On the Computational Complexity of Verifying One-Counter Processes

Stefan Göller
Universität Bremen, Germany

Richard Mayr
University of Edinburgh, UK

Anthony Widjaja To
University of Edinburgh, UK

***Abstract*—One-counter processes are pushdown systems over a singleton stack alphabet (plus a stack-bottom symbol). We study the complexity of two closely related verification problems over one-counter processes: model checking with the temporal logic EF, where formulas are given as directed acyclic graphs, and weak bisimilarity checking against finite systems. We show that both problems are P^{NP}-complete. This is achieved by establishing a close correspondence with the membership problem for a natural fragment of Presburger Arithmetic, which we show to be P^{NP}-complete. This fragment is also a suitable representation for the global versions of the problems. We also show that there already exists a fixed EF formula (resp. a fixed finite system) such that model checking (resp. weak bisimulation) over one-counter processes is hard for $\mathsf{P}^{\mathsf{NP}[\log]}$. However, the complexity drops to P if the one-counter process is fixed.**

***Keywords*-Complexity theory, Logic**

I. Introduction

The state of the art. Pushdown automata (PDA) (or recursive state machines; RSM) are a natural model for sequential programs with recursive procedure calls, and their verification problems have been studied extensively. The reachability problem for PDA is polynomial [1, 4]. Model checking with the modal μ-calculus was shown to be EXPTIME-complete in [29], and the global version of the model checking problem has been considered in [2, 18]. The EXPTIME lower bound for model checking PDA also holds for the simpler logic CTL and its fragment EG [28], even for a fixed formula. By modifying the construction in [28], it can easily be shown that EXPTIME-hardness even holds for PDA with a finite control of size 1 (also known as basic process algebras, BPA for short) and a general EG formula. On the other hand, model checking PDA with the logic EF (another natural fragment of CTL) is PSPACE-complete [28], and again the lower bound still holds if either the formula or the finite-state control of the PDA is fixed.

Another important aspect is the complexity of checking semantic equivalences between two pushdown automata and between pushdown automata and finite-state systems. Strong- and weak bisimulation equivalence [16] are among the most important semantic equivalences, since they are the ones induced by standard temporal logics with strong (resp. weak) modalities. When comparing two pushdown automata, weak bisimilarity is undecidable [22], while strong bisimilarity is decidable [19] and EXPTIME-hard [12]. When comparing pushdown automata with finite-state systems [12, 14], weak- and strong bisimilarity are PSPACE-complete, and for weak bisimilarity the lower bound even holds for a fixed finite system. See the survey in [23] for further results.

One-counter processes (OCP) are Minsky counter machines with just one counter and action labels on the transitions. They can also be seen as a special case of pushdown automata with just one stack symbol, plus a non-removable bottom symbol which indicates an empty stack (and thus allows to test the counter for zero). So far, unlike for PDA, the picture on the complexity of verification problems for OCP was incomplete. While all upper complexity bounds carry over from PDA, it was not clear how much easier the verification problems are for OCP. Fixed-parameter tractability results also take a slightly different meaning for one-counter processes, because (unlike for PDA, which vary in the number of used stack symbols) fixing the finite control of one-counter processes results in a fixed process.

Model checking OCP with the μ-calculus is PSPACE-complete. The PSPACE upper bound was shown in [20], and a matching lower bound can easily be shown by a reduction from emptiness of alternating unary finite automata, which was shown to be PSPACE-complete in [5, 10]. This lower bound even holds if either the one-counter process or the formula is fixed. However, the complexity of model checking OCP with CTL/EF was open. While the PSPACE upper bound carries over from μ-calculus, the best known lower bound for CTL/EF was DP-hardness [9].

When comparing two one-counter processes, weak bisimilarity is undecidable [15], while strong bisimilarity is decidable [6] and PSPACE-hard [24]. When comparing a one-counter process to a finite-state system, strong bisimilarity is polynomial [11], but weak bisimilarity was only known to be in PSPACE and DP-hard [9, 11].

Our contribution. We establish the complexity of two main open problems by showing them to be P^{NP}-complete.

- EF logic model checking one-counter processes.
- Checking weak bisimilarity of a one-counter process and a finite-state system.

These tight bounds cannot be shown by a direct application of automata theoretic techniques and semilinear sets as in

The first author was supported by the DFG project GELO during his stay at Universität Leipzig. The third author was supported by EPSRC grant E005039 and ORSAS Award.

Logic	PDA	OCP
μ-calc.	EXPTIME	PSPACE
μ-calc., fixed formula	EXPTIME	PSPACE
μ-calc., fixed ctrl.	EXPTIME	PSPACE
EF	PSPACE	P^{NP} (*)
EF, fixed formula	PSPACE	$\mathsf{P}^{\mathsf{NP}[\log]}$ hard (*)
EF, fixed ctrl.	PSPACE	in P (*)

Figure 1. Model checking over PDA and OCP

Bisimulation type	PDA	OCP
$\approx$ finite system	PSPACE	P^{NP} (*)
$\approx$ fixed finite system	PSPACE	$\mathsf{P}^{\mathsf{NP}[\log]}$ hard (*)
fixed ctrl. $\approx$ finite system	P	in P (*)
$\sim$ finite system	PSPACE	P
$\sim$ fixed finite system	in P	in P
fixed ctrl. $\sim$ finite system	P	in P

Figure 2. Bisimulation over PDA and OCP

the case of PDA; instead they require a careful analysis of a special fragment of Presburger Arithmetic (see below).

By default, we represent EF formulae as a dag (i.e., as a directed acyclic graph, not as a tree) which allows one to re-use common subformulae efficiently. (Note that the PSPACE upper bound for EF model checking PDA [28] can easily be extended to formulae in this dag representation.) We establish a close correspondence between the EF model checking problem for one-counter processes and the membership problem for a natural fragment of Presburger Arithmetic, which we show to be P^{NP}-complete. We use quantifier-free Presburger Arithmetic over one variable with addition, subtraction, inequalities, tests of divisibility, and min/max operations. This fragment of Presburger Arithmetic is also shown to be a suitable global representation when one considers the global model checking problem. A matching P^{NP} lower bound is shown by a reduction from DSAT for the case of EF formulae represented as a dag, while the best lower bound for tree-represented EF formulae is $\mathsf{P}^{\mathsf{NP}[\log]}$. In fact, there already exists a fixed EF formula such that model checking one-counter processes is hard for $\mathsf{P}^{\mathsf{NP}[\log]}$, i.e., the data-complexity is $\mathsf{P}^{\mathsf{NP}[\log]}$-hard. However, unlike for most other model checking problems, the expression complexity (i.e., the complexity in the size of the formula) is polynomial. In particular, if the one-counter process is fixed, then the problem becomes solvable in P.

Checking weak bisimilarity of a one-counter process and a finite-state system is shown to be in P^{NP} by a polynomial-time reduction to the EF model checking problem, using the polynomial-size characteristic EF formulae (in dag representation) introduced in [8] (for a more accessible presentation of characteristic formulae see [7]). A matching P^{NP} lower bound is shown by a reduction from DSAT. Analogously to the model checking problem, there exists a fixed finite system such that weak bisimulation checking with one-counter processes is hard for $\mathsf{P}^{\mathsf{NP}[\log]}$. However, if the one-counter process is fixed, checking weak bisimulation with finite-state systems becomes solvable in P.

Figure 1 summarizes the picture on the complexity of model checking for PDA and OCP, whereas Figure 2 summarizes the complexity of weak ($\approx$) and strong ($\sim$) bisimulation with finite systems for PDA and OCP. Our results are marked with (*).

This paper is structured as follows. In Section II we formally define the considered problems and introduce some basic notation. In Section III we define our special fragment of Presburger Arithmetic and establish its fundamental properties. In Section IV we show how to efficiently transform one-counter processes into a suitable normal form, and in Section V we describe the construction of the Presburger formulas which solve the EF model checking problem in P^{NP}. Lower bounds for model checking EF over OCP are shown in Section VI. In Section VII we study the problem of checking weak bisimilarity of a one-counter process and a finite-state system. The paper concludes with a brief section on future work.

II. Preliminaries

General notation: By $\mathbb{N} = \{0, 1, \ldots\}$ we denote the natural numbers. For $i, j \in \mathbb{N}$ we define $[i, j] = \{i, i+1, \ldots, j\}$ and $[j] = [1, j]$. Extend the usual arithmetic operations on $\mathbb{N}$ to sets of numbers as follows. For each $\odot \in \{+, -, \cdot\}$ and $A, B \subseteq \mathbb{N}$, we define $A \odot B = \{a \odot b \mid a \in A, b \in B\}$. For $a \in \mathbb{N}$, we shall also write $a \odot B$ to mean $\{a\} \odot B$. An *arithmetic progression* is any set of numbers of the form $a + b\mathbb{N}$ defined as $a + b \cdot \mathbb{N}$ for some $a, b \in \mathbb{N}$. The number a (resp. b) is said to be the *offset* (resp. *period*) of $a + b\mathbb{N}$. By $\log$ we denote the *logarithm to base* 2. A *dag* is a directed acyclic graph.

Computational complexity: We assume familiarity with the standard complexity classes P, NP, PSPACE, and with notions of completeness for a complexity class. In this paper, *reductions* are always polynomial-time many-to-one reductions. By P^{NP} (resp. $\mathsf{P}^{\mathsf{NP}[\log]}$) we denote the class of languages decidable by some deterministic polynomial-time Turing machine that can invoke some NP oracle (resp. at most logarithmically many times). Recall that

$$\mathsf{NP} \cup \mathsf{coNP} \subseteq \mathsf{P}^{\mathsf{NP}[\log]} \subseteq \mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{PSPACE}.$$

Moreover recall that the second level of the polynomial hierarchy lies between P^{NP} and PSPACE. For more details, we refer the reader to the textbook [17].

One-counter processes and transition systems: Fix some countable set Σ of *actions*. A *transition system* is a pair $T = (S, \{\rightarrow_a \mid a \in \Sigma\})$, where S is a set of *states* and $\rightarrow_a \subseteq S \times S$ is a set of a-labeled *transitions* for each $a \in \Sigma$. We write $s_1 \rightarrow_a s_2$ to abbreviate $(s_1, s_2) \in \rightarrow_a$. We write

$s_1 \to_T s_2$ whenever $s_1 \to_a s_2$ for some $a \in \Sigma$. We define the *size* of T as $|T| = |S| + \sum_{a\in\Sigma} |{\to_a}|$.

A *one-counter process* is a tuple $\mathcal{O} = (Q, \delta_0, \delta_{>0})$, where Q is a finite set of *control locations*, $\delta_0 \subseteq Q\times\Sigma\times Q\times\{0,1\}$ is a finite set of *zero transitions*, and $\delta_{>0} \subseteq Q \times \Sigma \times Q \times \{-1, 0, 1\}$ is a finite set of *positive transitions*. The *size* of a one-counter process is defined as $|\mathcal{O}| = |Q| + |\delta_0| + |\delta_{>0}|$. A *one-counter net* is a one-counter process that additionally satisfies $\delta_0 \subseteq \delta_{>0}$.

A one-counter process $\mathcal{O} = (Q, \delta_0, \delta_{>0})$ defines a transition system $T(\mathcal{O}) = (Q \times \mathbb{N}, \{\to_a |\ a \in \Sigma\})$, where $(q, n) \to_a (q', n + k)$ if and only if either $n = 0$ and $(q, a, q', k) \in \delta_0$, or $n > 0$ and $(q, a, q', k) \in \delta_{>0}$. If $(q_1, n_1) \to_a (q_2, n_2)$ for some $a \in \Sigma$, we also write $(q_1, n_1) \to_{\mathcal{O}} (q_2, n_2)$.

EF logic and weak bisimulation: An *EF dag-formula* is a finite sequence of definitions $\varphi = (\varphi_i)_{i\in[l]}$ for some $l \in \mathbb{N}$, where for each $i \in [l]$ the *definition* φ_i is exactly one of the following, either: $\varphi_i = \mathtt{true}$, $\varphi_i = \neg\varphi_j$ for some $j \in [i-1]$, $\varphi_i = \varphi_j \wedge \varphi_k$ for some $j, k \in [i-1]$, $\varphi_i = \langle a\rangle\varphi_j$ for some $a \in \Sigma$ and some $j \in [i-1]$, or $\varphi_i = \mathrm{EF}\varphi_j$ for some $j \in [i-1]$. For each $i \in [l]$, we define the *size* $|\varphi_i| = 1$ if $\varphi_i = \mathtt{true}$ and $|\varphi_i| = \lceil \log i \rceil$ otherwise. The *size* of φ is defined as $|\varphi| = \sum_{i=1}^{l} |\varphi_i|$. Define the partial order $\prec_\varphi \subseteq [l] \times [l]$ as $(j, i) \in \prec_\varphi$ if and only if φ_j appears in the definition of φ_i. If $j \prec_\varphi^+ i$, we say that φ_j is a *subformula* of φ_i. Note that i is minimal with respect to $\prec_\varphi^+$ whenever $\varphi_i = \mathtt{true}$. Observe that $([l], \prec_\varphi)$ is a dag. We call φ an *EF tree-formula* if moreover $([l], \prec_\varphi)$ is a directed tree. Next, we define the semantics. For this, let $T = (S, \{\to_a |\ a \in \Sigma\})$ be a transition system. Let us define $[\![\varphi_i]\!]_T \subseteq S$ for each $i \in [l]$ by induction on $\prec_\varphi^+$ as follows: $[\![\mathtt{true}]\!]_T = S$, $[\![\neg\varphi_j]\!]_T = S \setminus [\![\varphi_j]\!]_T$, $[\![\varphi_j \wedge \varphi_k]\!]_T = [\![\varphi_j]\!]_T \cap [\![\varphi_k]\!]_T$, $[\![\langle a\rangle\varphi_j]\!]_T = \{s \in T \mid \exists t \in [\![\varphi_j]\!]_T : s \to_a t\}$, and $[\![\mathrm{EF}\varphi_j]\!]_T = \{s \in S \mid \exists t \in [\![\varphi_j]\!]_T : s \to_T^* t\}$. We define $[\![\varphi]\!]_T = [\![\varphi_l]\!]_T$. We also write $(T, s) \models \varphi_i$ whenever $s \in [\![\varphi_i]\!]_T$. We deal with the model checking problem for EF over one-counter processes defined as follows.

MODEL CHECKING

Instance: A one-counter process $\mathcal{O}$, a state (q, n) of $T(\mathcal{O})$ with n given in binary, and an EF dag/tree-formula φ.

Question: $(T(\mathcal{O}), (q, n)) \models \varphi$?

We also consider *global model checking* in this paper, which asks to compute a function mapping each control location q of $\mathcal{O}$ to a representation of all $n \in \mathbb{N}$ such that $(T(\mathcal{O}), (q, n)) \models \varphi$.

We recall the definition of weak bisimulation [7]. Let $T = (S, \{\to_a |\ a \in \Sigma\})$ be a transition system and assume some distinguished internal symbol $\tau \in \Sigma$. Define the *extended transition relation* $\Rightarrow_a \subseteq S \times S$ for each $a \in \Sigma$ as $s \Rightarrow_a t$ if and only if either $a \neq \tau$ and there exist $s', t' \in S$ such that $s \to_\tau^* s' \to_a t' \to_\tau^* t$, or $a = \tau$ and $s \to_\tau^* t$. Given two transition systems $T = (S, \{\to_a |\ a \in \Sigma\})$ and $T' = (S', \{\to'_a |\ a \in \Sigma\})$, a relation $R \subseteq S \times S'$ is a *w-bisimulation* if it is nonempty, and whenever $(s, s') \in R$, then for every $a \in \Sigma$ the following two conditions hold (i) if $s \to_a t$ for some $t \in S$, then $s' \Rightarrow'_a t'$ for some $t' \in S'$ such that $(t, t') \in R$, and (ii) if $s' \to'_a t'$ for some $t' \in S'$, then $s \Rightarrow_a t$ for some $t \in S$ such that $(t, t') \in R$. For every $s \in S$ and every $s' \in S'$, we say that s and s' are *w-bisimilar*, written $s \approx s'$, whenever there exists a w-bisimulation $R \subseteq S \times S'$ such that $(s, s') \in R$. We now define *w-bisimilarity checking* of one-counter processes against finite systems as follows.

W-BISIMILARITY CHECKING

Instance: A one-counter process $\mathcal{O}$, a state (q, n) of $T(\mathcal{O})$ with n given in binary, a finite system T, and a state t of T.

Question: $(q, n) \approx t$?

III. A SUITABLE QUANTIFIER-FREE LOGIC

In this section we introduce a suitable representation of natural numbers in terms of a logic that we call MMA for *Min-Max Arithmetic*. In fact, the sets definable in MMA equal the sets definable in Presburger Arithmetic with one free variable or equivalently the one-dimensional semilinear sets. MMA can be seen as a syntactic variant of Presburger Arithmetic that is tailored towards having a fairly low complexity (P^{NP}) of the membership problem. Moreover, MMA is a suitable formalism for representing solutions of global EF model checking and global w-bisimilarity checking against finite state systems over one-counter processes.

Formally, an MMA *dag-formula* is a sequence of definitions $\alpha = (\alpha_i)_{i\in[l]}$ for some $l \geq 1$, where for each $i \in [l]$ the *definition* α_i is precisely one of the following, where $j, k \in [i-1]$ and where $\sim \in \{\leq, \geq\}$:

(1) $\equiv m \bmod n$, where $n > 0$ and $m \in \mathbb{Z}/n\mathbb{Z}$,
(2) $\sim n$, where $n \in \mathbb{N}$,
(3) $\neg\alpha_j$
(4) $\alpha_j \wedge \alpha_k$,
(5) $\sim \min \alpha_j$,
(6) $n \sim \min \alpha_j$, where $n \in \mathbb{N}$,
(7) $\sim \max(\alpha_j, n)$, where $n \in \mathbb{N}$, or
(8) $m \sim \max(\alpha_j, n)$, where $m, n \in \mathbb{N}$.

We call α_i *atomic* in case it is of type (1) or (2). We will introduce usual abbreviations $<$, $>$, and $=$ as expected. We formally put $\min \emptyset = \infty$, $\max \emptyset = -1$. Moreover we put $k \leq \infty$ and $k \not\geq \infty$, $k \not\leq -1$, and $k \geq -1$ for each $k \in \mathbb{N}$. Define the binary relation $\prec_\alpha \subseteq [l] \times [l]$ as $j \prec_\alpha i$ if and only if α_j occurs in the definition of α_i for each $i, j \in [l]$. Note that $([l], \prec_\alpha)$ is a dag and hence $([l], \prec_\alpha^+)$ is a strict partial order. Recall that i is minimal with respect to $\prec_\alpha^+$ if and only if α_i is atomic. We say α is a MMA *tree-formula* if $([l], \prec_\alpha)$ is a directed tree. Let us now define the semantics of MMA dag-formulas. For each α_i we define the set $[\![\alpha_i]\!] \subseteq \mathbb{N}$ by induction on i w.r.t. $\prec_\alpha^+$ as follows:

(1) $[\![\equiv m \bmod n]\!] = \{k \in \mathbb{N} \mid k \equiv m \bmod n\}$,
(2) $[\![\sim n]\!] = \{k \in \mathbb{N} \mid k \sim n\}$,
(3) $[\![\neg\alpha_j]\!] = \mathbb{N} \setminus [\![\alpha_j]\!]$,
(4) $[\![\alpha_j \wedge \alpha_k]\!] = [\![\alpha_j]\!] \cap [\![\alpha_k]\!]$,
(5) $[\![\sim \min \alpha_j]\!] = \{k \in \mathbb{N} \mid k \sim \min[\![\alpha_j]\!]\}$,
(6) $[\![n \sim \min \alpha_j]\!] = \begin{cases} \mathbb{N} & \text{if } n \sim \min[\![\alpha_j]\!] \\ \emptyset & \text{otherwise} \end{cases}$,
(7) $[\![\sim \max(\alpha_j, n)]\!] = \{k \in \mathbb{N} \mid k \sim \max([\![\alpha_j]\!] \cap [0, n])\}$,
(8)

$$[\![m \sim \max(\alpha_j, n)]\!] = \begin{cases} \mathbb{N} & \text{if } m \sim \max([\![\alpha_j]\!] \cap [0, n]) \\ \emptyset & \text{otherwise.} \end{cases}$$

Observe that MMA can be seen as a fragment of Presburger Arithmetic. We define $[\![\alpha]\!] = [\![\alpha_l]\!]$. We call α *valid* if $[\![\alpha]\!] = \mathbb{N}$, for example ≥ 0 is valid. Define the *size* $|\alpha_i|$ by case distinction as follows: $|\equiv m \bmod n| = |\sim n| = \lceil \log n \rceil$, $|\neg\alpha_j| = \lceil \log j \rceil$, $|\alpha_j \wedge \alpha_k| = \lceil \log j \rceil + \lceil \log k \rceil$, $|\sim \min \alpha_j| = \lceil \log j \rceil$, $|n \sim \min \alpha_j| = \lceil \log n \rceil + \lceil \log j \rceil$, $|\sim \max(\alpha_j, n)| = \lceil \log j \rceil + \lceil \log n \rceil$, and finally $|m \sim \max(\alpha_j, n)| = \lceil \log m \rceil + \lceil \log j \rceil + \lceil \log n \rceil$. Define the *size* of α as $|\alpha| = \sum_{i \in [l]} |\alpha_i|$. For better readability, we will allow more complex definitions such as e.g. $\alpha_i = \neg\alpha_j \wedge (x \equiv 3 \bmod 5)$.

Syntactic sugar - Extended MMA **:** In order to ease our reduction from model checking OCP to evaluating MMA formulas, we introduce *extended* MMA *formulas*. Extended MMA formulas allow definitions of the kind $\alpha_i = \alpha_j - 1$ and $\alpha_i = \alpha_j + 1$. For $\odot \in \{+, -\}$, the semantics is defined as $[\![\alpha_j \odot 1]\!] = [\![\alpha_j]\!] \odot 1$. Observe that, in general, the two operators cannot be interchanged, i.e. we generally do *not* have $[\![(\alpha - 1) + 1]\!] = [\![(\alpha + 1) - 1]\!]$. But observe that $[\![\alpha]\!] = [\![(\alpha + 1) - 1]\!]$. Define the *size* of $|\alpha_j + 1| = |\alpha_j - 1| = \lceil \log j \rceil + 1$. For each natural k define $\alpha_j \odot k$ to be the abbreviation for

$$(\cdots(\alpha_j \underbrace{\odot 1)\cdots) \odot 1}_{k \text{ many times}}$$

for each $\odot \in \{-, +\}$.

Let $\alpha = (\alpha_i)_{i \in [l]}$ be an extended MMA dag-formula. Define ν_i to be maximal number n such that α_j is $\sim n$ or $\sim \max(\alpha_j, n)$ for some $j \in [i]$. Define L_i to be the least common multiple of all $n > 0$ such that the definition of α_j is $\equiv m \bmod n$ for some $j \in [i-1]$ and some $m \in \mathbb{Z}/n\mathbb{Z}$, and 1 if no such formula exists. Observe that $i < j$ implies $L_i | L_j$ and moreover $L_i \in \exp(|\alpha|)$, thus polynomially (in $|\alpha|$) many bits suffice to represent each L_i. For extended MMA, we now state a simple but important periodicity lemma.

Lemma 1 (Periodicity Lemma for extended MMA**)** *Let $i \in [l]$ and assume $n_1, n_2 > i \cdot L_i + \nu_i$. Then the following implication holds:*

$$n_1 \equiv n_2 \bmod L_i \quad \Rightarrow \quad (n_1 \in [\![\alpha_i]\!] \Leftrightarrow n_2 \in [\![\alpha_i]\!])$$

Proof: We prove the lemma by induction on $\prec_\alpha^+$.

For the induction base, assume i is minimal with respect to $\prec_\alpha^+$, i.e. α_i is atomic.

Case α_i is $\equiv m \bmod n$ for some $n > 0$ and some $m \in \mathbb{Z}/n\mathbb{Z}$. Observe that the implication holds since L_i is a multiple of n by definition.

Case α_i is $\sim n$, where $\sim \in \{\leq, \geq\}$ and where $n \in \mathbb{N}$. By definition we have $\nu_i \geq n$. The implication clearly holds since natural numbers exceeding n are either all contained in $[\![\alpha_i]\!]$ or are all not contained in $[\![\alpha_i]\!]$.

For the induction step, assume that i is not minimal with respect to $\prec_\alpha^+$. For this, we make a case distinction according to α_i.

Case $\alpha_i = \neg\alpha_j$ for some $j \in [i-1]$. The required implication holds trivially due to induction hypothesis and the fact that $j \cdot L_j + \nu_j \leq i \cdot L_i + \nu_i$ and $L_i = L_j$.

Case $\alpha_i = \alpha_j \wedge \alpha_k$ for some $j, k \in [i-1]$. First, we have that both L_k and L_j divide L_i. Second, both $j \cdot L_j + \nu_j$ and $k \cdot L_k + \nu_k$ are at most $i \cdot L_i + \nu_i$. Now let $n_1, n_2 > i \cdot L_i + \nu_i$ and assume $n_1 \equiv n_2 \bmod L_i$. Then we have $n_1 \in [\![\alpha_i]\!]$ if and only if $n_1 \in [\![\alpha_j]\!]$ and $n_1 \in [\![\alpha_k]\!]$. By induction hypothesis, the latter is equivalent to $n_2 \in [\![\alpha_j]\!]$ and $n_2 \in [\![\alpha_k]\!]$ which is in turn equivalent to $n_2 \in [\![\alpha_i]\!]$.

Case α_i is $\sim \min \alpha_j$ for some $\sim \in \{\leq, \geq\}$ and for some $j \in [i-1]$. We claim that the implication holds for i by distinguishing if $[\![\alpha_j]\!]$ is empty or not. In case $[\![\alpha_j]\!] = \emptyset$, (recall $\min \emptyset = \infty$), then $[\![\alpha_i]\!]$ either equals $\mathbb{N}$ or $\emptyset$, depending on $\sim$. The implication obviously holds in this case. In case $[\![\alpha_j]\!] \neq \emptyset$, then there exists some $n \in [\![\alpha_j]\!]$ with $n \leq j \cdot L_j + \nu_j + L_j$ by induction hypothesis. Observe that the latter is less than or equal to $(i-1) \cdot L_i + \nu_i + L_i = i \cdot L_i + \nu_i$. Again, depending on $\sim$, all naturals exceeding n (in particular naturals exceeding $i \cdot L_i + \nu_i$) are either all in $[\![\alpha_i]\!]$ or are all not in $[\![\alpha_i]\!]$. Thus, the implication holds.

Case $\alpha_i = n \sim \min \alpha_j$ for some $n \in \mathbb{N}$, some $\sim \in \{\leq, \geq\}$, and some $j \in [i-1]$. Since $[\![\alpha_i]\!]$ either equals $\mathbb{N}$ or $\emptyset$, the implication trivially holds.

Case α_i is $\sim \max(\alpha_j, n)$ for some $\sim \in \{\leq, \geq\}$, some $j \in [i-1]$, and some $n \in \mathbb{N}$. First observe that $n \leq \nu_i$ by definition. Second, all naturals exceeding n (in particular those exceeding ν_i) either all satisfy α_i or all do not, depending on $\sim$. Thus, the implication holds.

Case α_i is $m \sim \max(\alpha_j, n)$ for some $\sim \in \{\leq, \geq\}$, some $j \in [i-1]$, and some $m, n \in \mathbb{N}$. Since $[\![\alpha_i]\!]$ either equals $\mathbb{N}$ or $\emptyset$, the implication holds trivially.

Case $\alpha_i = \alpha_j \odot 1$ for some $j \in [i-1]$ and some $\odot \in \{+, -\}$. The implication follows directly from $j \cdot L_j + \nu_j + 1 \leq (i-1) \cdot L_i + \nu_i + L_i = i \cdot L_i + \nu_i$ and induction hypothesis. ∎

It turns out that extended MMA formulas are neither more expressive nor more succinct than MMA formulas.

Lemma 2 *The following problem is computable in polynomial time:*

INPUT: An extended MMA *dag-formula* α.

OUTPUT: A MMA *dag-formula* β *such that* $[\![\alpha]\!] = [\![\beta]\!]$.

By a bottom-up computation and combining Lemma 1 and Lemma 2, we can deduce a P^{NP} upper bound for the membership problem for extended MMA.

Proposition 3 *The following problem is in* P^{NP}*:*

INPUT: $n_0 \in \mathbb{N}$ *in binary and an extended* MMA *dag-formula* α.

QUESTION: $n_0 \in [\![\alpha]\!]$*?*

Proof: First, it is easy to see that if α is an MMA dag-formula with no occurrence of $\min$ and $\max$ then checking whether $n_0 \in [\![\alpha]\!]$ can be done is polynomial time. We shall make use of this basic fact in the proof. In a first step, we apply Lemma 2 and compute in polynomial time an MMA dag-formula formula β such that $[\![\beta]\!] = [\![\alpha]\!]$. Assume $\beta = (\beta_i)_{i\in[l]}$. In a second step, we eliminate $\min$ and $\max$ operators that occur in β inductively on $\prec_\beta^+$.

For this, let $i \in [l]$ be minimal with respect to $\prec_\beta^+$ such that the definition β_i contains either $\min$ or $\max$. In case β_i is $\sim \min \beta_j$, we know that for each $n_1, n_2 > j \cdot L_j + \nu_j$ with $n_1 \equiv n_2 \bmod L_j$ we have $n_1 \in [\![\beta_j]\!]$ if and only if $n_2 \in [\![\beta_j]\!]$ by Lemma 1. This implies that $\min[\![\beta_j]\!] \in [0, (j+1)\cdot L_j + \nu_j] \cup \{\infty\}$. Moreover, observe that $(j+1)\cdot L_j + \nu_j$ can represented using polynomially many bits in $|\beta|$. Furthermore, note that we can decide deterministically in polynomial time whether, for a given m given in binary, we have $m \in [\![\beta_j]\!]$, since neither the definition of β_j nor the definition of β_k contains $\min$ or $\max$, for each $k \prec_\beta^+ j$. Via a binary search method, we can compute

$$\mu \;=\; \min\{m \in [0, (j+1)\cdot L_j + \nu_j] \mid m \in [\![\beta_j]\!]\}$$

by some deterministic polynomially time bounded Turing machine that has access to an NP oracle. After that, we "symbolically" modify β by replacing β_i's previous definition $\sim \min \beta_j$ by $\sim \mu$.

The cases when $\beta_i = n \sim \min \beta_j$, $\beta_i = \sim \max(\beta_j, n)$, or $\beta_i = m \sim \max(\beta_j, n)$ can be dealt with analogously.

We repeat this replacement process until β does not contain any $\min$ or $\max$ operator. Finally, we check if $n_0 \in [\![\beta]\!]$ in polynomial time. ∎

Observe that we can think of extended MMA to be MMA enhanced with the usage of "offsets" of the kind $\alpha_j \pm k$, where k is a natural that is given in *unary*. The question arises, why one does not extend MMA more generally, by allowing offsets to be given in *binary*. However, this yields a logic with a PSPACE-complete membership problem.

Proposition 4 *The following problem is* PSPACE*-complete:*
INPUT: An extended MMA *dag-formula* α*, where offsets are given in binary.*
QUESTION: $0 \in [\![\alpha]\!]$*?*

IV. Saturating one-counter processes and computing small arithmetic progressions

For the rest of this section, let us fix some one-counter process $\mathcal{O} = (Q, \delta_0, \delta_{>0})$. For technical reasons, we add a new transition label $\lambda \in \Sigma$ that does not previously occur in $\delta_0 \cup \delta_{>0}$ and which we fix for the rest of this section. Our goal is to "saturate" $\mathcal{O}$ with λ-labeled transitions so that we only have to consider *normalized paths*, i.e. paths, where the sequence of counter values of the involved states are first non-increasing and then non-decreasing. Let $\mathcal{O}'$ denote the resulting OCP *after saturation*. Our saturation construction has the following motivation: (1) We can compute in polynomial time all information needed for representing normalized paths in $T(\mathcal{O}')$ in terms of few small arithmetic progressions. (2) For every EF dag-formula φ in which λ does not occur we have $(T(\mathcal{O}), s) \models \varphi$ if and only if $(T(\mathcal{O}'), s) \models \varphi$ for every state $s \in Q \times \mathbb{N}$.

A *path* in $T(\mathcal{O})$ is a non-empty finite sequence of states $\pi = (q_1, n_1) \to_\mathcal{O} (q_2, n_2) \cdots \to_\mathcal{O} (q_k, n_k)$. We call π *mountain*, if $n_1 = n_k$ and $n_i \geq n_1$ for each $i \in [k]$. We call π *zero*, if $n_i = 0$ for some $i \in [k]$, otherwise we call π *positive*. Let $(q_1, n_1), (q_2, n_2) \in Q \times \mathbb{N}$ be states. Then, we write $(q_1, n_1) \downarrow_\mathcal{O} (q_2, n_2)$ (resp. $(q_1, n_1) \uparrow_\mathcal{O} (q_2, n_2)$) whenever $(q_1, n_1) \to_\mathcal{O} (q_2, n_2)$ and $n_2 \leq n_1$ (resp. and $n_2 \geq n_1$). We now present a saturation construction that allows us to shortcut mountain paths by adding λ-transitions.

Choosing control locations $q, q' \in Q$ and $\delta \in \{\delta_0, \delta_{>0}\}$, we now present rules (R1) to (R4) that can be applied only if $(q, \lambda, q', 0) \notin \delta$. In this case, we can add the transition $(q, \lambda, q', 0)$ to δ if at least one of the following conditions holds:

(R1) $(q, a, q', 0) \in \delta$ for some $a \in \Sigma$.
(R2) $(q, a_1, q_1, +1) \in \delta$ and $(q_1, a_2, q', -1) \in \delta_{>0}$ for some $q_1 \in Q$ and some $a_1, a_2 \in \Sigma$.
(R3) $(q, a_1, q_1, +1) \in \delta$, $(q_1, \lambda, q_2, 0) \in \delta_{>0}$, and $(q_2, a_2, q', -1) \in \delta_{>0}$ for some $q_1, q_2 \in Q$ and some $a_1, a_2 \in \Sigma$.
(R4) $(q, \lambda, q_1, 0) \in \delta$ and $(q_1, \lambda, q', 0) \in \delta$ for some $q_1 \in Q$.

Formally, let $\mathcal{O}' = (Q, \delta_0', \delta_{>0}')$ denote the unique one-counter process that we obtain from $\mathcal{O}$ by applying rules (R1)–(R4) until it is no longer possible.

Lemma 5 *Let* $s, t \in Q \times \mathbb{N}$ *be states. Then, the following three statements are equivalent:*

(1) $s \to_\mathcal{O}^* t$.
(2) $s \to_{\mathcal{O}'}^* t$.
(3) There exists some state $u \in Q \times \mathbb{N}$ *such that* $s \downarrow_{\mathcal{O}'}^* u$ *and* $u \uparrow_{\mathcal{O}'}^* t$.

Observe that if $(q_1, n_1) \uparrow^*_{\mathcal{O}'} (q_2, n_2)$ and $n_1 > 0$, then also $(q_1, n_1 + i) \uparrow^*_{\mathcal{O}'} (q_2, n_2 + i)$ for each $i \in \mathbb{N}$. Similarly, if $(q_1, n_1) \downarrow^*_{\mathcal{O}'} (q_2, n_2)$ and $n_2 > 0$, then also $(q_1, n_1 + i) \downarrow^*_{\mathcal{O}'} (q_2, n_2 + i)$ for each $i \in \mathbb{N}$. This motivates us to define, for each $q_1, q_2 \in Q$, the following set of differences of counter values of monotone positive paths:

$$\Delta^{>0}_{\uparrow}(q_1, q_2) = \{d \in \mathbb{N} \mid (q_1, 1) \uparrow^*_{\mathcal{O}'} (q_2, d + 1)\}$$
$$\Delta^{>0}_{\downarrow}(q_1, q_2) = \{d \in \mathbb{N} \mid (q_1, d + 1) \downarrow^*_{\mathcal{O}'} (q_2, 1)\}$$

Analogously, we collect the set of differences of counter values of monotone zero paths:

$$\Delta^{=0}_{\uparrow}(q_1, q_2) = \{d \in \mathbb{N} \mid (q_1, 0) \uparrow^*_{\mathcal{O}'} (q_2, d)\}$$
$$\Delta^{=0}_{\downarrow}(q_1, q_2) = \{d \in \mathbb{N} \mid (q_1, d) \downarrow^*_{\mathcal{O}'} (q_2, 0)\}$$

A theorem due to Chrobak [3] and Martinez [13] states that from a nondeterministic finite automaton over a unary alphabet one can compute in polynomial time an at most quadratically larger equivalent one that is in a certain normal form (Chrobak normal form). However, both papers contain a subtle flaw that was recently fixed in [25]. The following lemma will use this result.

Lemma 6 *Each of the sets $\Delta^{>0}_{\uparrow}(q_1, q_2)$, $\Delta^{>0}_{\downarrow}(q_1, q_2)$, $\Delta^{=0}_{\uparrow}(q_1, q_2)$, $\Delta^{=0}_{\downarrow}(q_1, q_2)$ is equivalent to a union of $O(|Q|^2)$ arithmetic progressions with offsets bounded by $O(|Q|^2)$ and periods bounded by $O(|Q|)$ that are moreover computable in polynomial time.*

Let $q_1, q_2 \in Q$ be control locations. Note that if $(q_1, n) \downarrow^*_{\mathcal{O}'} (q_3, 1) \uparrow^*_{\mathcal{O}'} (q_2, n)$ for some $q_3 \in Q$, then also $(q_1, n+i) \downarrow^*_{\mathcal{O}'} (q_3, 1+i) \uparrow^*_{\mathcal{O}'} (q_2, n+i)$. Therefore, we define $\nabla(q_1, q_2) \in \mathbb{N} \cup \{\infty\}$ to be

$$\min\{n > 0 \mid \exists q_3 \in Q : (q_1, n) \downarrow^*_{\mathcal{O}'} (q_3, 1) \uparrow^*_{\mathcal{O}'} (q_2, n)\}.$$

Observe that $\nabla(q, q) = 1$ for every $q \in Q$.

Lemma 7 *Either $\nabla(q_1, q_2) = \infty$ or $\nabla \in O(|Q|^2)$. Moreover $\nabla(q_1, q_2)$ can be computed in polynomial time.*

The next lemma characterizes zero paths.

Lemma 8 *There is a zero path from (q, n) to (q', n') in $T(\mathcal{O}')$ if and only if $n \in \Delta^{=0}_{\downarrow}(q, q'')$ and $n' \in \Delta^{=0}_{\uparrow}(q'', q')$ for some $q'' \in Q$.*

The next lemma characterizes positive paths.

Lemma 9 *Assume $n \leq n'$. Then there exists a positive path in $T(\mathcal{O}')$ from (q, n) to (q', n') if and only if $n \geq \nabla(q, q'')$ and $n' - n \in \Delta^{>0}_{\uparrow}(q'', q')$ for some $q'' \in Q$.*
Assume $n \geq n'$. Then there exists a positive path from (q, n) to (q', n') in $T(\mathcal{O}')$ if and only if $n' \geq \nabla(q'', q')$ and $n - n' \in \Delta^{>0}_{\downarrow}(q, q'')$ for some $q'' \in Q$.

V. A translation to MMA

For the rest of this section, let us fix a one-counter process $\mathcal{O} = (Q, \delta_0, \delta_{>0})$ and an EF dag-formula $\varphi = (\varphi_i)_{i \in [l]}$. Assume that $Q = \{q_1, \ldots, q_k\}$. Due to technical simplicity, we will confuse each element $(q_i, j) \in Q \times [l]$ with the corresponding natural number $i + (j-1) \cdot k \in [k \cdot l]$. The goal is to present a polynomial time algorithm to compute an extended MMA formula $\alpha = (\alpha_{(q,j)})_{(q,j) \in Q \times [l]}$ such that $[\![\alpha_{(q,j)}]\!] = \{n \in \mathbb{N} \mid (T(\mathcal{O}), (q, n)) \models \varphi_j\}$ for each $(q, j) \in Q \times [l]$.
First, we saturate $\mathcal{O}$ in the sense of Section IV in polynomial time. Then, we apply Lemma 6 and compute in polynomial time the sets $\Delta^{>0}_{\downarrow}(q, q')$, $\Delta^{>0}_{\uparrow}(q, q')$, $\Delta^{=0}_{\downarrow}(q, q')$, and $\Delta^{=0}_{\uparrow}(q, q')$, which are each unions of $O(|Q|^2)$ arithmetic progressions with offsets bounded by $O(|Q|^2)$ and periods bounded by $O(|Q|)$ for each $q, q' \in Q$. By applying Lemma 9 we compute in polynomial time $\nabla(q, q') \in [n_\nabla] \cup \{\infty\}$ for each $q, q' \in Q$, where $n_\nabla \in O(|Q|^2)$.

Let us now present the computation of $\alpha = (\alpha_{(q,j)})_{(q,j) \in Q \times [l]}$. We will do this by induction on j with respect to $\prec^+_\varphi$ and simultaneously for each $q \in Q$.

Base. Assume j is minimal with respect to $\prec^+_\varphi$. Then $\varphi_j = \texttt{true}$ and we put $\alpha_{(q,j)} = (\geq 0)$ for each $q \in Q$.

Step.

Assume $\varphi_j = \neg\varphi_{j'}$ for some $j' \in [j-1]$. Then we put $\alpha_{(q,j)} = \neg\alpha_{(q,j')}$ for each $q \in Q$.

Assume $\varphi_j = \varphi_{j_1} \wedge \varphi_{j_2}$ for some $j_1, j_2 \in [j-1]$. Then we put $\alpha_{(q,j)} = \alpha_{(q,j_1)} \wedge \alpha_{(q,j_2)}$ for each $q \in Q$.

Assume $\varphi_j = \langle a \rangle \varphi_{j'}$ for some $a \in \Sigma$ and some $j' \in [j-1]$. By induction hypothesis, we have

$$\alpha_{(q',j')} = \{n \in \mathbb{N} \mid (q', n) \models \varphi_{j'}\}$$

for each $q' \in Q$. By putting $\widehat{+} = -$ and $\widehat{-} = +$, we define $\alpha_{(q,j)}$ as the conjunction of

$$(= 0) \to \left(\bigvee_{\substack{q' \in Q: \\ (q,a,q',+1) \in \delta_0}} \alpha_{(q',j')} - 1 \quad \vee \bigvee_{\substack{q' \in Q: \\ (q,a,q',0) \in \delta_0}} \alpha_{(q',j')} \right)$$

and

$$(> 0) \to \left(\bigvee_{\substack{q' \in Q, \odot \in \{-,+\}: \\ (q,a,q',\odot 1) \in \delta_{>0}}} \alpha_{(q',j')} \widehat{\odot} 1 \quad \vee \bigvee_{\substack{q' \in Q: \\ (q,a,q',0) \in \delta_{>0}}} \alpha_{(q',j')} \right).$$

Finally, assume $\varphi_j = \mathrm{EF}\varphi_{j'}$ for some $j' \in [j-1]$. Let us first fix control locations $q, q' \in Q$. By induction hypothesis

$$[\![\alpha_{(q',j')}]\!] = \{n \in \mathbb{N} \mid (q', n) \models \varphi_{j'}\}.$$

By Lemma 1 we know that for each $n_1, n_2 \in \mathbb{N}$ which exceed a threshold $t_{(q',j')}$ such that $n_1 \equiv n_2 \bmod L_{(q',j')}$ we

have $n_1 \in [\![\alpha_{(q',j')}]\!]$ if and only if $n_2 \in [\![\alpha_{(q',j')}]\!]$. Note that this implies that $[\![\alpha_{(q',j')}]\!]$ is infinite if and only if there exists some $n \in [\![\alpha_{(q',j')}]\!]$ such that $t_{(q',j')} < n \leq t_{(q',j')} + L_{(q',j')}$.

Let us now fix an arithmetic progression $a+b\mathbb{N}$ with $b > 0$ that is a subset of some of the sets $\Delta_{\downarrow}^{>0}(q,q')$,$\Delta_{\uparrow}^{>0}(q,q')$, $\Delta_{\downarrow}^{=0}(q,q')$, or $\Delta_{\uparrow}^{=0}(q,q')$. Moreover, let $c \in \mathbb{Z}/b\mathbb{Z}$ be some residue class. We aim at defining an MMA formula $\inf(q',j',c,b)$ that is valid if and only if there are infinitely many naturals that satisfy $\alpha_{(q',j')}$ and that are congruent c modulo b. Now observe that the latter is the case exactly whenever there exists some $n \in [\![\alpha_{(q',j')}]\!]$ such that $t_{(q',j')} < n \leq t_{(q',j')} + L_{(q',j')}$ and moreover $n \equiv c \mod \gcd(b, L_{(q',j')})$. Let us define the auxiliary MMA formula

$$\psi(q',j',c,b) \;=\; \alpha_{(q',j')} \wedge (\equiv c \mod \gcd(b, L_{(q',j')}))$$

and finally $\inf(q',j',c,b)$ as

$$t_{(q',j')} < \max(\psi(q',j',b,c), t_{(q',j')} + L_{(q',j')}).$$

Next, we aim at defining the set

$$\{n \in \mathbb{N} \mid \exists n' \in \mathbb{N} : (q,n) \to_{\mathcal{O}}^{*} (q',n'), (q',n') \models \varphi_{j'}\}$$

in terms of an extended MMA formula. For this, assume that there is a path π from (q,n) to (q',n') in $T(\mathcal{O})$ such that $(q',n') \models \varphi_{j'}$, where $n' \in \mathbb{N}$. We distinguish three (not necessarily distinct) cases. Either (1) π is positive and $n \leq n'$, (2) π is positive and $n \geq n'$, or (3) π is zero. We will realize each of these cases by corresponding extended MMA dag-formulas $\beta_1(q,q')$, $\beta_2(q,q')$, and $\beta_3(q,q')$ respectively.

Let us consider case (1), i.e. π is positive and $n \leq n'$. Then $n \geq \nabla(q,q'')$ and $n' - n \in \Delta_{\uparrow}^{>0}(q'',q')$ for some $q'' \in Q$ by Lemma 9. Thus, $(n'-n) \in a + b\mathbb{N}$ for some arithmetic progression $a + b\mathbb{N} \subseteq \Delta_{\uparrow}^{>0}(q'',q')$. Furthermore, let $c \in \mathbb{Z}/b\mathbb{Z}$ be the residue class of n' modulo b. So altogether, we will fix the witnesses q'', $a + b\mathbb{N}$, and c in the following.

Let us first assume that $b > 0$. We now distinguish the cases when there are either (i) infinitely or (ii) finitely many $n'' \in \mathbb{N}$ such that $n'' \equiv c \mod b$ and $(q',n'') \models \varphi_{j'}$.
Case (i) is expressed by the formula $\gamma(q'',a,b,c)_{\infty}$ which is defined as

$$\inf(q',j',c,b) \wedge \;\geq \nabla(q,q'') \wedge \;\equiv (c-a) \bmod b.$$

Case (ii) can be realized by saying that the maximal such n'' is reachable from n via the arithmetic progression $a + b\mathbb{N}$. For this, let the formula $\gamma(q'',a,b,c)_{<\infty}$ be defined as the conjunction of

$$\neg \inf(q',j',c,b) \wedge \;\geq \nabla(q,q'')$$

and

$$\equiv (c-a) \mod b \wedge \;\leq \max(\alpha_{(q',j')} - a, t_{(q',j')}).$$

The last conjunct guarantees that $n + a \leq n''$, which is necessary since we have to have that $n'' \in n + a + b\mathbb{N}$.

The case when $b = 0$ can easily be realized by putting

$$\gamma(q'',a) \;= \geq \nabla(q,q'') \;\wedge\; (\alpha_{(q',j')} - a).$$

Altogether, we put

$$\beta_1(q,q') \quad = \bigvee_{q'' \in Q} \bigvee_{a+0\mathbb{N} \subseteq \Delta_{\uparrow}^{>0}(q,q'')} \gamma(q'',a)$$

$$\vee \bigvee_{\substack{a+b\mathbb{N} \subseteq \Delta_{\uparrow}^{>0}(q,q'') \\ b>0, c \in \mathbb{Z}/b\mathbb{Z}}} \gamma(q'',a,b,c)_{\infty} \vee \gamma(q'',a,b,c)_{<\infty}.$$

Let us now consider case (2), i.e. when π is positive and $n \geq n'$. Then $n' \geq \nabla(q'',q')$ and $n - n' \in \Delta_{\downarrow}^{>0}(q,q'')$ for some $q'' \in Q$ by Lemma 9. Hence, $n - n' \in a + b\mathbb{N}$ for some $a + b\mathbb{N} \subseteq \Delta_{\downarrow}^{>0}(q,q'')$.
Firstly, let us assume that $b > 0$. Recall that $c \in \mathbb{Z}/b\mathbb{Z}$ is the residue class of n'. Now the simple observation is that the witness n' can be replaced by the minimal $n'' \in \mathbb{N}$ such that $n'' \equiv n' \equiv c \mod b$, $n'' \geq \nabla(q'',q')$, and $(q',n'') \models \varphi_{j'}$. We realize this by the formula $\theta(q'',a,b,c)$ defined as the conjunction of $\equiv c + a \bmod b$ and

$$\geq \min\left((\geq \nabla(q'',q') \;\wedge\; \alpha_{(q',j')} \;\wedge\; \equiv c \bmod b) + a \right).$$

Secondly, let us assume that $b = 0$. This case is realized by the formula

$$\theta(q'',a) \;= \geq (\nabla(q'',q') + a) \;\wedge\; (\alpha_{(q',j')} + a).$$

Altogether, we define $\beta_2(q,q')$ to be

$$\bigvee_{q'' \in Q} \bigvee_{\substack{a+b\mathbb{N} \subseteq \Delta_{\downarrow}^{>0}(q,q'') \\ b>0, c \in \mathbb{Z}/b\mathbb{Z}}} \theta(q'',a,b,c) \quad \vee \bigvee_{a+0\mathbb{N} \subseteq \Delta_{\downarrow}^{>0}(q,q'')} \theta(q'',a).$$

Finally, let us consider case (3), i.e. when π is zero. For each $q'' \in Q$, define the predicate

$$\exists m \in \mathbb{N} : m \in \Delta_{\uparrow}^{=0}(q'',q') \wedge (q',m) \models \varphi_{j'}.$$

In other words, case (3) can be rephrased as $n \in \Delta_{\downarrow}^{=0}(q,q'')$ and $\pi(q'')$ for some $q'' \in Q$ by Lemma 8. Now check that the predicate $\pi(q'')$ can be expressed as

$$\bigvee_{\substack{a+b\mathbb{N} \subseteq \Delta_{\uparrow}^{=0}(q'',q') \\ b>0}} 0 \leq \max(\equiv a \bmod b \wedge \alpha_{(q',j')}, t_{(q',j')} + L_{(q',j')})$$

$$\vee \bigvee_{a+0\mathbb{N} \subseteq \Delta_{\uparrow}^{=0}(q'',q')} a = \min((\alpha_{(q',j')} - a) + a).$$

Define $\beta_3(q,q') = \bigvee_{q'' \in Q} \pi(q'') \wedge \rho(q'')$, where $\rho(q'')$ is

$$\bigvee_{a+b\mathbb{N} \subseteq \Delta_{\downarrow}^{=0}(q,q'')} (\equiv a \mod b) \quad \vee \bigvee_{a+0\mathbb{N} \subseteq \Delta_{\downarrow}^{=0}(q,q'')} (= a).$$

We finally put $\alpha_{(q,j)} = \bigvee_{q' \in Q} \beta_1(q,q') \vee \beta_2(q,q') \vee \beta_3(q,q')$. This concludes the definition of α. It is straightforward to see that α can be computed in time polynomially in $|\mathcal{O}| + |\varphi|$.

By additionally applying Lemma 2, we obtain the following theorem.

Theorem 10 *From a given one-counter process $\mathcal{O}$ and a given EF dag-formula φ, we can compute in polynomial time for each control location q of $\mathcal{O}$ an* MMA *dag-formula $\alpha(q)$ such that $[\![\alpha(q)]\!] = \{n \in \mathbb{N} \mid (T(\mathcal{O}), (q, n)) \models \varphi\}$.*

By combining Theorem 10 with Proposition 3 we obtain the following corollary.

Corollary 11 *The following problem is in P^{NP}:*

INPUT: A one-counter process $\mathcal{O} = (Q, \delta_0, \delta_{>0})$, a state $(q, n) \in Q \times \mathbb{N}$ with n given in binary, and an EF dag-formula φ.

QUESTION: $(T(\mathcal{O}), (q, n)) \models \varphi$?

A more precise analysis our translation allows us to derive the following polynomial time upper bound, when the one-counter process is fixed.

Theorem 12 *For every fixed one-counter process $\mathcal{O} = (Q, \delta_0, \delta_{>0})$ the following problem is in P:*

INPUT: A state $(q, n) \in Q \times \mathbb{N}$, where n is given in binary and an EF dag-formula φ.

QUESTION: $(T(\mathcal{O}), (q, n)) \models \varphi$?

VI. Lower bounds on model checking

In this section we give two results concerning lower bounds for EF model checking. We first prove a matching P^{NP} lower bound for model checking EF logic dag-formulas. Then we prove a $\mathsf{P}^{\mathsf{NP}[\log]}$ lower bound for model checking with a *fixed* EF formula (in this case, tree or dag representations make no difference). Both results hold even over one-counter nets. First, we prove a P^{NP} lower bound for the problem of model checking EF logic dag-formulas. The problem we will reduce from is the well-known P^{NP}-complete problem DSAT [17], which takes the following input: a sequence of boolean formulas $F_1, \ldots, F_n$ with variables $x_1, \ldots, x_n$ and sets of variables $Z_1, \ldots, Z_n$ such that the formula F_i can take only variables from $\{x_1, \ldots, x_{i-1}\}$ and Z_i. The goal is to decide whether there exists an assignment $\sigma : \{x_1, \ldots, x_n\} \to \{0, 1\}$ that sets x_n to 1 such that the following are satisfied for all $i \in [n]$:

$$\sigma(x_i) = 1 \quad \Leftrightarrow \quad \exists Z_i F_i(x_1, \ldots, x_{i-1}, Z_i). \qquad (1)$$

Notice that imposing the constraint in (1) ensures that there exists a *unique* assignment σ. The only question is whether this assignment satisfies $\sigma(x_n) = 1$.

Theorem 13 *The problem of model checking EF logic dag-formulas over one-counter nets is P^{NP}-hard.*

Proof Sketch.: The reduction is from DSAT. Given a sequence of boolean formulas $F_1, \ldots, F_n$ with variables $x_1, \ldots, x_n$ and sets $Z_1, \ldots, Z_n$ of variables, we construct a one-counter net $\mathcal{O}$ and an EF dag-formula φ that mimic the structure of the given sequence of formulas. By using a well-known *Gödel encoding technique* that was also used in [9, 11], we can view each natural $m \in \mathbb{N}$ as the truth assigment $\alpha_m : \{x_1, \ldots, x_n\} \to \{0, 1\}$, where $\alpha_m(x_i)$ is assigned to 1 if the i-th prime number p_i divides m and 0 otherwise. Checking divisibility by p_i can be easily done by looping through a cycle of length p_i. Now, both $\mathcal{O}$ and φ consist of n "levels", where each level can refer only to lower levels. Testing satisfiability of F_i is replaced by divisibility tests and satisfiability of appropriate F_js, where $j < i$. Since a variable x_i might be referred to several times in the sequence $F_1, \ldots, F_n$, we need to use EF dag-formulas to avoid an exponential blowup. ■

We proceed to the model checking problem with the formula fixed. First, define the problem INDEX-ODD from which our reduction is performed: given a list $F_1, \ldots, F_n$ of boolean formulas in 3-CNF, does there exist an odd index $i \in [n]$ s.t. $F_1, \ldots, F_i$ are all satisfiable and $F_{i+1}, \ldots, F_n$ are all unsatisfiable? This problem is $\mathsf{P}^{\mathsf{NP}[\log]}$-complete. A $\mathsf{P}^{\mathsf{NP}[\log]}$ upper bound is immediate by a simple binary search in the list $F_1, \ldots, F_n$ by invoking an NP oracle at each step to determine the rightmost satisfiable formula F_i. The lower bound is also immediate from Wagner's sufficient conditions for $\mathsf{P}^{\mathsf{NP}[\log]}$-hardness [27, Theorem 5.2] (see also [21, Lemma 7]). The reduction for the following theorem can be achieved via Gödel encoding as in the proof of Theorem 13.

Theorem 14 *There exists a fixed EF formula φ such that model checking φ over one-counter nets is $\mathsf{P}^{\mathsf{NP}[\log]}$-hard.*

It is worth observing that the above lower bounds imply that the membership problem of MMA dag-formulas of *fixed* nesting depth of both $\min$ and $\max$ operators is hard for $\mathsf{P}^{\mathsf{NP}[\log]}$. The reason for this is that the min and max nesting depth of the resulting MMA formula in the translation of the previous section depends only on the size of the given EF formula not on the size of the one-counter process.

On the other hand, our best upper bound of determining only the parity of the minimal (resp. maximal up to some given threshold) natural number satisfying a given MMA formula that is both $\min$-free and $\max$-free is P^{NP}. This is basically the reason why we cannot improve the upper bound of EF over OCP when the formula is *fixed*.

VII. Equivalence checking

In this section, we study the complexity of the w-bisimilarity checking problem. We show that the general problem is P^{NP}-complete. We then proceed by showing that the problem is

solvable in P for each fixed one-counter process. Finally, we show that that there exists a fixed finite system for which the w-bisimilarity checking problem is already $\mathsf{P}^{\mathsf{NP}[\log]}$-hard.

We extend our EF logic with the following operator $\mathrm{EF}\langle\tau^*\rangle$, which we will also abbreviate as EF'. The meaning is that for a transition system $T = (S, \{\rightarrow_a |\ a \in \Sigma\})$ and a state $s \in S$ we have $T, s \models \mathrm{EF}'\varphi$ if and only if there exists $s' \in S$ such that $s \Rightarrow_\tau s'$ and $T, s' \models \varphi$. Note that our P^{NP} upper bound easily carries over to this extended EF logic by simply restricting ourselves to τ-transitions when we have EF' operators. The following is a well-known result from [8] (and a more recent presentation can be found in [7]).

Lemma 15 ([8]) *Let $T_1 = (S_1, \{\overset{1}{\rightarrow}_a |\ a \in \Sigma\})$ be a transition system and $T_2 = (S_2, \{\overset{2}{\rightarrow}_a |\ a \in \Sigma\})$ be a finite transition system with k states. Then, given any state $s_2 \in S_2$, we can construct a dag-formula φ_{s_2,T_2} in the extended EF logic in polynomial time (in k) such that, for every state $s_1 \in S_1$, it is the case that $s_1 \approx s_2$ if and only if $(T_1, s_1) \models \varphi_{s_2,T_2}$.*

In other words, Lemma 15 implies that the w-bisimulation checking problem is polynomial-time reducible to the problem of model checking extended EF dag-formulas. Combining this lemma with our results in the previous sections, the following theorem can easily be derived.

Theorem 16 *The w-bisimilarity checking problem is solvable in P^{NP}. The problem becomes solvable in P when the one-counter process is fixed.*

We now consider the problem of computing a *symbolic representation of the w-bisimulation relation* for the w-bisimilarity checking problem, i.e., a function mapping each state t in the given finite system and each control location q of the one-counter process to a representation of all $n \in \mathbb{N}$ such that $(q, n) \approx t$. If we are allowed to represent subsets of $\mathbb{N}$ as MMA formulas, it also follows that such representations can be computed in polynomial time.

We now proceed to lower bounds. We can show that w-bisimilarity checking is P^{NP}-hard, even for one-counter nets. Our result can be seen as a substantial improvement of Kučera's DP lower bound proof [11] for the w-bisimilarity checking problem. (The complexity class DP is located in the second level of the boolean hierarchy, which in turn is subsumed in $\mathsf{P}^{\mathsf{NP}[\log]}$; see [17].) In the following, we use a standard way of describing w-bisimulations using simple pebble games between Attacker and Defender (e.g. see [7, 15]). Such games are nothing but Ehrenfeucht-Fraïssé games for the extended EF logic in which Defender attempts to show that two transition systems are w-bisimilar, while Attacker tries to show otherwise.

Theorem 17 *The problem of checking weak-bisimulation between a given one-counter net and a given finite system is hard for P^{NP}.*

Proof Sketch.: The proof of Theorem 17 uses similar techniques as the proof of Theorem 13, but is substantially more involved. We reduce DSAT to the w-bisimulation checking problem by constructing a suitable one-counter net and a finite system. The finite system contains (among many others) states P_1 and P_2, and the one-counter net contains control locations q_i for $i \in [n]$ such that the following statements are equivalent:

- $\langle F_1, \ldots, F_i \rangle \in$ DSAT.
- $P_1 \approx (q_i, l)$ for all $l \in \mathbb{N}$.
- P_2 is not w-bisimilar to (q_i, l) for some l in $\mathbb{N}$.

In the particular case of $i = n$, this is the reduction we are looking for. The idea is that checking the truth of every subformula $\exists Z_i F_i(x_1, \ldots, x_{i-1}, Z_i)$ of the DSAT problem is encoded into a complex w-bisimulation game for $P_1 \approx (q_i, l)$. In this game, the defender player gets to choose (by a long $\overset{\tau}{\Rightarrow}$ move) a natural number l' which is stored in the one-counter net. This number encodes (by Gödel encoding) the assignment of values to the boolean variables in the block Z_i. Later in the game, these values can be tested by special counter-decreasing loops, which implement divisibility tests on l'. The variables x_k are treated differently, because they depend on other subformulae F_k with smaller index numbers $k < i$. If the value of some x_k (for $k < i$) needs to be tested, then the w-bisimulation game jumps to some subgame which tests either $P_1 \approx (q_k, l')$ or $P_2 \approx (q_k, l')$, depending on whether the value of x_k is claimed as true or false. This efficient way of re-using subgames for smaller indices $k < i$ as building blocks corresponds to an efficient way of re-using sub-formulae in the dag-representation of EF-formulae in the proof of Theorem 13.

The main technical difficulty of the proof is to restrict the freedom of the players in the w-bisimulation game, so that they exactly make the choices needed in the verification game for the formula $\exists Z_i F_i(x_1, \ldots, x_{i-1}, Z_i)$ in the right step, and do not make a move that is reserved for the other player. This is tricky, because of the asymmetry of the two compared systems, one infinite-state one-counter process and a finite system. In particular, isomorphic copies of the finite system are replicated in the control locations of the one-counter net. ■

In contrast to Kučera's DP lower bound [11] which holds for a fixed one-counter net, our proof of Theorem 17 requires that the finite system is not fixed. Nevertheless, we can show that the w-bisimilarity checking problem for fixed finite systems is harder than DP.

Theorem 18 *There exists a fixed finite system for which the w-bisimilarity checking problem is hard for $\mathsf{P}^{\mathsf{NP}[\log]}$ even for one-counter nets.*

Proof Sketch.: The proof is done by a reduction from INDEX-ODD, which uses a construction from [11] as a subroutine. ■

VIII. Future work

As mentioned in the introduction, model checking the modal μ-calculus over one-counter processes is in PSPACE by [20]. PSPACE-hardness can both be shown when fixing the one-counter process and when fixing the μ-calculus formula. The latter follows quite immediately from PSPACE-hardness of the emptiness problem for alternating finite automata over a unary alphabet [5, 10]. However, the precise complexity for the temporal logic CTL, extending EF logic and being a fragment of the modal μ-calculus, seems to be worth investigating, lying somewhere between $\mathsf{P}^{\mathsf{NP}[\log]}$ and PSPACE. Another interesting extension of EF-logic could be the first-order logic FO(R) with reachability operator and its two variable fragment. It was recently shown [26] that these extensions are both PSPACE-complete. We would also like to investigate the precise complexity of model checking EF-logic over one-counter processes when formulas are represented as trees (or additionally when they are fixed). Our results imply that it is between $\mathsf{P}^{\mathsf{NP}[\log]}$ and P^{NP}. The same complexity gap remains for weak bisimilarity of one-counter processes against fixed finite systems.

Acknowledgment

We thank Dietrich Kuske, Leonid Libkin, Markus Lohrey, and Christian Mathissen for their comments.

References

[1] A. Bouajjani, J. Esparza and O. Maler. Reachability analysis of pushdown automata: application to model checking. In *CONCUR'97*, pages 135–150.

[2] T. Cachat. Uniform Solution of Parity Games on Prefix-Recognizable Graphs. *ENTCS* 86(6): (2002).

[3] M. Chrobak. Finite Automata and Unary Languages, *Theor. Comput. Sci.* 47(3):149–158 (1986).

[4] J. Esparza, D. Hansel, P. Rossmanith and S. Schwoon. Efficient Algorithms for Model Checking Pushdown Systems. In *CAV'00*, pages 232–247.

[5] M. Holzer. On emptiness and counting for alternating finite automata. In *DLT'95*, pages 88-97.

[6] P. Jančar. Decidability of Bisimilarity for One-Counter Processes. *Inf. Comput.* 158(1): 1–17 (2000).

[7] P. Jančar and A. Kučera. Equivalence Checking on Infinite-State Systems: Techniques and Results. *TPLP* 6(3):227–264 (2006).

[8] P. Jančar, A. Kučera and R. Mayr. Deciding Bisimulation-Like Equivalences with Finite-State Processes. *TCS* 258(1–2):409–433 (2001).

[9] P. Jančar, A. Kučera, F. Moller and Z. Sawa. DP Lower bounds for equivalence-checking and model-checking of one-counter automata. *Inf. Comput.* 188(1): 1–19 (2004).

[10] P. Jančar and Z. Sawa. A note on emptiness for alternating finite automata with a one-letter alphabet. *IPL* 104(5):164–167 (2007).

[11] A. Kučera. Efficient Verification Algorithms for One-Counter Processes. In *ICALP'00*, pages 317–328.

[12] A. Kučera and R. Mayr. On the Complexity of Semantic Equivalences for Pushdown Automata and BPA. In *MFCS'02*, pages 433-445.

[13] A. Martinez. Efficient computation of regular expressions from unary NFAs. In *DCFS'02*, pages 216–230.

[14] R. Mayr. On the Complexity of Bisimulation Problems for Pushdown Automata. In *IFIP TCS'00*, pages 474–488.

[15] R. Mayr. Undecidability of Weak Bisimulation Equivalence for 1-counter Processes. In *ICALP'03*, pages 570–583.

[16] R. Milner. *Communication and Concurrency*. Prentice, 1989.

[17] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[18] N. Piterman and M. Y. Vardi. Global Model-Checking of Infinite-State Systems. In *CAV'04*, pages 387–400.

[19] G. Sénizergues. The Bisimulation Problem for Equational Graphs of Finite Out-Degree. *SIAM J. Comput.* 34(5):1025–1106 (2005).

[20] O. Serre. Parity games played on transition graphs of one-counter processes. In *FOSSACS'06*, pages 337–351.

[21] H. Spakowski and J. Vogel. Θ_p^2-Completeness: A Classical Approach for New Results. In FSTTCS'00, pages 348–360.

[22] J. Srba. Undecidability of Weak Bisimilarity for Pushdown Processes. In *CONCUR'02*, pages 579–593.

[23] J. Srba. Roadmap of Infinite results. Bulletin of the EATCS 78:163–175 (2002). http://www.brics.dk/~srba/roadmap

[24] J. Srba. Visibly Pushdown Automata: From Language Equivalence to Simulation and Bisimulation. In CSL'06, pages 89–103.

[25] A. W. To. Unary finite automata vs. arithmetic progressions. *Tech. report EDI-INF-RR-1306*, Univ. of Edinburgh, 2008.

[26] A. W. To. Model checking FO(R) over one-counter processes and beyond. To appear in *CSL'09*.

[27] K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theor. Comput. Sci.* 51(1–2):53–80 (1978).

[28] I. Walukiewicz. Model checking CTL Properties of Pushdown Systems. In *FSTTCS'00*, pages 127–138.

[29] I. Walukiewicz. Pushdown Processes: Games and Model-Checking. *Inf. Comput.* 164(2):234–263 (2001).