12. If q is of the form $q_{i,m+1}$ with $i \in \{1, \ldots, m\}$ and an action different from decrementation is performed after $q_{i,F}$, then $\rho, j \not\models \varphi_{0\sim 6}$. When a decrementation is performed after $q_{i,F}$, we get $\rho, j \models \varphi_{0\sim 6} \wedge \neg\varphi_{\neg 3/7}$.
13. If q is of the form $q'_{i,k}$ for some $i \in \{1, \ldots, m\}$, $k \in \{1, \ldots, m-1\}$ such that either $|i-k| > 2$ or $k > i$, then $\rho, j \not\models \varphi_{\neg 3/7}$.
14. If q is of the form $q'_{i,i-1}$ with $i \in \{2, \ldots, m\}$, then $\rho, j \not\models \varphi_{\neg 3/7}$.
15. If q is of the form $q'_{i,i-2}$ with $i \in \{3, \ldots, m\}$, then $\rho, j \not\models \varphi_{\neg 3/7}$.
16. If q is of the form $q'_{i,m}$ with $i \in \{1, \ldots, m\}$, then $\rho, j \not\models \varphi_{\neg 3/7}$.
17. If q is of the form $q'_{i,m+1}$ with $i \in \{1, \ldots, m\}$, then $\rho, j \not\models \varphi_{0\sim 6}$. Indeed, the 6th next position, if any, is of the form q_k^3 for some $k \in \{1, \ldots, m\}$. The counter value at such a position is strictly greater than the one at the position j whatever is the action performed after $q_{i,F}$.
18. If q is of the form $q_{i,F}$ with $i \in \{1, \ldots, m\}$ and the action performed after $q_{i,F}$ is not a decrementation, then $\rho, j \not\models \varphi_{0\sim 6}$. When a decrementation is performed after $q_{i,F}$, we get $\rho, j \models \varphi_{0\sim 6} \wedge \neg\varphi_{\neg 3/7}$.

For $i \in \{1, \ldots, m\}$, let us define the formula $\phi_i = \mathtt{X}^{6+2(i-1)} \downarrow_1 \mathtt{X}^2 \neg \uparrow_1$. One can check that in the run of $\mathcal{A}_P$, STA $\wedge\, \phi_i$ holds true iff the current state is q_i and there are at least 6 following positions.

Let ϕ be a formula in $\mathrm{LTL}_n^{\downarrow,Q}$. We define ϕ_P as the formula $\mathrm{T}(\phi)$ such that the map T 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.

- $\mathrm{T}(q_i) = \phi_i$,
- $\mathrm{T}(\mathtt{X}\phi) = \mathtt{X}^{9+2(m+1)+1}\mathrm{T}(\phi)$,
- $\mathrm{T}(\phi \mathtt{U} \phi') = \big(\mathrm{STA} \Rightarrow \mathrm{T}(\phi)\big)\mathtt{U}\big(\mathrm{STA} \wedge \mathrm{T}(\phi')\big)$.

Observe that ϕ and ϕ_P have the same amount of registers unless ϕ has no register. For each accepting run in $\mathcal{A}$, there exists an accepting run in $\mathcal{A}_P$ and conversely for each accepting run in $\mathcal{A}_P$, there exists an accepting run in $\mathcal{A}$. Furthermore the sequence of counter values for the configurations of each of these runs which have a state in Q match. □

Lemma 6 (Purification for $\mathrm{FO}(\sim, <, +1)$**).** *Given a one-counter automaton $\mathcal{A}$ and an $\mathrm{FO}^Q(\sim, <, +1)$ sentence ϕ with n variables, one can compute in logarithmic space in $|\mathcal{A}| + |\phi|$ a one-counter automaton $\mathcal{A}_P$ and ϕ_P in $\mathrm{FO}^{\emptyset}(\sim, <, +1)$ with at most $n+2$ variables such that $\mathcal{A} \models^* \phi$ [resp. $\mathcal{A} \models^\omega \phi$] iff $\mathcal{A}_P \models^* \phi_P$ [resp. $\mathcal{A}_P \models^\omega \phi_P$]. Moreover, $\mathcal{A}$ is deterministic iff $\mathcal{A}_P$ is deterministic.*

Proof. The proof follows the lines of the proof of Lemma 5 by considering the first-order formulae corresponding to the formulae STA and ϕ_i and the same automaton construction. In order to make this construction feasible, we need to use formulae of the form $\mathtt{x} = \mathtt{y} + k$. In fact, the formulae of the form $\mathtt{x} = \mathtt{y} + 1$ are translated into formulae of the form $\mathtt{x} = \mathtt{y} + 9 + 2(m+1)$ (this case is identical to the case of the formulae of the form $\mathtt{X}\phi$). Typically, encoding $\mathtt{x} = \mathtt{y} + k$ for the constant k requires two auxiliary variables. For instance we can encode the formula $\mathtt{x} = \mathtt{y} + 4$ as follows:

$$\exists\ \mathtt{y}_2\ \mathtt{x} = \mathtt{y}_2 + 1 \wedge (\exists\ \mathtt{y}_1\ \mathtt{y}_2 = \mathtt{y}_1 + 1 \wedge (\exists\ \mathtt{y}_2\ \mathtt{y}_1 = \mathtt{y}_2 + 1 \wedge \mathtt{y}_2 = \mathtt{y} + 1))$$

Here again, we recycle the variables $\mathtt{y}_1$ and $\mathtt{y}_2$. □

3. Model checking deterministic one-counter automata

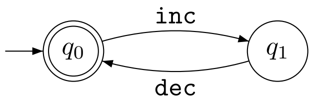
In this section, we show that $\mathrm{MC(LTL)}^*$ and $\mathrm{MC(LTL)}^\omega$ restricted to deterministic one-counter automata is PSPACE-complete.

3.1. PSPACE *lower bound*

We show below a PSPACE-hardness result by taking advantage of the alphabet of states by means of a reduction from QBF ("Quantified Boolean Formula") that is a standard PSPACE-complete problem.

Proposition 7. $\mathrm{PureMC(LTL)}^*$ *and* $\mathrm{PureMC(LTL)}^\omega$ *restricted to deterministic one-counter automata are* PSPACE*-hard problems. Furthermore, for* $\mathrm{PureMC(LTL)}^*$ *[resp.* $\mathrm{PureMC(LTL)}^\omega$*] this results holds for formulae using only the temporal operators* X *and* F *[resp.* F*].*

Proof. Consider a QBF instance ϕ: $\phi = \forall \mathrm{p}_1\ \exists \mathrm{p}_2\ \cdots\ \forall \mathrm{p}_{2N-1}\ \exists\ \mathrm{p}_{2N}\ \Psi(\mathrm{p}_1, ..., \mathrm{p}_{2N})$ 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 $\psi = \downarrow_{2N+1} \psi_1$.

- $\psi_{2N+1} = \Psi(\uparrow_1 \Leftrightarrow \uparrow_{2N+1}, \ldots, \uparrow_{2N} \Leftrightarrow \uparrow_{2N+1})$,
- 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 $\mathcal{A} \models^\omega \psi$.

To do so, we proceed as follows. For $i \in \{0, 2, 4, 6, \ldots, 2N\}$, let ϕ_i be

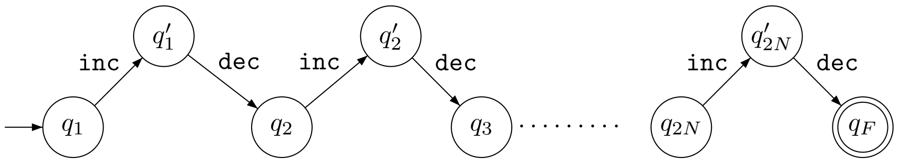
$$\phi_i = \forall \mathrm{p}_{i+1}\ \exists \mathrm{p}_{i+2}\ \cdots\ \forall \mathrm{p}_{2N-1}\ \exists\ \mathrm{p}_{2N}\ \Psi(\mathrm{p}_1, ..., \mathrm{p}_{2N}).$$

So ϕ_0 is precisely ϕ. Similarly, for $i \in \{1, 3, 5, \ldots, 2N-1\}$, let ϕ_i be

$$\phi_i = \exists \mathrm{p}_{i+1}\ \forall \mathrm{p}_{i+2}\ \cdots\ \forall \mathrm{p}_{2N-1}\ \exists\ \mathrm{p}_{2N}\ \Psi(\mathrm{p}_1, ..., \mathrm{p}_{2N}).$$

Observe that the free propositional variables in ϕ_i are exactly $\mathrm{p}_1, \ldots, \mathrm{p}_i$ and ϕ_i is obtained from ϕ by removing the i first quantifications. Given a propositional valuation $v : \{\mathrm{p}_1, \ldots, \mathrm{p}_i\} \rightarrow \{\top, \bot\}$ for some $i \in \{1, \ldots, 2N\}$, we write $\overline{v}$ to denote a register valuation such that its restriction to $\{1, \ldots, i, 2N+1\}$ satisfies: $v(\mathrm{p}_j) = \top$ iff $\overline{v}(j) = 0$ for $j \in \{1, \ldots, i\}$ and $\overline{v}(2N+1) = 0$. One can show by induction that for $k \geq 0$, $v \models \phi_{i-1}$ (in QBF) iff $\rho^\omega_{\mathcal{A}}, k \models_{\overline{v}} \psi_i$, where $\rho^\omega_{\mathcal{A}}$ denotes the unique infinite run for $\mathcal{A}$. Consequently, if $v \models \phi$ for some propositional valuation, then $\rho^\omega_{\mathcal{A}}, 0 \models_{\overline{v}} \psi$. Similarly, if $\rho^\omega_{\mathcal{A}}, 0 \models_v \psi$, then there is a propositional valuation v' such that $\overline{v'} = v$ and $v' \models \phi$.

For the finitary problem $\mathrm{PureMC(LTL)}^*$, the above proof does not work because the occurrences of G related to universal quantification in the QBF formula might lead to the end of the run, leaving no choice for the next quantifications. Consequently, one need to use another deterministic one-counter automaton with $4N+1$ states such that the sequence of counter values from the accepting run is $(01)^{2N}0$ (again we omit useless `ifzero` transitions). Let us consider the deterministic counter automaton $\mathcal{A}'$ below.



We shall build another formula ψ in $\mathrm{LTL}^{\downarrow,\emptyset}$ defined from the formulae below with $\psi = \downarrow_{2N+1} \psi_1$.

- $\psi_{2N+1} = \Psi(\uparrow_1 \Leftrightarrow \uparrow_{2N+1}, \ldots, \uparrow_{2N} \Leftrightarrow \uparrow_{2N+1})$,
- for $i \in \{1, ..., N\}$:
  - $\psi_{2i} = \mathtt{F}\big((\mathtt{X}^{4N-4i+2}\ \top) \wedge \downarrow_{2i} \psi_{2i+1}\big)$ and
  - $\psi_{2i-1} = \mathtt{G}\big((\mathtt{X}^{4N-4i+4}\ \top) \Rightarrow \downarrow_{2i-1} \psi_{2i}\big)$.

  Herein, $\top$ holds for the truth value that can be encoded with $\downarrow_1 \vee \neg \downarrow_1$ (remember there are no propositional variables in the pure version of the model-checking problems).

Using a similar proof by induction as the one done for the infinite case, we obtain that ϕ is satisfiable iff $\mathcal{A}' \models^* \psi$. □

Observe that in the reduction for $\mathrm{PureMC(LTL)}^\omega$, we use an unbounded number of registers (see Theorem 14) but a fixed deterministic one-counter automaton.

By Lemmas 6 and 2, we obtain the following corollary.

Corollary 8. $\mathrm{PureMC(FO)}^*$ *and* $\mathrm{PureMC(FO)}^\omega$ *restricted to deterministic one-counter automata are* PSPACE-*hard problems.*

3.2. 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 state is repeated infinitely often, 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 infinite 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 9 below is a key result to show the forthcoming PSPACE upper bound. Basically, the unique run of deterministic one-counter automata has regularities that can be described in polynomial size.

Lemma 9. *Let $\mathcal{A}$ be a deterministic one-counter automaton with an infinite run. There are K_1, K_2, K_{inc} such that $K_1 + K_2 \leq |Q|^3$, $K_{inc} \leq |Q|$ and for every $i \geq K_1$, $\langle q_{i+K_2}, n_{i+K_2}\rangle = \langle q_i, n_i + K_{inc}\rangle$.*

Hence, the run $\rho^\omega_\mathcal{A}$ can be encoded by its first $K_1 + K_2$ configurations. It is worth noting that we have deliberately decided to keep the three constants K_1, K_2 and K_{inc} in order to provide a more explicit analysis.

Proof. (Lemma 9) We write ZERO($\mathcal{A}$) to denote the set of positions of $\rho^\omega_\mathcal{A}$ where a zero-test has been successful. By convention, 0 belongs to ZERO($\mathcal{A}$) since in a run we require that the first configuration is the initial configuration of $\mathcal{A}$ with counter value 0. Hence, $\mathrm{ZERO}(\mathcal{A}) \overset{\text{def}}{=} \{0\} \cup \{i > 0 : n_i = n_{i+1} = 0\}$. Let us first establish Lemma 10 below.

Lemma 10. *Let $i < j$ be in ZERO($\mathcal{A}$) for which there is no $i < k < j$ with $k \in$ ZERO($\mathcal{A}$). Then, $(j - i) \leq |Q|^2$.*

The proof essentially establishes that the counter cannot go beyond $|Q|$ between two positions with successful zero-tests.

Proof. (Lemma 10) First observe that there are no $i < k < k' < j$ such that $q_k = q_{k'}$ and $n_k \leq n_{k'}$. Indeed, if it is the case since there is no successful zero-tests in $\langle q_{i+1}, n_{i+1}\rangle \cdots \langle q_k, n_k\rangle \cdots \langle q_{k'}, n_{k'}\rangle$ and $\mathcal{A}$ is deterministic we would obtain from $\langle q_{k'}, n_{k'}\rangle$ an infinite path with no zero-test, a contradiction with the existence of $\langle q_j, n_j\rangle$. Hence, if there are $i < k < k' < j$ such that $q_k = q_{k'}$, then $n_{k'} < n_k$. Now suppose that there is $i < k < j$ such that $n_k \geq |Q|$. We can extract a subsequence $\langle q_{i_0}, n_{i_0}\rangle \cdots \langle q_{i_s}, n_{i_s}\rangle$ from $\langle q_i, n_i\rangle \cdots \langle q_{n_k}, n_k\rangle$ such that $i_0 = i$, $i_s = k$ and for $0 \leq l < s$, $n_{i_{l+1}} = n_{i_l} + 1$. Consequently, there are l, l' such that $q_{i_l} = q_{i_{l'}}$ and $n_{i_l} < n_{i_{l'}}$, which leads to a contradiction from the above point. Hence, for $k \in \{i, \ldots, j\}$, $n_k \leq |Q| - 1$. Since $\mathcal{A}$ is deterministic, this implies that $(j - i) \leq |Q| \times |Q|$. □

Let us come back to the rest of the proof.

First, suppose that ZERO($\mathcal{A}$) is infinite. Let $i_0 < i_1 < i_2 < \ldots$ be the infinite sequence composed of elements from ZERO($\mathcal{A}$) ($i_0 = 0$). There are $l, l' \leq |Q|$ such that $\langle q_{i_l}, n_{i_l}\rangle = \langle q_{i_{l'}}, n_{i_{l'}}\rangle$. By Lemma 10, $i_{l'} \leq |Q| \times |Q|^2$. Take $K_1 = i_l$ and $K_2 = i_{l'} - i_l$.

Second, suppose that ZERO($\mathcal{A}$) is finite, say equal to $\{0, i_1, \ldots, i_l\}$ for some $l \leq |Q| - 1$ (if $l \geq |Q|$ we are in the first case). By Lemma 10, $i_l \leq (|Q| - 1) \times |Q|^2$. For all $i_l \leq k < k'$, if $q_k = q_{k'}$, then $n_k \leq n_{k'}$ (if it were not the case, there would eventually be another zero-test in the path starting with $\langle q_{i_l}, n_{i_l}\rangle$). Now there are $i_l \leq k < k' \leq i_l + |Q|$ such that $q_k = q_{k'}$ and consequently $n_k \leq n_{k'}$. Take $K_1 = k$, $K_2 = k' - k$ and $K_{inc} = n_{k'} - n_k$. We have $K_{inc} \leq |Q|$ because $k' - k \leq |Q|$. □

$\rho^\omega_{\mathcal{A}}$ 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_{inc} 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 state 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 state occurs in the prefix or in the loop.

When $K_{inc} = 0$ and $\mathcal{A}$ has an infinite run, $\rho^\omega_{\mathcal{A}}$ is exactly

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

It is then possible to apply a polynomial-space labelling algorithm à la CTL for model checking 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., [26]), model checking restricted to deterministic automata with $K_{inc} = 0$ is still PSpace-hard (see the proof of Proposition 7).

3.3. A PSpace *symbolic model-checking algorithm*

In this section, we provide decision procedures for solving MC(FO)* and MC(FO)$^\omega$ 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 $\mathcal{A}$.

We establish that whenever $K_{inc} > 0$, two positions with identical counter values are separated by a distance that is bounded by a polynomial in $|Q|$.

Let us introduce a few constants related to the one-counter automaton $\mathcal{A}$ when $K_{inc} > 0$.

- Let $\beta_1, \beta_2 \geq 0$ be the smallest natural numbers such that for every $i \in [K_1, K_1 + K_2 - 1]$, $n_i \in [n_{K_1} - \beta_1, n_{K_1} + \beta_2]$.
- Let γ be the greatest value amongst $\{n_0, \ldots, n_{K_1-1}\}$.
- $L = 1 + \gamma + \left\lceil \frac{\beta_1+\beta_2}{K_{inc}} \right\rceil$ where $\lceil \cdot \rceil$ denotes the ceiling function.

Intuitively, the constant LK_2 is greater than any distance between two positions belonging to the loop of the unique infinite run of $\mathcal{A}$ which have the same counter value. The next lemma formalizes this idea.

Lemma 11. *Suppose $K_{inc} > 0$ and let i, j be in $\mathbb{N}$.*

1. *If $i, j \geq K_1$ and $|i - j| \geq LK_2$, then $n_i \neq n_j$.*
2. *If $i < K_1$ and $j \geq K_1 + LK_2$, then $n_i \neq n_j$.*

Proof. (1) Assume that $i, j \geq K_1$ and $(i - j) \geq LK_2$. By using the Euclidean division, we introduce the following values: $r_i = (i - K_1) \bmod (K_2)$, $r_j = (j - K_1) \bmod (K_2)$ and the quotients a_i and a_j such that $i - K_1 = a_i K_2 + r_i$ and $j - K_1 = a_j K_2 + r_j$. Note that $0 \leq r_i, r_j < K_2$ and since $(i - j) \geq LK_2$, we necessarily have $a_i - a_j > L - 1$. Using the definition of the constants β_1 and β_2, we know that $n_{r_i+K_1}, n_{r_j+K_1} \in \{n_{K_1} - \beta_1, \ldots, n_{K_1} + \beta_2\}$. Since $i = a_i K_2 + r_i + K_1$ and $j = a_j K_2 + r_j + K_1$, by Lemma 9, we have $n_i = n_{r_i+K_1} + a_i K_{inc}$ and $n_j = n_{r_j+K_1} + a_j K_{inc}$. We obtain the following inequalities:

$$n_{K_1} - \beta_1 + a_i K_{inc} \leq n_i \leq n_{K_1} + \beta_2 + a_i K_{inc}$$
$$n_{K_1} - \beta_1 + a_j K_{inc} \leq n_j \leq n_{K_1} + \beta_2 + a_j K_{inc}$$

Consequently,

$$-\beta_1 - \beta_2 + (a_i - a_j)K_{inc} \leq n_i - n_j \leq \beta_1 + \beta_2 + (a_i - a_j)K_{inc}$$

Considering that $(a_i - a_j) > L - 1$ and using the definition of L, we obtain:

$$0 \leq \gamma K_{inc} < n_i - n_j$$

Hence $n_i \neq n_j$. The same proof can be done when we initially assume that $(j - i) \geq LK_2$.

(2) Let us assume that $i < K_1$ and $j \geq K_1 + LK_2$. Let a_j, r_j be defined as for the case (1). By using the same method, we obtain the following inequality:

$$n_{K_1} - \beta_1 + a_j K_{inc} \leq n_j \leq n_{K_1} + \beta_2 + a_j K_{inc}$$

Sine $\beta_2 \geq 0$, we have:

$$n_{K_1} - \beta_1 - \beta_2 + a_j K_{inc} - n_i \leq n_j - n_i$$

Moreover, since $j \geq K_1 + LK_2$, we get $a_j \geq L$. Consequently,

$$n_{K_1} - \beta_1 - \beta_2 + LK_{inc} - n_i \leq n_j - n_i$$

Using the definition of L, we get

$$n_{K_1} - \beta_1 - \beta_2 + (1 + \gamma)K_{inc} + \beta_1 + \beta_2 - n_i \leq n_{K_1} - \beta_1 - \beta_2 + LK_{inc} - n_i \leq n_j - n_i$$

Since $\gamma \times K_{inc} \geq n_i$, we get

$$n_{K_1} + K_{inc} \leq n_j - n_i$$

Consequently, $n_j > n_i$. □

Let us introduce the intermediate sets $P^1_\sim$ and $P^2_\sim$:

$$P^1_\sim = \{\langle i, j \rangle \in \{0, \ldots, K_1 + LK_2 - 1\}^2 \mid n_i = n_j \text{ and } i \leq j\}$$
$$P^2_\sim = \{\langle i, j \rangle \in \{0, \ldots, K_1 + LK_2 - 1\}^2 \mid n_i = n_j + LK_{inc} \text{ and } j < i\}$$

In the sequel, we write $P_\sim$ to denote the set $P^1_\sim \cup P^2_\sim$. We will now characterize the positions of $\rho^\omega_{\mathcal{A}}$ using the set $P_\sim$ and the constants L, K_1, K_2 and K_{inc} introduced before.

Lemma 12. *Suppose $K_{inc} > 0$ and let $j \geq i$ be in $\mathbb{N}$. Then, $n_i = n_j$ iff one the conditions below is true.*

1. $\langle i,j\rangle \in P^1_\sim$.
2. $i,j \geq K_1$, $\langle K_1 + (i-K_1) \bmod (LK_2), K_1 + (j-K_1) \bmod (LK_2)\rangle \in P_\sim$ *and* $(j-i) < LK_2$.

Proof. Let $i,j \in \mathbb{N}$ such that $i \leq j$. If (1) is satisfied, then by definition of $P^1_\sim$, we get $n_i = n_j$.

If (2) is satisfied, then let $r_i = (i-K_1) \bmod (LK_2)$, $r_j = (j-K_1) \bmod (LK_2)$ and a_i,a_j be quotients such that $i - K_1 = a_i LK_2 + r_i$ and $j - K_1 = a_j LK_2 + r_j$. By Lemma 9, we have $n_i = n_{r_i+K_1+a_iLK_2} = n_{r_i+K_1} + a_i LK_{inc}$ and $n_j = n_{r_j+K_1+a_jLK_2} = n_{r_j+K_1} + a_j LK_{inc}$. Since $(j-i) < LK_2$, we have $(a_j - a_i)LK_2 + (r_j - r_i) < LK_2$. Furthermore, we have by hypothesis $\langle K_1 + r_i, K_1 + r_j\rangle \in P_\sim$. We then distinguish two cases. First if $\langle K_1 + r_i, K_1 + r_j\rangle \in P^1_\sim$, we deduce that $r_i \leq r_j$ and consequently $a_i = a_j$. Hence $n_i = n_j$. Second if $\langle K_1 + r_i, K_1 + r_j\rangle \in P^2_\sim$, we deduce that $r_j < r_i$ and consequently $a_j = a_i + 1$. Hence $n_j = n_{r_j+K_1} + (a_i+1)LK_{inc}$ and since $n_{r_j+K_1} + LK_{inc} = n_{r_i+K_1}$, we obtain $n_i = n_j$.

We now suppose that $n_i = n_j$ and we perform the following case analysis.

- Assume that $i < K_1$ and $j < K_1$. By definition of $P^1_\sim$, we have $\langle i,j\rangle \in P^1_\sim$ and the condition (1) is therefore satisfied.
- Assume that $i,j \geq K_1$. By Lemma 11, we have $(j-i) < LK_2$ (otherwise we would have $n_i \neq n_j$). Let $r_i = (i-K_1) \bmod (LK_2)$, $r_j = (j-K_1) \bmod (LK_2)$ and a_i,a_j be quotients such that $i-K_1 = a_iLK_2+r_i$ and $j - K_1 = a_jLK_2 + r_j$. By Lemma 9, we have $n_i = n_{r_i+K_1+a_iLK_2} = n_{r_i+K_1} + a_iLK_{inc}$ and $n_j = n_{r_j+K_1+a_jLK_2} = n_{r_j+K_1} + a_jLK_{inc}$. We consider then two cases, according to the satisfaction of $a_i = a_j$.
  - Suppose $a_i = a_j$. Consequently, $n_{r_i+K_1} = n_{r_j+K_1}$ and since $i \leq j$, we have $r_i \leq r_j$. Condition (2) is therefore satisfied.
  - Suppose $a_i \neq a_j$. Since $(j-i) < LK_2$, necessarily, $a_j = a_i+1$. Hence $n_{r_j+K_1} = n_i - (a_i+1)LK_{inc}$, and since $(a_j - a_i)LK_2 + (r_j - r_i) < LK_2$, we also have $r_j < r_i$ from which we can conclude that condition (2) is again satisfied (we also have $n_{r_j+K_1} + LK_{inc} = n_{r_i+K_1}$).
- Assume that $i < K_1$ and $j \geq K_1$. By Lemma 11, we have $j < K_1 + LK_2$, and consequently $\langle i,j\rangle \in P^1_\sim$, hence condition (1) is satisfied.

All the values for i,j are covered by the above analysis. □

We show below how to reduce an instance of the model-checking problem (restricted to deterministic one-counter automata) to an instance of the problem mentioned in Theorem 4 by taking advantage of Lemma 12. First let us build finite words s,t over some finite alphabet Σ. By Lemma 6, we can assume that the formula ϕ belongs to the pure fragment of FO$(\sim, <, +1)$.

- $\Sigma = \{0, \ldots, K_1 + LK_2 - 1\}$.
- $s = \{0\} \cdot \{1\} \cdot \cdots \cdot \{K_1 - 1\}$.
- $t = \{K_1\} \cdot \{K_1 + 1\} \cdot \cdots \cdot \{K_1 + LK_2 - 1\}$.

Given a sentence ϕ in FO$(\sim, <, +1)$ let us define a sentence $T(\phi)$ in FO$^\Sigma(<, +1)$ according to the definition below:

- T is the identity for atomic formulae of the form $\mathtt{x} < \mathtt{y}$ and $\mathtt{x} = \mathtt{y} + 1$.
- T is homomorphic for Boolean connectives and first-order quantification.
- $T(\mathtt{x} \sim \mathtt{y}) = \big(\mathtt{x} \leq \mathtt{y} \wedge T_1(\mathtt{x},\mathtt{y})\big) \vee \big(\mathtt{y} \leq \mathtt{x} \wedge T_1(\mathtt{y},\mathtt{x})\big)$ and $T_1(\mathtt{x},\mathtt{y})$ is equal to

$$(\mathtt{y} - \mathtt{x}) < LK_2 \;\wedge\; \Big(\mathtt{x} < K_1 \Rightarrow \bigvee_{\langle I,J\rangle \in P^1_\sim} I(\mathtt{x}) \wedge J(\mathtt{y})\Big) \;\wedge\; \Big(\mathtt{x} \geq K_1 \Rightarrow \bigvee_{\langle I,J\rangle \in P_\sim} I(\mathtt{x}) \wedge J(\mathtt{y})\Big)$$

Observe that the formula of the form $(\mathtt{y}-\mathtt{x}) < LK_2$ is a shortcut for a formula in $\mathrm{FO}^Q(<,+1)$ of polynomial size in $|\mathcal{A}|$. For instance, when $\mathtt{x} \geq K_1 \wedge \mathtt{y} \geq K_1 \wedge \mathtt{y} > \mathtt{x}$ holds, $(\mathtt{y}-\mathtt{x}) < LK_2$ is equivalent to a formula with at most 3 variables, namely

$$\neg \bigwedge_{I=K_1}^{K_1+LK_2-1} \exists\, \mathtt{z}\ \mathtt{x} \leq \mathtt{z} < \mathtt{y} \wedge I(\mathtt{z}).$$

Lemma 13. *$\mathcal{A} \models^{\omega} \phi$ iff $s \cdot t^{\omega} \models T(\phi)$.*

Proof. The proof is by structural induction. We show that for each subformula ψ of ϕ and for each variable valuation u, $\mathcal{A} \models^{\omega}_{u} \psi$ iff $s \cdot t^{\omega} \models_u T(\psi)$. Since the formula ϕ belongs to the pure fragment of $\mathrm{FO}(\sim,<,+1)$ the only case that needs to be checked is for atomic formulae of the form $\mathtt{x} \sim \mathtt{y}$. Before giving the rest of the proof, we remark that since σ is an infinite word $s \cdot t^{\omega}$ built over the alphabet $\Sigma = \{0, \ldots, K_1 + LK_2 - 1\}$, for all $i \geq K_1$, we have $\sigma(i) = K_1 + (i - K_1) \bmod (LK_2)$. Let u be a variable valuation such that $u(\mathtt{x})$ and $u(\mathtt{y})$ are defined (if $u(\mathtt{x})$ or $u(\mathtt{y})$ is not defined, then it is easy to show that $\mathcal{A} \not\models^{\omega}_{u} \mathtt{x} \sim \mathtt{y}$ and that $s \cdot t^{\omega} \not\models_u T(\mathtt{x} \sim \mathtt{y})$).

First we suppose that $\mathcal{A} \models^{\omega}_{u} \mathtt{x} \sim \mathtt{y}$, this means that the unique infinite accepting run $\rho^{\omega}_{\mathcal{A}}$ of $\mathcal{A}$ satisfies $\rho^{\omega}_{\mathcal{A}} \models_u \mathtt{x} \sim \mathtt{y}$. Hence, we have $n_{u(\mathtt{x})} = n_{u(\mathtt{y})}$. We show that $s \cdot t^{\omega} \models_u T(\mathtt{x} \sim \mathtt{y})$. We suppose $u(\mathtt{x}) \leq u(\mathtt{y})$ (the proof is similar for the case $u(\mathtt{y}) \leq u(\mathtt{x})$). We proceed by a case analysis using Lemma 12 and the definition for $T(\mathtt{x} \sim \mathtt{y})$:

- If $u(\mathtt{x}) < K_1$, then necessarily $(u(\mathtt{y}) - u(\mathtt{x})) < LK_2$, hence $\sigma(u(\mathtt{x})) = u(\mathtt{x})$ and $\sigma(u(\mathtt{y})) = u(\mathtt{y})$, furthermore by Lemma 12 $\langle u(\mathtt{x}), u(\mathtt{y})\rangle \in P^1_{\sim}$, so we have $\sigma \models_u T(\mathtt{x} \sim \mathtt{y})$.
- If $u(\mathtt{x}) \geq K_1$, again we have $(u(\mathtt{y}) - u(\mathtt{x})) < LK_2$ and also $\sigma(u(\mathtt{x})) = K_1 + (i - u(\mathtt{x})) \bmod (LK_2)$ and $\sigma(u(\mathtt{y})) = K_1 + (i - u(\mathtt{y})) \bmod (LK_2)$. Using Lemma 12, we have $\langle \sigma(u(\mathtt{x})), \sigma(u(\mathtt{y}))\rangle \in P_{\sim}$, which implies $\sigma \models_u T(\mathtt{x} \sim \mathtt{y})$.

Now, let us suppose that $s \cdot t^{\omega} \models_u T(\mathtt{x} \sim \mathtt{y})$. Again, we perform a case analysis and we suppose that $u(\mathtt{x}) \leq u(\mathtt{y})$ (the proof for the case $u(\mathtt{y}) \leq u(\mathtt{x})$ is the same):

- If $u(\mathtt{x}) < K_1$ then $u(\mathtt{y}) < K_1 + LK_2$. Hence $\sigma(u(\mathtt{x})) = u(\mathtt{x})$ and $\sigma(u(\mathtt{y})) = u(\mathtt{y})$. Since $\langle u(\mathtt{x}), u(\mathtt{y})\rangle \in P^1_{\sim}$, we have $n_{u(\mathtt{x})} = n_{u(\mathtt{y})}$.
- If $u(\mathtt{x}) \geq K_1$ then $(u(\mathtt{y}) - u(\mathtt{x})) < LK_2$ and $\langle \sigma(u(\mathtt{x})), \sigma(u(\mathtt{y}))\rangle \in P_{\sim}$. Since $\sigma(u(\mathtt{x})) = K_1 + (i - u(\mathtt{x})) \bmod (LK_2)$ and $\sigma(u(\mathtt{y})) = K_1 + (i - u(\mathtt{y})) \bmod (LK_2)$, we obtain using Lemma 12 that $n_{u(\mathtt{x})} = n_{u(\mathtt{y})}$.

□

This allows us to characterize the complexity of model checking.

Theorem 14. $\mathrm{MC(FO)}^{\omega}$ *restricted to deterministic one-counter automata is* PSPACE*-complete.*

Proof. Let $\mathcal{A}$ be a one-counter automaton and ϕ be a pure formula in $\mathrm{FO}(\sim,<,+1)$. If either $\mathcal{A}$ has no infinite run or its infinite run is not accepting, then this can be checked in polynomial-time in $|\mathcal{A}|$. In that case $\mathcal{A} \models^{\omega} \phi$ does not hold. Moreover, observe that if $\mathcal{A}$ has no infinite run, then the length of the maximal finite run is in $\mathcal{O}(|Q|^3)$ by using arguments from Lemma 9.

In the case $\mathcal{A}$ has an infinite accepting run and $K_{inc} > 0$, as shown previously the prefixes s, t as well as the formula $T(\phi)$ can be computed in in polynomial time in $|\mathcal{A}| + |\phi|$. Moreover, by Theorem 4 [26], $s \cdot t^{\omega} \models T(\phi)$ can be checked in polynomial space in $|s| + |t| + |T(\phi)|$. In the case $K_{inc} = 0$, the prefixes s and t are defined as follows with $\Sigma = \{0, \ldots, K_1 + K_2 - 1\}$: $s = \{0\} \cdot \{1\} \cdots \{K_1 - 1\}$ and $t = \{K_1\} \cdot \{K_1 + 1\} \cdots \{K_1 + K_2 - 1\}$. The map $T(\cdot)$ is defined as previouly except that $T(\mathtt{x} \sim \mathtt{y}) = \bigvee_{\langle I,J\rangle \in P^3_{\sim}} I(\mathtt{x}) \wedge J(\mathtt{y})$ with $P^3_{\sim} = \{\langle i,j\rangle \in \{0, \ldots, K_1 + K_2 - 1\}^2 \mid n_i = n_j\}$.

Hence, $\mathrm{PureMC(FO)}^{\omega}$ is in polynomial space. Using the Purification Lemma 6, we deduce that $\mathrm{MC(FO)}^{\omega}$ is also in polynomial space. The PSPACE-hardness is a consequence of the PSPACE-hardness of $\mathrm{MC(LTL)}^{\omega}$ (since there is an obvious logspace translation from LTL^Q into $\mathrm{FO}^Q(\sim,<,+1)$). □

Theorem 15. $\mathrm{MC(FO)}^*$ *restricted to deterministic one-counter automata is* PSPACE*-complete.*

Proof. Let $\mathcal{A}$ be a one-counter automaton and ϕ be a pure formula in $\mathrm{FO}(\sim,<,+1)$. If $\mathcal{A}$ has an infinite run, then the finite words s and t are computed as in the infinitary case. We then need another intermediate set P_F which will characterize the positions of the unique run labelled with an accepting state:

$$P_F = \{i \in \{0,\ldots,K_1+LK_2-1\} \mid q_i \in F\}$$

The pure formula ϕ is then translated into

$$\exists\ \mathtt{x}_{end}\ (\bigvee_{I \in P_F} I(\mathtt{x}_{end})) \wedge T'(\phi),$$

where $T'(\phi)$ is defined as $T(\phi)$ for the infinitary case except that the clause for first-order quantification becomes $T'(\exists\ \mathtt{x}\ \psi) = \exists\ \mathtt{x}\ \mathtt{x} \leq \mathtt{x}_{end} \wedge T'(\psi)$ (relativization). As in the proof of Theorem 14, we get the PSPACE upper bound for $\mathrm{MC(FO)}^*$. In the case $\mathcal{A}$ has no infinite run, then the lengh K of the maximal finite run is in $\mathcal{O}(|Q|^3)$ and it can therefore be computed in polynomial-time. The prefixes s and t are defined as follows with $\Sigma = \{0,\ldots,K-1,\perp\}$: $s = \{0\}\cdot\{1\}\cdots\cdot\{K-1\}$ and $t = \{\perp\}$. The map $T(\cdot)$ is defined as previouly except that $T(\mathtt{x} \sim \mathtt{y}) = \bigvee_{\langle I,J\rangle \in P^4_\sim} I(\mathtt{x}) \wedge J(\mathtt{y})$ with $P^4_\sim = \{\langle i,j\rangle \in \{0,\ldots,K-1\}^2 \mid n_i = n_j\}$. The pure formula ϕ is translated into $\exists\ \mathtt{x}_{end}\ (\bigvee_{I \in P'_F} I(\mathtt{x}_{end})) \wedge \neg \perp (\mathtt{x}_{end}) \wedge T'(\phi)$, with $P'_F = \{i \in \{0,\ldots,K-1\} \mid q_i \in F\}$. The formula $T'(\phi)$ is defined as $T(\phi)$ for the infinitary case except that the clause for first-order quantification becomes $T'(\exists\ \mathtt{x}\ \psi) = \exists\ \mathtt{x}\ \mathtt{x} \leq \mathtt{x}_{end} \wedge T'(\psi)$. □

This improves the complexity bounds from [30]. Using the translation from $\mathrm{LTL}^{\downarrow}$ into $\mathrm{FO}(\sim,<,+1)$ from Lemma 2, we deduce the following corollary.

Corollary 16. $\mathrm{MC(LTL)}^*$ *and* $\mathrm{MC(LTL)}^\omega$ *are* PSPACE*-complete.*

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 by following a principle introduced in [11]. Undecidability is preserved even in presence of a unique register. This is quite surprising since $*$-SAT-$\mathrm{LTL}^{\downarrow}$ restricted to one register and satisfiability for $\mathrm{FO}_2(\sim,<,+1)$ are decidable [7, 8].

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 [8]. 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 [8] so that runs with incrementing errors are excluded. More precisely, in the reduction in [8], 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 17. $\mathrm{MC(LTL)}^*_1$ *restricted to formulae using only the temporal operators* `X` *and* `F` *is* Σ^0_1*-complete.*

Proof. The Σ^0_1 upper bound is by an easy verification since the existence of a finite run (encoded in $\mathbb{N}$) verifying an $\mathrm{LTL}^{\downarrow,Q}_1$ formula (encoded in first-order arithmetic) can be encoded by a Σ^0_1 formula. So, let us reduce the halting problem for two-counter automata to $\mathrm{MC(LTL)}^*_1$ restricted to $\{\mathtt{X},\mathtt{F}\}$. Let $\mathcal{A} = \langle Q, q_I, \delta, F\rangle$ be a two-counter automaton: the set of instructions L is $\{\mathtt{inc},\mathtt{dec},\mathtt{ifzero}\} \times \{1,2\}$. Without any loss of generality, we can assume that all the instructions from q_I are incrementations. We build a one-counter automaton $\mathcal{B} = \langle Q', q'_I, \delta', F'\rangle$ and a sentence ϕ in $\mathrm{LTL}^{\downarrow,Q'}_1$ such that $\mathcal{A}$ reaches an accepting state iff $\mathcal{B} \models^* \phi$.

For each run in $\mathcal{A}$ of the form

$$\begin{pmatrix} q_I \\ c_1^0 = 0 \\ c_2^0 = 0 \end{pmatrix} \xrightarrow{\texttt{inst}^0} \begin{pmatrix} q^1 \\ c_1^1 \\ c_2^1 \end{pmatrix} \xrightarrow{\texttt{inst}^1} \ldots \begin{pmatrix} q^N \\ c_1^N \\ c_2^N \end{pmatrix}$$

where the $\texttt{inst}^{\texttt{i}}$'s are instructions, we associate a run in $\mathcal{B}$ of the form below:

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

where $\xrightarrow{\star}$ hides steps for updating the counter according to the constraints described below. The set of states Q' will contain the set of transitions δ from $\mathcal{A}$.

We first define the one-counter automaton $\mathcal{B} = \langle Q', q_I', \delta', F' \rangle$. In order to ease the presentation, the construction of $\mathcal{B}$ is mainly provided graphically.

- Q' is the following set of states:

$$\begin{array}{rl} Q' = & \delta \uplus \{q_I\} \uplus \{i_0\} \\ & \uplus\{i_t^{last}, i_t^{\neg last} \mid t = \langle q, \texttt{inc}, c, q' \rangle \in \delta\} \\ & \uplus\{d_t^{last}, d_t^{\neg last} \mid t = \langle q, \texttt{dec}, c, q' \rangle \in \delta\} \\ & \uplus\{z_t^{down} \mid t = \langle q, \texttt{ifzero}, c, q' \rangle \in \delta\} \\ & \uplus\{z_q \mid q \in Q\} \uplus Q_{aux} \end{array}$$

  where Q_{aux} is a set of auxiliary states that we do not specify (but which can be identified as the states with no label in Figures 4, 5 and 6),
- F' is the set of states $\{z_q \mid q \in F\}$.
- The transition relation δ' is the smallest transition relation satisfying the conditions below:
  - The transitions in Figure 3 belong to δ'.
  - For each incrementation transition $t = \langle q_I, \texttt{inc}, c, q \rangle$, the transitions in Figure 4 belong to δ'.
  - For each decrementation transition $t = \langle q_I, \texttt{dec}, c, q \rangle$, the transitions in Figure 5 belong to δ'.
  - For each zero-test transition $t = \langle q_I, \texttt{ifzero}, c, q \rangle$, the transitions in Figure 6 belong to δ'.

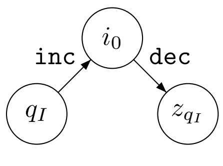


Figure 3: Initial transitions in δ'

In runs of $\mathcal{B}$, we are only interested in configurations whose state belongs to δ. The structure of $\mathcal{B}$ ensures that the sequence of transitions in $\mathcal{A}$ is valid assuming that we ignore the intermediate (auxiliary or busy) configurations

Before defining the formula ϕ, let us introduce a few intermediate formulae that allow us to check whether the current configuration has a state belonging to a specific set. For each counter $i \in \{1, 2\}$, we define the formulae below:

- I_i is the disjunction of i_0 with all the transitions t that increment the counter i in $\mathcal{A}$, hence $I_i = i_0 \vee \bigvee_{\{t \in \delta \mid t = \langle q, \texttt{inc}, i, q' \rangle\}} t$.

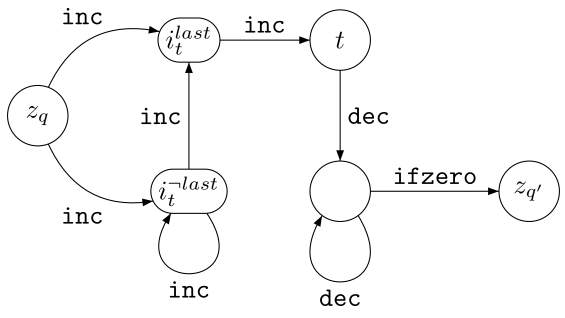


Figure 4: Gadget in $\mathcal{B}$ for encoding an incrementation from $\mathcal{A}$

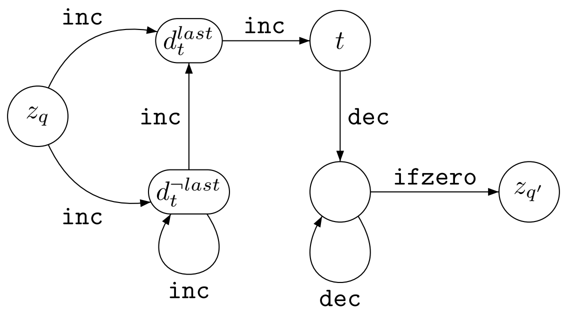


Figure 5: Gadget in $\mathcal{B}$ for encoding a decrementation from $\mathcal{A}$

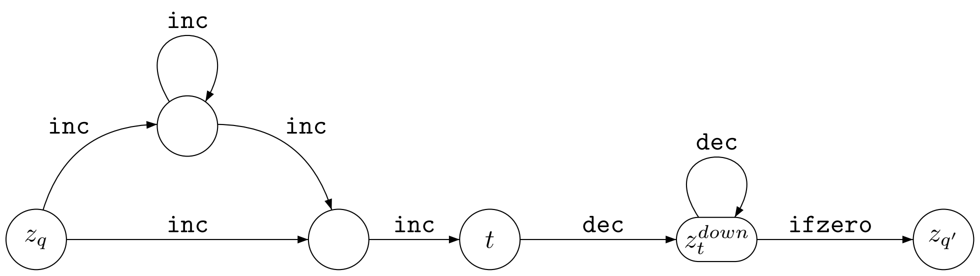


Figure 6: Gadget in $\mathcal{B}$ for encoding a zero-test from $\mathcal{A}$

- D_i is the disjunction of i_0 with all the transitions t that decrement the counter i in $\mathcal{A}$, hence $D_i = i_0 \vee \bigvee_{\{t \in \delta | t = \langle q, \mathtt{dec}, i, q' \rangle\}} t$.
- I_i^{last} is the disjunction of all states of the form i_t^{last} where t is a transition that increments the counter i, hence $I_i^{last} = \bigvee_{\{t \in \delta | t = \langle q, \mathtt{inc}, i, q' \rangle\}} i_t^{last}$.
- $I_i^{\neg last}$ is the disjunction of all states of the form $i_t^{\neg last}$ where t is a transition that increments the counter i, hence $I_i^{\neg last} = \bigvee_{\{t \in \delta | t = \langle q, \mathtt{inc}, i, q' \rangle\}} i_t^{\neg last}$.
- D_i^{last} is the disjunction of all states of the form d_t^{last} where t is a transition that decrements the counter i, hence $D_i^{last} = \bigvee_{\{t \in \delta | t = \langle q, \mathtt{dec}, i, q' \rangle\}} d_t^{last}$.
- $D_i^{\neg last}$ is the disjunction of all states of the form $d_t^{\neg last}$ where t is a transition that decrements the counter i, hence $D_i^{\neg last} = \bigvee_{\{t \in \delta | t = \langle q, \mathtt{dec}, i, q' \rangle\}} d_t^{\neg last}$.
- Z_i is the disjunction of all the transitions t that test to zero the counter i in $\mathcal{A}$, hence $Z_i = \bigvee_{\{t \in \delta | t = \langle q, \mathtt{ifzero}, i, q' \rangle\}} t$.
- Z_i^{down} is the disjunction of the states of the form z_t^{down} where t is a zero-test on the counter i, hence $Z_t^{down} = \bigvee_{\{t \in \delta | t = \langle q, \mathtt{ifzero}, i, q' \rangle\}} z_t^{down}$.

In order to define ϕ, we take advantage of the structure of $\mathcal{B}$ so that to match runs of $\mathcal{B}$ with runs of $\mathcal{A}$. A crucial idea consists in associating to each action on one of the two counters, a natural number so that an incrementation gets a new value. Moreover, we require that the natural number associated to an incrementation is obtained by increasing by one the natural number associated to the previous incrementation. We satisfy a similar property for the natural numbers associated to decrementations except that these values should not exceed the value associated to the previous incrementation. In this way, we guarantee that there are no more decrementations than incrementations. In order to simulate the zero-test, we reach a value above all the values that have been used so far. Then we check that for all the smaller values that are associated to an incrementation, it is also associated to a decrementation (for the same counter).

In the following formulae, we use $\mathtt{G}^+$ and $\mathtt{F}^+$ to represent the formulae $\mathtt{XG}$ and $\mathtt{XF}$, respectively. We also omit the subscript "1" in $\downarrow_1$ and $\uparrow_1$ because we assume that we always use the same register. For each counter $i \in \{1, 2\}$, we define the following formulae:

(i) After each configuration satisfying I_i, there is no strict future configuration satisfying I_i with the same data value:
$$\mathtt{G}\big(I_i \Rightarrow \downarrow \mathtt{G}^+(I_i \Rightarrow \neg \uparrow)\big)$$

(ii) After each configuration satisfying D_i, there is no strict future configuration satisfying D_i with the same data value:
$$\mathtt{G}\big(D_i \Rightarrow \downarrow \mathtt{G}^+(D_i \Rightarrow \neg \uparrow)\big)$$

(iii) After each configuration satisfying D_i, there is no strict future configuration satisfying I_i with the same data value:
$$\mathtt{G}\big(D_i \Rightarrow \downarrow \mathtt{G}^+(I_i \Rightarrow \neg \uparrow)\big)$$

(iv) When a new data value is needed for an incrementation of the counter i, the chosen value is exactly the next value after the greatest value used so far for an incrementation of the counter i:
$$\begin{array}{c}\mathtt{G}\big(I_i \Rightarrow (\downarrow \mathtt{F}(I_i^{\neg last} \wedge \uparrow) \Rightarrow \downarrow \mathtt{F}(I_i^{last} \wedge \uparrow))\big) \\ \wedge \mathtt{G}\big((I_i^{last} \vee I_i^{\neg last}) \Rightarrow \downarrow \mathtt{G}^+(I_i \Rightarrow \neg \uparrow)\big)\end{array}$$

(v) When a new data value is needed for a decrementation of the counter i, the chosen value is exactly the next value after the greatest value used so far for a decrementation of the counter i:
$$\begin{array}{c}\mathtt{G}\big(D_i \Rightarrow (\downarrow \mathtt{F}(D_i^{\neg last} \wedge \uparrow) \Rightarrow \downarrow \mathtt{F}(D_i^{last} \wedge \uparrow))\big) \\ \wedge \mathtt{G}\big((D_i^{last} \vee D_i^{\neg last}) \Rightarrow \downarrow \mathtt{G}^+(D_i \Rightarrow \neg \uparrow)\big)\end{array}$$

(vi) The data value associated to a decrementation of the counter i is never strictly greater than the greatest previous value used in incrementations of the counter i:

$$\begin{array}{l}\mathtt{G}\big(I_i \Rightarrow (\downarrow \mathtt{F}(D_i^{\neg last} \wedge \uparrow) \Rightarrow \downarrow \mathtt{F}(I_i^{last} \wedge \uparrow))\big)\\ \wedge \mathtt{G}\big(I_i \Rightarrow (\downarrow \mathtt{F}(D_i^{last} \wedge \uparrow) \Rightarrow \downarrow \mathtt{F}(I_i^{last} \wedge \uparrow))\big)\\ \wedge \mathtt{G}\big(D_i^{\neg last} \Rightarrow \downarrow \mathtt{G}^+(I_i^{last} \Rightarrow \neg \uparrow)\big)\end{array}$$

(vii) For each configuration satisfying Z_i, the associated data value is always strictly greater than the greatest previous value used in incrementations of the counter i :

$$\mathtt{G}\big(I_i \Rightarrow \downarrow \mathtt{G}(Z_i \Rightarrow \neg \uparrow)\big)$$

(viii) When the automaton $\mathcal{B}$ is in the decrementing slope to encode a zero-test in $\mathcal{A}$, which means when the formula Z_i^{down} is satisfied, and when a data value already used for an incrementation is met, then the same data value is used previously for a decrementation in $\mathcal{B}$:

$$\neg \mathtt{F}\big(I_i \wedge \downarrow \mathtt{F}(Z_i^{down} \wedge \uparrow) \wedge \neg \downarrow \mathtt{F}(\uparrow \wedge D_i)\big) \wedge \neg \mathtt{F}\big(Z_i^{down} \wedge \downarrow \mathtt{F}(D_i \wedge \uparrow)\big)$$

Let us recall the book-keeping of the values.

- A new value used for an incrementation is always one plus the greatest value used so far for an incrementation (see (iv)). The first counter value for an incrementation is 2.
- A new value used for decrementation is always 1 + the greatest value used so far for a decrementation (see (v)), and is always smaller or equal to the greatest value used so for a incrementation (see (vi)). The first counter value for a decrementation is 2.
- Zero-tests consist in:
  (1) going to a value strictly greater than any value used so far for incrementations (encoded in $\mathcal{B}$ and see (vii)),
  (2) then decrementing the counter to zero (encoded in $\mathcal{B}$) and whenever a value is met that is used for an incrementation, check that a corresponding decrementation has occured before (see (viii)).

In order to ease the comprehension, we explain why the rule (vi) ensures that the value associated to a decrementation of the counter i is never strictly greater than the value used for the last incrementation of the same counter i. First, we assume that the rules (i)–(vi) are satisfied and *ad absurdum* we suppose that the value used for a decrementation is strictly greater than the value used for the last incrementation of the counter i. If this value is greater of exactly one unit, then we are in the case of the second line of the formula given by the rule (vi). Hence, there must exist an incrementation with the same value as the one for the decrementation, and this incrementation necessarily happens between the first considered incrementation and the decrementation, according to the rules (i)–(iii). This leads to a contradiction because the first considered incrementation is not the last one. Secondly, suppose that the value associated to the decrementation is greater of k units with $k > 1$. We are in the case of the first line of the formula given by the rule (vi), and consequently there exists an incrementation after the first considered incrementation which has an associated value greater of one unit. The last line of the formula of the rule (vi) ensures that this incrementation occurs necessarily before the decrementation, which leads again to a contradiction, because the first considered incrementation cannot be the last one.

Figure 7 gives an example of the beginning of a run of $\mathcal{B}$ which respects the rules (i)–(viii) and that encodes the following sequence of instructions $(\mathtt{inc}, 1), (\mathtt{inc}, 1), (\mathtt{dec}, 1), (\mathtt{dec}, 1), (\mathtt{ifzero}, 1)$. In the decreasing part after the position labeled by Z_1, each value used in a previous incrementation can be matched with a value associated to a decrementation. The formula ϕ is defined as the conjunction of (i)–(viii) plus (ix) that specifies that a state in F' is reached. Now consider any run of $\mathcal{B}$ which satisfies (i)–(viii). For any counter $c \in \{1, 2\}$, we can define its value as the number of I_t letters with t of the form $\langle q, \mathtt{inc}, c, q' \rangle$ for which a

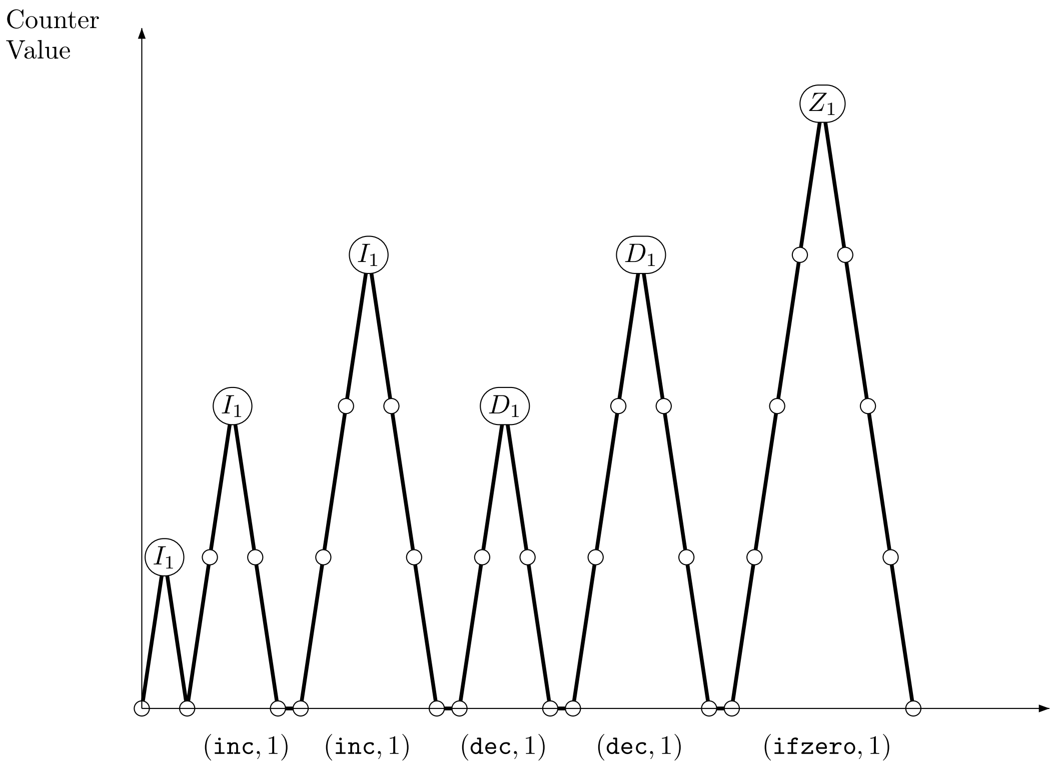


Figure 7: Run for $\mathcal{B}$ satisfying the rules (i)–(viii)

later letter $\langle q_1, \texttt{dec}, c, q_1' \rangle$ with the same value of the counter $\mathcal{B}$ has not yet occurred. We will now prove that $\mathcal{B} \models^* \phi$ if and only if the automaton $\mathcal{A}$ has an accepting run.

Let $\rho = \langle p_0, 0\rangle \xrightarrow{a_0} \langle p_1, n_1\rangle \xrightarrow{a_1} \langle p_2, n_2\rangle \ldots \langle q_m, n_m\rangle$ be a finite run of $\mathcal{B}$ satisfying the rules (i)–(viii) and such that $p_0 = q_I$ and $p_m = q$ for some $q \in Q$. We consider the sequence of indices $i_1, \ldots, i_k \in \{0, \ldots, m\}$ such that for all $j \in \{1, \ldots, m\}$, $p_{i_j} \in \delta$ and such that there is no $i \in \{1, \ldots, m\}$ with $p_i \in \delta$ and $i \notin \{i_1, \ldots, i_k\}$. We will show that the sequence $p_{i_1} p_{i_2} \ldots p_{i_k}$ induces a run of $\mathcal{A}$. This means that there exist k configurations $c_1, c_2, \ldots c_k \in Q \times \mathbb{N}^2$ such that $\langle q_I, 0, 0\rangle \xrightarrow{p_{i_1}} c_1 \xrightarrow{p_{i_2}} c_2 \ldots \xrightarrow{p_{i_k}} c_k$ is a run of $\mathcal{A}$.

The proof is by induction on k. If $k = 1$, then by construction of the automaton $\mathcal{B}$, there exist $i \in \{1, 2\}$ and $q' \in Q$ such that $p_{i_1} = \langle q_0, \texttt{inc}, i, q'\rangle$. This is simply due to the fact that we have assumed that any instruction starting in q_I is an incrementation. Since it is always possible to perform an incrementation, there is a configuration $c_1 \in Q \times \mathbb{N}^2$ such that $\langle q_I, 0, 0\rangle \xrightarrow{p_{i_1}} c_1$.

We suppose that the property is true for k and we show that it also holds for $k + 1$.

First, let us write down the properties verified by the sequence

$$\langle p_{i_0}, n_{i_0}\rangle, \ldots, \langle p_{i_k}, n_{i_k}\rangle$$

made of configurations of $\mathcal{B}$. For each counter $i \in \{1, 2\}$, we write Inc_i to denote the set $\{j \in \{1, \ldots, k\} \mid p_{i_j}$ is of the form $\langle q, \texttt{inc}, i, q'\rangle\}$ and Dec_i to denote the set $\{j \in \{1, \ldots, k\} \mid p_{i_j}$ is of the form $\langle q, \texttt{dec}, i, q'\rangle\}$. Let i be one of the counters in $\{1, 2\}$. The rule (i) ensures that for every $j \in Inc_i$, $n_{i_j} > 1$, and for all $j, \ell \in Inc_i$, $n_{i_j} \neq n_{i_\ell}$. This is because i_0 is a disjunct of I_i, the counter value in the state i_0 is always 1 and for all $j \in Inc_i$, p_{i_j} satisfies I_i. Furthermore the rule (iv) implies that for all $j, \ell \in Inc_i$ such that $j < \ell$, if there is no $j' \in Inc_i$ such that $j < j' < \ell$, then necessarily $n_{i_\ell} = n_{i_j} + 1$. Moreover, if j is the smallest index of Inc_i then $n_{i_j} = 2$. In fact, if j is the smallest index of Inc_i, then n_{i_j} is greater or equal to 2 (because the integer value in i_0 is always 1). If n_{i_j} is strictly greater than 2, then the run of $\mathcal{B}$ should reach a state that satisfies I_i^{last} or $I_i^{\neg last}$ with a value equal to 2, but since j is the smallest index of Inc_i, the rule (iv) would not be satisfied. To show the other property about the indices in Inc_i, this can be done by induction on the indices of Inc_i by using again the rule (iv). Similarly, it can be proved that the set Dec_i verifies the same properties. Hence, $\{n_{i_j} \mid j \in Inc_i\} = \{2, \ldots, |Inc_i| + 1\}$ and $\{n_{i_j} \mid j \in Dec_i\} = \{2, \ldots, |Dec_i| + 1\}$. Finally, the rule (vi) guarantees that for every $j \in Dec_i$, there is $\ell \in Inc_i$ such that $i_\ell \leq i_j$ and $n_{i_j} \leq n_{i_\ell}$. By combining these different properties, we deduce that $|Dec_i| \leq |Inc_i|$.

We suppose that $p_{i_k} = \langle q, a', i', q'\rangle$. By construction of $\mathcal{B}$, we have $p_{i_{k+1}} = \langle q', a, i, q''\rangle$. If a is equal to $\texttt{inc}$, then the property is satisfied because an incrementation can always be performed (unlike decrementations and zero-tests). Now, suppose that $a = \texttt{dec}$. The transition $p_{i_{k+1}} = \langle q', a, i, q''\rangle$ is not firable only if $|Dec_i| = |Inc_i|$ (the number of incrementations is equal to the number of decrementations). This situation cannot occur since ρ satisfies the rules (i)—(viii), and therefore $n_{i_{k+1}} = n_{i_H} + 1$ where H is the greatest index of $|Dec_i|$ and there exists $h \in |Inc_i|$ such that $i_h \leq i_{k+1}$ and $n_{i_{k+1}} \leq n_{i_h}$. Hence, if $|Dec_i| = |Inc_i|$, according to the previous properties, we would have that there exists $j \in Dec_i$ such that $n_{i_h} = n_{i_j}$ and consequently $n_{i_H} + 1 \leq n_{i_j}$ which leads to a contradiction (by definition of H). Now, suppose that $a = \texttt{ifzero}$. The transition $p_{i_{k+1}}$ is not firable only if $|Inc_i| > |Dec_i|$ (there are more incrementations than decrementations). This situation cannot occur since ρ satisfies the rules (i)–(viii) and according to the rule (vii) and to the properties verified by Inc_i, for all $j \in Inc_i$, $n_{i_j} < n_{i_{k+1}}$. After the i_{k+1}th configuration, the $n_{i_{k+1}}$ next configurations contain a state that satisfies Z_i^{down}. If $|Inc_i| > |Dec_i|$, then this means that there is an index $h \in Inc_i$ such that for all $j \in Dec_i$, $n_{i_j} < n_{i_h}$ and there exists also $l \in \{i_{k+1}, \ldots, i_{k+1} + n_{i_{k+1}}\}$ such that p_l satisfies Z_i^{down} and $n_l = n_h$, which is in contradiction with the rule (viii).

We conclude that if ρ is a finite run of $\mathcal{B}$ satisfying the rules (i)–(viii) and visiting a state z_q in F' then there is a corresponding run in the two-counter automaton $\mathcal{A}$ starting from the initial configuration $\langle q_I, 0, 0\rangle$ and visiting the accepting state q.

Now, we consider a run of $\mathcal{A}$ of the form $\langle q_I, 0, 0\rangle \xrightarrow{t_0} c_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{h-1}} c_h$. We show how to build a run of the one-counter automaton $\mathcal{B}$, $\langle p_0, 0\rangle \to \langle p_1, n_1\rangle \to \ldots \to \langle p_m, n_m\rangle$ with $p_0 = q_I$ and $p_m = z_q$ for some $q \in Q$. We introduce similar notations as in the converse case. For such a run, we consider the sequence of indices $i_1, \ldots, i_k \in \{0, \ldots, m\}$ such that for all $j \in \{1, \ldots, m\}$, $p_{i_j} \in \delta$ and such that there is no $i \in \{1, \ldots, m\}$ verifying $p_i \in \delta$ and $i \notin \{i_1, \ldots, i_k\}$. For each counter $i \in \{1, 2\}$, we write Inc_i to denote the set $\{j \in \{0, \ldots, k\} \mid$

p_{i_j} is of the form $\langle q, \texttt{inc}, i, q' \rangle\}$ and Dec_i to denote the set $\{j \in \{0, \ldots, k\} \mid p_{i_j}$ is of the form $\langle q, \texttt{dec}, i, q' \rangle\}$. Finally, we define the set $Zero_i = \{j \in \{0, \ldots, k\} \mid p_{i_j}$ is of the form $\langle q, \texttt{ifzero}, i, q' \rangle\}$. We build a run ρ of $\mathcal{B}$ such that the following properties are verified :

(a) $k = h$ and for all $j \in \{1, \ldots, k\}$, $p_{i_j} = t_{j-1}$,
(b) if j is the smallest index of Inc_i, then $n_{i_j} = 2$,
(c) if j is the smallest index of Dec_i, then $n_{i_j} = 2$,
(d) for all $j, \ell \in Inc_i$ such that $j < \ell$, if there is no $j' \in Inc_i$ such that $j < j' < \ell$, then $n_{i_\ell} = n_{i_j} + 1$,
(e) for all $j, \ell \in Dec_i$ such that $j < \ell$, if there is no $j' \in Inc_i$ such that $j < j' < \ell$,then $n_{i_\ell} = n_{i_j} + 1$,
(f) for all $j \in Dec_i$, there exists $\ell \in Inc_i$ such that $i_\ell < i_j$ and $n_{i_j} \leq n_{i_\ell}$,
(g) for all $j \in Zero_i$, and for all $\ell \in Inc_i$ such that $i_\ell < i_j$, we have $n_{i_\ell} < n_{i_j}$ and there is $m \in Inc_i$ such that $i_m < i_j$ and $n_{i_j} = n_{i_m} + 1$.

By construction of $\mathcal{B}$, it is possible to build a run ρ of $\mathcal{B}$ that satisfies the properties (a)–(g).

Now, we suppose that ρ is a run of $\mathcal{B}$ verifying these properties and it remains to check that ρ satisfies the rules (i)–(viii). First, we consider the rules (i)–(ii). These two rules are satisfied because all the elements of Inc_i and of Dec_i are built with distinct values for incrementations and decrementations. The rule (iii) is satisfied because of the properties (e) and (f). The rule (iv) is satisfied, because if the run is in a position i_j with $j \in Inc_i$ and if there exists a position ℓ in the future which satisfies $I_i^{\neg last}$, then there exists a position $i_{j'}$ such that $\ell < i_{j'}$ with $j' \in Inc_i$ and $n_{i_{j'}} > n_{i_j} + 1$ (by construction of $\mathcal{B}$ and by (d)). Moreover, the definition of $\mathcal{B}$ implies there exists a position h such that $i_j < h < \ell$, h satisfies I_i^{last}, $n_h = n_{i_j}$, q_{h+1} satisfies I_i and $n_{h+1} = n_{i_j} + 1$. Similar arguments are used to establish that the rule (v) is satisfied by using (c) and (e). The rule (vi) is satisfied because of the property (f). Finally the rules (vii)–(viii) are satisfied by using (g) and the properties about the sets Inc_i and Dec_i. Hence if there is a run of $\mathcal{A}$ leaving from $\langle q_I, 0, 0 \rangle$ and visiting a state q in F, we can build a finite run ρ of $\mathcal{B}$ such that $\rho \models \phi$.

Furthermore the formula ϕ uses only the temporal operators $\texttt{X}$ and $\texttt{F}$ (the operator $\texttt{G}$ can be easily obtained from $\texttt{F}$). □

Theorem 18. $\mathrm{MC}(\mathrm{LTL})_1^\omega$ *restricted to* $\{\texttt{X}, \texttt{F}\}$ *is* Σ_1^1*-complete.*

The proof is similar to the proof of Theorem 17 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 [20]. 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 19. $\mathrm{PureMC}(\mathrm{LTL})_1^*$ *restricted to* $\{\texttt{X}, \texttt{F}\}$ *is* Σ_1^0*-complete.* $\mathrm{PureMC}(\mathrm{LTL})_1^\omega$ *restricted to* $\{\texttt{X}, \texttt{F}\}$ *is* Σ_1^1*-complete.*

This refines results stated in [30].

Using Theorem 3.2(a) in [8], we can obtain the following corollary by a direct analysis of the formulae involved in the proof of Theorem 17 (every temporal operator is prefixed by a freeze operator or can occur equivalently in such a form).

Corollary 20. $\mathrm{MC}(\mathrm{FO})_2^*$ *[resp.* $\mathrm{MC}(\mathrm{FO})_2^\omega$*] without the predicate* $+1$ *is* Σ_1^0*-complete [resp.* Σ_1^1*-complete] and* $\mathrm{PureMC}(\mathrm{FO})_4^*$ *[resp.* $\mathrm{PureMC}(\mathrm{FO})_4^\omega$*] is* Σ_1^0*-complete [resp.* Σ_1^1*-complete].*

The absence of the predicate $+1$ in the above corollary is due to the fact that in the proof of Theorem 17, $\texttt{X}$ occurs only to encode $\texttt{F}^+$ and $\texttt{G}^+$. The above-mentioned undecidability is true even if we restrict ourselves to one-counter automata for which there are no transitions with identical instructions leaving from the same state. A one-counter automaton $\mathcal{A}$ is *weakly deterministic* whenever for every state 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 21. PureMC(LTL)$_1^*$ *[resp.* PureMC(LTL)$_1^\omega$*] restricted to weakly deterministic one-counter automata is* Σ_1^0*-complete [resp.* Σ_1^1*-complete].*

Proof. In the proof of the Purification Lemma, weak determinisn of the one-counter automata is preserved. It is sufficient to show that given a one-counter automaton $\mathcal{A}$ and a sentence ϕ in LTL$^{\downarrow,Q}$, one can compute a weakly deterministic automaton $\mathcal{A}'$ and ϕ' in LTL$^{\downarrow,Q'}$ ($Q \subseteq Q'$) such that $\mathcal{A} \models^* \phi$ [resp. $\mathcal{A} \models^\omega \phi$] iff $\mathcal{A}' \models^* \phi'$ [resp. $\mathcal{A}' \models^\omega \phi'$].

Figure 8 illustrates with examples how transitions from a state with identical instructions can be transformed so that to obtain a weakly deterministic automaton. In Figure 8, we have omitted the transitions labelled by a zero-test or a decrementation when they are never fired. This can be easily generalized to all the transitions of $\mathcal{A}$. The formula ϕ' is defined as $\mathrm{T}(\phi)$ with the map T that is homomorphic for Boolean operators and $\downarrow_r$, and its restriction to atomic formulae is identity. It remains to define the map for the temporal operators, which corresponds to perform a relativization:

- $\mathrm{T}(\phi_1 \mathtt{U} \phi_2) = \big((\bigvee_{q \in Q} q) \Rightarrow \mathrm{T}(\phi_1)\big)\mathtt{U}\big(\bigvee_{q \in Q} q \wedge \mathrm{T}(\phi_2)\big)$,
- $\mathrm{T}(\mathtt{X}\psi) = \mathtt{X}\big((\neg \bigvee_{q \in Q} q)\ \mathtt{U}\ (\bigvee_{q \in Q} q \wedge \mathrm{T}(\psi))\big)$.

It can be easily proved that $\mathcal{A}'$ and ϕ' satisfy the desired properties. □

5. Conclusion

In the paper, we have studied complexity issues related to the model-checking problem for LTL with registers over one-counter automata. Our results are quite different from those for satisfiability. We have shown that model checking LTL$^\downarrow$ restricted to the operators $\{\mathtt{X}, \mathtt{F}\}$ and FO$_2(\sim, <, +1)$ over one-counter automata is undecidable, which contrasts with the decidability of many verification problems for one-counter automata [27, 28, 29] and with the results in [7, 8]. For instance, we have shown that model checking nondeterministic one-counter automata over LTL$^\downarrow$ restricted to a unique register and without alphabet [resp. FO$_2(\sim, <, +1)$] is already Σ_1^1-complete in the infinitary case. On the decidability side, the PSPACE upper bound for model checking LTL$^\downarrow$ and FO$(\sim, <, +1)$ over deterministic one-counter automata in the infinitary and finitary cases is established by using in an essential way [26] (and simplifying the proofs from [30]). In particular, we have established that the runs of deterministic one-counter automata admit descriptions that require polynomial size only. Hence, our results essentially deal with LTL with registers but they can be also understood as a contribution to refine the decidability border for problems on one-counter automata.

Viewing runs as data words is an idea that can be pushed further. Indeed, our results pave the way for model checking memoryful (linear-time) logics (possibly extended to multicounters) over other classes of operational models that are known to admit powerful techniques for solving verification tasks. For instance, the reachability relation is known to be Presburger-definable for reversal-bounded counter automata [32]. Nevertheless, model checking LTL$^\downarrow$ over this class of counter machines has been recently shown undecidable [33]; other subclasses of counter machines for which the reachability problem is decidable have been considered in this recent work.

Acknowledgement: We would like to thank Philippe Schnoebelen for suggesting simplifications in the proofs of Lemma 5 and Proposition 7 and Luc Segoufin for fruitful discussions that lead us to improve significantly the results from [30].

[1] M. Minsky, Computation, Finite and Infinite Machines, Prentice Hall, 1967.
[2] R. Alur, D. Dill, A theory of timed automata, Theoretical Computer Science 126 (1994) 183–235.
[3] H. Björklund, M. Bojanczyk, Shuffle expressions and words with nested data, in: MFCS'07, Vol. 4708 of Lecture Notes in Computer Science, Springer, 2007, pp. 750–761.
[4] M. Bojańczyk, A. Muscholl, T. Schwentick, L. Segoufin, Two-variable logic on data trees and XML reasoning, J. ACM 56 (3).
[5] H. Björklund, M. Bojańczyk, Bounded depth data trees, in: ICALP'07, Vol. 4596 of Lecture Notes in Computer Science, Springer, 2007, pp. 862–874.
[6] M. Jurdziński, R. Lazić, Alternation-free modal mu-calculus for data trees, in: LICS'07, IEEE, 2007, pp. 131–140.

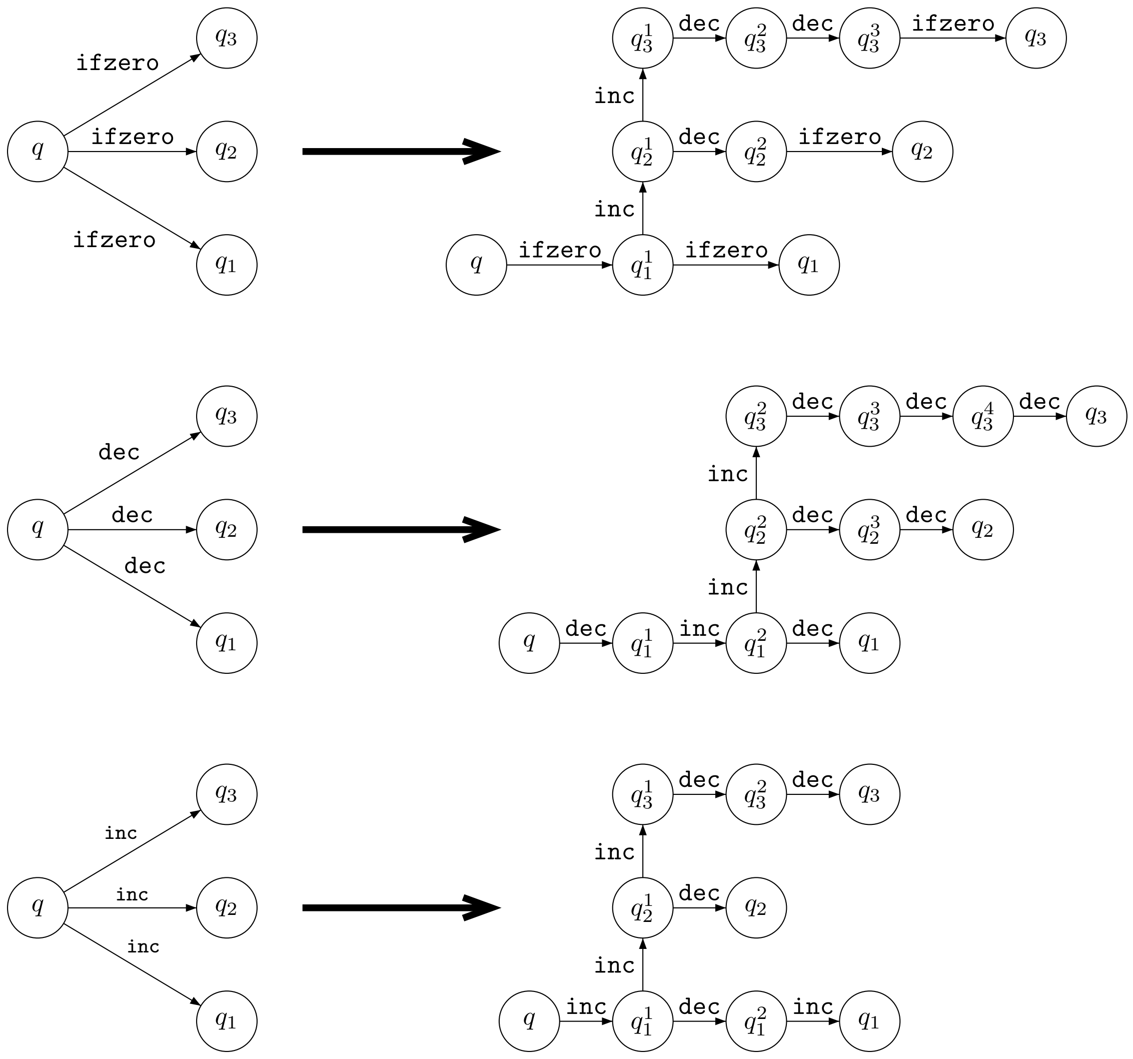


Figure 8: Weak determinization of one-counter automata

[7] M. Bojańczyk, A. Muscholl, T. Schwentick, L. Segoufin, C. David, Two-variable logic on words with data, in: LICS'06, IEEE, 2006, pp. 7–16.

[8] S. Demri, R. Lazić, LTL with the freeze quantifier and register automata, ACM Trans. Comput. Log. 10 (3).

[9] F. Laroussinie, N. Markey, P. Schnoebelen, Temporal logic with forgettable past, in: LICS'02, IEEE, 2002, pp. 383–392.

[10] O. Kupferman, M. Vardi, Memoryful Branching-Time Logic, in: LICS'06, IEEE, 2006, pp. 265–274.

[11] C. David, Mots et données infinies, Master's thesis, LIAFA, in French. 45 pages. (2004).

[12] S. Demri, R. Lazić, D. Nowak, On the freeze quantifier in constraint LTL: decidability and complexity, Information & Computation 205 (1) (2007) 2–24.

[13] R. Lazić, Safely freezing LTL, in: FST&TCS'06, Vol. 4337 of Lecture Notes in Computer Science, 2006, pp. 381–392.

[14] J. Ouaknine, J. Worrell, On Metric Temporal Logic and faulty Turing machines, in: FOSSACS'06, Vol. 3921 of Lecture Notes in Computer Science, Springer, 2006, pp. 217–230.

[15] J. Ouaknine, J. Worrell, On the decidability and complexity of metric temporal logic over finite words, Logical Methods in Computer Science 3 (1:8) (2007) 1–27.

[16] P. Bouyer, A. Petit, D. Thérien, An algebraic approach to data languages and timed languages, Information & Computation 182 (2) (2003) 137–162.

[17] F. Neven, T. Schwentick, V. Vianu, Finite state machines for strings over infinite alphabets, ACM Trans. Comput. Log. 5 (3) (2004) 403–435.

[18] L. Segoufin, Automata and logics for words and trees over an infinite alphabet, in: CSL'06, Vol. 4207 of Lecture Notes in Computer Science, Springer, 2006, pp. 41–57.

[19] H. Björklund, T. Schwentick, On notions of regularity for data languages, in: FCT'07, Vol. 4639 of Lecture Notes in Computer Science, Springer, 2007, pp. 88–99.

[20] R. Alur, T. Henzinger, A really temporal logic, in: FOCS'89, IEEE, 1989, pp. 164–169.

[21] V. Goranko, Hierarchies of modal and temporal logics with references pointers, Journal of Logic, Language, and Information 5 (1996) 1–24.

[22] T. Schwentick, V. Weber, Bounded-variable fragments of hybrid logics, in: STACS'07, Vol. 4393 of Lecture Notes in Computer Science, Springer, 2007, pp. 561–572.

[23] M. Franceschet, M. de Rijke, B.-H. Schlingloff, Hybrid logics on linear structures: Expressivity and complexity, in: TIME-ICTL 2003, IEEE, 2003, pp. 164–171.

[24] M. Franceschet, M. de Rijke, Model checking hybrid logics (with an application to semistructured data), Journal of Applied Logic 4 (3) (2006) 279–304.

[25] B. ten Cate, M. Franceschet, On the complexity of hybrid logics with binders, in: CSL'05, Vol. 3634 of Lecture Notes in Computer Science, Springer, 2005, pp. 339–354.

[26] N. Markey, P. Schnoebelen, Model checking a path, in: CONCUR'03, Vol. 2761 of Lecture Notes in Computer Science, Springer, 2003, pp. 251–261.

[27] P. Jančar, A. Kučera, F. Moller, Z. Sawa, DP lower bounds for equivalence-checking and model-checking of one-counter automata, Information & Computation 188 (1) (2004) 1–19.

[28] O. Serre, Parity games played on transition graphs of one-counter processes, in: FOSSACS'06, Vol. 3921 of Lecture Notes in Computer Science, Springer, 2006, pp. 337–351.

[29] S. Demri, R. Gascon, The effects of bounding syntactic resources on Presburger LTL (extended abstract), in: TIME'07, IEEE, 2007, pp. 94–104.

[30] S. Demri, R. Lazić, A. Sangnier, Model checking freeze LTL over one-counter automata, in: FOSSACS'08, Vol. 4692 of Lecture Notes in Computer Science, Springer, 2008, pp. 490–504, see also the technical report LSV-08-11, LSV (ENS Cachan).

[31] D. M. Gabbay, Expressive functional completeness in tense logic, in: Aspects of Philosophical Logic, Reidel, 1981, pp. 91–117.

[32] O. Ibarra, Reversal-bounded multicounter machines and their decision problems, Journal of the ACM 25 (1) (1978) 116–133.

[33] S. Demri, A. Sangnier, When model checking freeze LTL over counter machines becomes decidable, in: FOSSACS'10, Vol. 6014 of Lecture Notes in Computer Science, Springer, 2010, to appear.