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 $P^{NP[log]}$-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 $P^{NP[log]}$. 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 $\mu$-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 holds even 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 $\mu$-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 $\mu$-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 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 PSPACE-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 PSPACE lower bound is shown by a reduction from DSAT for the case of EF formulas represented as a dag, while the best lower bound for tree-represented EF formulae is PSPACE. In fact, there already exists a fixed EF formula such that model checking one-counter processes is hard for PSPACE. 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 PNP (resp. PNP[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

$$NP \cup coNP \subseteq PNP[log] \subseteq PNP \subseteq PSPACE.$$ 

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

One-counter processes and transition systems: Fix some countable set $\Sigma$ of actions. A transition system is a pair $T = (S, \{a\rightarrow a\mid a \in \Sigma\})$, where $S$ is a set of states and $\rightarrow_a S \subseteq S \times S$ 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

<table>
<thead>
<tr>
<th>Logic</th>
<th>PDA</th>
<th>OCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$-calc.</td>
<td>EXPTIME</td>
<td>PSPACE</td>
</tr>
<tr>
<td>$\mu$-calc., fixed formula</td>
<td>EXPTIME</td>
<td>PSPACE</td>
</tr>
<tr>
<td>$\mu$-calc., fixed ctrl.</td>
<td>EXPTIME</td>
<td>PSPACE</td>
</tr>
</tbody>
</table>

| EF, fixed formula | PSPACE | $\text{PSPACE}^{\text{NP}}$ hard (*) |
| EF, fixed ctrl. | PSPACE | in $\text{P}$ (*) |

Figure 1. Model checking over PDA and OCP

<table>
<thead>
<tr>
<th>Bisimulation type</th>
<th>PDA</th>
<th>OCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\approx$ finite system</td>
<td>PSPACE</td>
<td>$\text{PSPACE}^{\text{NP}}$ (*)</td>
</tr>
<tr>
<td>$\approx$ fixed finite system</td>
<td>PSPACE</td>
<td>$\text{PSPACE}^{\text{NP}[\log]}$ hard (*)</td>
</tr>
<tr>
<td>fixed ctrl. $\approx$ finite system</td>
<td>$\text{P}$ in $\text{P}$ (*)</td>
<td></td>
</tr>
<tr>
<td>$\sim$ finite system</td>
<td>PSPACE</td>
<td>P</td>
</tr>
<tr>
<td>$\sim$ fixed finite system</td>
<td>in $\text{P}$ in $\text{P}$</td>
<td></td>
</tr>
<tr>
<td>fixed ctrl. $\sim$ finite system</td>
<td>P in P</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Bisimulation over PDA and OCP
s_1 \rightarrow_T s_2 \text{ whenever } s_1 \rightarrow_a s_2 \text{ for some } a \in \Sigma. \text{ We define the size of } T \text{ as } |T| = |S| + \sum_{a \in \Sigma} |\rightarrow_a|.

A one-counter process is a tuple \( O = (Q, \delta_0, \delta_{>0}) \), 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 \(|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 \( O = (Q, \delta_0, \delta_{>0}) \) defines a transition system \( T(O) = (Q \times \mathbb{N}, \{ \rightarrow_a \ a \in \Sigma \}) \), where \( (q, n) \rightarrow_a (q', n+k) \) 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 \((q_1,n_1) \rightarrow_a (q_2,n_2) \) for some \( a \in \Sigma \), we also write \((q_1,n_1) \rightarrow_C (q_2,n_2) \).

**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 \( \varphi_i \) is exactly one of the following, either: \( \varphi_i = \text{true}, \varphi_i = \neg \varphi_j \) for some \( j \in [i-1], \varphi_i = (a) \varphi_j \) for some \( a \in \Sigma \) and some \( j \in [i-1], or \varphi_i = \text{EF} \varphi_j \) for some \( j \in [i-1]. For each \( i \in [l] \), we define the size \(|\varphi_i| = 1 \) if \( \varphi_i = \text{true} \) and \(|\varphi_i| = \log(i) \) otherwise. The size of \( \varphi \) is defined as \(|\varphi| = \sum_{i=1}^l |\varphi_i| \).

Define the partial order \( \prec \subseteq [l] \times [l] \) as \((j, i) \in \prec \) if and only if \( \varphi_j \) appears in the definition of \( \varphi_i \). If \( j \prec_i i \), we say that \( \varphi_j \) is a subformula of \( \varphi_i \). Note that \( i \) is minimal with respect to \( \prec_i \) whenever \( \varphi_i = \text{true} \). Observe that \([l], \prec \) is a dag. We call \( \varphi \) an EF tree-formula if moreover \(([l], \prec \) is a directed tree. Next, we define the semantics. For this, let \( T = (S, \{ \rightarrow_a \ a \in \Sigma \}) \) be a transition system. Let us define \( [\varphi_i]_T \subseteq S \) for each \( i \in [l] \) by induction on \( \prec_i \) as follows: \([\text{true}]_T = S, [\neg \varphi_j]_T = S \setminus [\varphi_j]_T, [a] \varphi_j]_T = [\varphi_j]_T \cap \{ s \in [\varphi_j]_T \colon t \in T \colon s \rightarrow_a t \}, \text{ and } [\text{EF} \varphi_j]_T = \{ s \in S \colon \exists t \in [\varphi_j]_T \colon s \rightarrow^*_T t \}. We define \( [\varphi]_T = [\varphi]_T \). 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 \( O \), a state \((q,n) \) of \( T(O) \) with \( n \) given in binary, and an EF dag/tree-formula \( \varphi \).

**Question:** \( (T(O),(q,n)) \models \varphi \)?

We also consider global model checking in this paper, which asks to compute a function mapping each control location \( q \) of \( O \) to a representation of all \( n \in \mathbb{N} \) such that \((T(O),(q,n)) \models \varphi \). We recall the definition of weak bisimulation [7]. Let \( T = (S, \{ \rightarrow_a \ a \in \Sigma \}) \) 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 \( \tau', t' \) such that \( s \rightarrow^*_\tau \tau' \rightarrow^*_\tau t', or a = \tau and s \rightarrow^*_\tau t \). Given two transition systems \( T = (S, \{ \rightarrow_a \ a \in \Sigma \}) \) and \( T' = (S', \{ \neg \alpha \ a \in \Sigma \}) \), 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 \( s \rightarrow_a t \) for some \( t \in S \), then \( s' \rightarrow^*_\alpha t' \) for some \( t' \in S' \) such that \((t,t') \in R \), and (ii) if \( s' \rightarrow^*_\alpha t' \) 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 \( O \), a state \((q,n) \) of \( T(O) \) with \( n \) given in binary, a finite system \( T \), and a state \( t \) of \( T \).

**Question:** \((q,n) \approx t \)?
(1) \[ \equiv m \text{ mod } n \] = \{ k \in \mathbb{N} \mid k \equiv m \text{ mod } n \},

(2) \[ \sim n \] = \{ k \in \mathbb{N} \mid k \sim n \},

(3) \[ \neg \alpha_j \in \mathbb{N} \setminus [\alpha_j] \},

(4) \[ \alpha_j \land \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, \\ 0 & \text{otherwise} \end{cases} \}

(7) \[ m \sim \max (\alpha_j, n) = \{ k \in \mathbb{N} \mid k \sim \max (\alpha_j \cap [0, n]) \} \}

Proof: We prove the lemma by induction on \( \prec^+ \).

For the induction base, assume \( i \) is minimal with respect to \( \prec^+ \), i.e. \( \alpha_i \) is atomic.

Case \( \alpha_i \) is \( \equiv m \text{ mod } n \) 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 \( \alpha_i \) is \( \sim n \), where \( \sim \in \{ \leq, \geq \} \) and \( n \in \mathbb{N} \). 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 \( \prec^+ \). For this, we make a case distinction according to \( \alpha_i \).

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 \( L_i \equiv L_j \).

Case \( \alpha_i = \alpha_j \land \alpha_k \) 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 \( n_1 \equiv n_2 \equiv 0 \mod L_i \). Then we have \( n_1 \in [\alpha_j] \) if and only if \( n_1 \in [\alpha_j] \) and \( n_2 \in [\alpha_k] \). 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 \( n_2 \in [\alpha_j] \).

Case \( \alpha_i \) 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 \( \emptyset \) 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 \( n \leq j \cdot L_j + \nu_j + \nu_i \) by induction hypothesis. Observe that the latter is less than or equal to \((i-1) \cdot L_i + \nu_i \)

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 \([\alpha_j]\) either equals \( \emptyset \) or \( \emptyset \), the implication trivially holds.

Case \( \alpha_i \) 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 \( n \leq \nu_j \) by definition. Second, all naturals exceeding \( n \) (in particular those exceeding \( \nu_j \)) either all satisfy \( \alpha_i \) or all do not, depending on \( \sim \). Thus, the implication holds.

Case \( \alpha_i \) 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 \([\alpha_j]\) either equals \( \mathbb{N} \) or \( \emptyset \), the implication holds trivially.

Case \( \alpha_i = \alpha_j \lor 1 \) for some \( j \in [i-1] \) and some \( \lor \in \{\sim, \geq\} \). 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.

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

\[ n_1 \equiv n_2 \mod L_i \Rightarrow (n_1 \in [\alpha_i] \iff n_2 \in [\alpha_i]) \]
Lemma 2 The following problem is computable in polynomial time:

**INPUT:** An extended MMA dag-formula $\alpha$.

**OUTPUT:** A MMA dag-formula $\beta$ 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 $\alpha$.

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

Proof: First, it is easy to see that if $\alpha$ is an MMA dag-formula with no occurrence of min and max then checking whether $n_0 \in [\alpha]$ can be done in 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 $\beta$ such that $[\beta] = [\alpha]$. Assume $\beta = (\beta_i)_{i \in [0]}$. In a second step, we eliminate min and max operators that occur in $\beta$ inductively on $\prec^+$.

For this, let $i \in [0]$ be minimal with respect to $\prec^+$ such that the definition $\beta_i$ contains either min or max. In case $\beta_i = \min j$, we know that for each $n_1, n_2 > j \cdot L_j + \nu_j$ with $n_1 \equiv n_2 \mod L_j$ 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 be 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 $\beta_j$ nor the definition of $\beta_k$ contains min or max, for each $k \prec^+ j$. 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 $\beta$ by replacing $\beta_j$’s previous definition $\sim \min \beta_j$ by $\sim \mu$.

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

We repeat this replacement process until $\beta$ 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 $\alpha_i \pm k$, 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 $\alpha$, 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 $O = (Q, \delta_0, \delta_{>0})$. 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” $O$ with $\lambda$-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 $O'$ 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 $T(O')$ in terms of few small arithmetic progressions. (2) For every EF dag-formula $\varphi$ in which $\lambda$ does not occur we have $(T(O'), s) \models \varphi$ if and only if $(T(O'), s) \models \varphi$ for every state $s \in Q \times \mathbb{N}$.

A path in $T(O)$ is a non-empty finite sequence of states $\pi = (q_1, n_1) \rightarrow_{0} (q_2, n_2) \cdots \rightarrow_{0} (q_k, n_k)$. We call $\pi$ mountain, if $n_1 = n_k$ and $n_i \geq n_1$ for each $i \in [k]$. We call $\pi$ zero, if $n_i = 0$ for some $i \in [k]$, otherwise we call $\pi$ positive. Let $(q_1, n_1), (q_2, n_2) \in Q \times \mathbb{N}$ be states. Then, we write $(q_1, n_1) \downarrow_{0} (q_2, n_2)$ (resp. $(q_1, n_1) \uparrow_{0} (q_2, n_2)$) whenever $(q_1, n_1) \rightarrow_{0} (q_2, n_2)$ 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 $\lambda$-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 $\delta$ 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 $O' = (Q, \delta_0, \delta_{>0}')$ denote the unique one-counter process that we obtain from $O$ 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. $s \rightarrow_{s}^{*} t$.
2. $s \rightarrow_{s}^{\lambda} t$.
3. There exists some state $u \in Q \times \mathbb{N}$ such that $s \downarrow_{s}^{\lambda} u$ and $u \uparrow_{s}^{\lambda} t$. 


Observe that if \((q_1, n_1) \uparrow^*_\varphi (q_2, n_2)\) and \(n_1 > 0\), then also \((q_1, n_1 + i) \uparrow^*_\varphi (q_2, n_2 + i)\) for each \(i \in \mathbb{N}\). Similarly, if \((q_1, n_1) \downarrow^*_\varphi (q_2, n_2)\) and \(n_2 > 0\), then also \((q_1, n_1 + i) \downarrow^*_\varphi (q_2, n_2 + i)\) 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 zero paths:

\[
\Delta^0_1(q_1, q_2) = \{d \in \mathbb{N} \mid (q_1, 1) \uparrow^*_\varphi (q_2, d + 1)\}
\]

\[
\Delta^{\geq 0}_1(q_1, q_2) = \{d \in \mathbb{N} \mid (q_1, d + 1) \downarrow^*_\varphi (q_2, 1)\}
\]

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

\[
\Delta^0_1(q_1, q_2) = \{d \in \mathbb{N} \mid (q_1, 1) \uparrow^*_\varphi (q_2, d)\}
\]

\[
\Delta^{\geq 0}_1(q_1, q_2) = \{d \in \mathbb{N} \mid (q_1, d) \downarrow^*_\varphi (q_2, 1)\}
\]

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 \(\Delta^0_1(q_1, q_2), \Delta^{\geq 0}_1(q_1, q_2), \Delta^0_2(q_1, q_2), \Delta^{\geq 0}_2(q_1, q_2)\) 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 \((q_1, n) \uparrow^*_\varphi (q_3, 1)\uparrow^*_\varphi (q_2, n)\) for some \(q_3 \in Q\), then also \((q_1, n + i) \uparrow^*_\varphi (q_3, 1 + i) \uparrow^*_\varphi (q_2, n + i)\). Therefore, we define \(\nabla(q_1, q_2) \in \mathbb{N} \cup \{\infty\}\) to be

\[
\min\{n > 0 \mid \exists q_3 \in Q : (q_1, n) \uparrow^*_\varphi (q_3, 1) \uparrow^*_\varphi (q_2, n)\}.
\]

Observe that \(\nabla(q_1, q_2) = 1\) 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 \(T(\varphi')\) if and only if \(n \in \Delta^0_1(q, q')\) and \(n' \in \Delta^0_1(q', q'')\) for some \(q'' \in Q\).

The next lemma characterizes positive paths.

**Lemma 9** Assume \(n \leq n'\). Then there exists a positive path in \(T(\varphi')\) from \((q, n)\) to \((q', n')\) if and only if \(n \geq \nabla(q, q')\) and \(n' - n \in \Delta^0_1(q', q'')\) for some \(q'' \in Q\).

Assume that \(n \geq n'\). Then there exists a positive path from \((q, n)\) to \((q', n')\) in \(T(\varphi')\) if and only if \(n' \geq \nabla(q', q'')\) and \(n - n' \in \Delta^0_1(q', q'')\) for some \(q'' \in Q\).

V. A TRANSLATION TO MMA

For the rest of this section, let us fix a one-counter process \(O = (Q, \delta_0, \delta_{>0})\) and an EF dag-formula \(\varphi = (\varphi_i)_{i \in \llbracket l \rrbracket}\). 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 \(\alpha = (\alpha(q_j, j))_{(q_j, j) \in Q \times \llbracket l \rrbracket}\) such that \([\alpha(q_j, j)] = \{n \in \mathbb{N} \mid (T(O), (q, n)) \models \varphi_j\}\) for each \((q, j) \in Q \times \llbracket l \rrbracket\).

First, we saturate \(O\) in the sense of Section IV in polynomial time. Then, we apply Lemma 6 and compute in polynomial time the sets \(\Delta^0(q, q'), \Delta^0(q, q''), \Delta^{\leq 0}(q, q')\), \(\Delta^{\leq 0}(q, q'')\), and \(\Delta^{\leq 0}(q, q'')\), 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 \(\nabla(q, q') \in [n_\varphi] \cup \{\infty\}\) for each \(q, q' \in Q\), where \(n_\varphi \in O(|Q|^2)\).

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

**Base.** Assume \(j\) is minimal with respect to \(\prec^\_\alpha\). Then \(\varphi_j = \text{true}\) and we put \(\alpha(q_j, j) = (\geq 0)\) for each \(q \in Q\).

**Step.**

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

Assume \(\varphi_j = \varphi_{j1} \land \varphi_{j2}\) for some \(j1, j2 \in [j - 1]\). Then we put \(\alpha(q_j, j) = \alpha(q_{j1}, j1) \land \alpha(q_{j2}, j2)\) for each \(q \in Q\).

Assume \(\varphi_j = (a)\varphi_{j}^\prime\) for some \(a \in \Sigma\) and some \(j' \in [j - 1]\). By induction hypothesis, we have

\[
\alpha(q_{j'}, j') = \{n \in \mathbb{N} \mid (q_{j'}, n) \models \varphi_{j'}\}
\]

for each \(q' \in Q\). By putting \(\hat{\varphi} = \text{and}\) and \(\hat{\varphi} = \text{or}\), we define \(\alpha(q_j, j)\) as the conjunction of

\[
(\geq 0) \to \left( \bigvee_{q' \in Q, (q, n, q', q'') \in \delta_0} \alpha(q_{j'}, j') - 1 \land \bigvee_{q' \in Q, (q, n, q', q'') \in \delta_0} \alpha(q_{j'}, j') \right)
\]

and

\[
(> 0) \to \left( \bigvee_{q' \in Q, (q, n, q', q'') \in \delta_0} \alpha(q_{j'}, j') \land \bigvee_{q' \in Q, (q, n, q', q'') \in \delta_0} \alpha(q_{j'}, j') \right).
\]

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

\[
[\alpha(q_{j'}, j')] = \{n \in \mathbb{N} \mid (q_{j'}, n) \models \varphi_{j'}\}
\]

By Lemma 1 we know that for each \(n_1, n_2 \in \mathbb{N}\) which exceed a threshold \(l(q_{j'}, j')\) such that \(n_1 \equiv n_2 \mod l(q_{j'}, j')\) we
have $n_1 \in [\alpha(q', r')]$ if and only if $n_2 \in [\alpha(q', r')]$. Note that this implies that $[\alpha(q', r')]$ is infinite if and only if there exists some $n \in [\alpha(q', r')]$ such that $t(q', r') < n \leq t(q', r') + L(q', r')$.

Let us now fix an arithmetic progression $a + b\mathbb{N}$ with $b > 0$ that is a subset of some of the sets $\Delta^0_1(q, q'), \Delta^0_2(q, q')$, or $\Delta^0_3(q, q')$. 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 $\alpha(q', r')$ and that are congruent $c$ modulo $b$. Now observe that the latter is the case exactly whenever there exists some $n \in [\alpha(q', r')]$ such that $t(q', r') < n \leq t(q', r') + L(q', r')$ and moreover $n \equiv c \mod \gcd(b, L(q', r'))$. Let us define the auxiliary MMA formula

$$\psi(q', j', c, b) = \alpha(q', r') \land (\exists c \mod \gcd(b, L(q', r')))$$

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

$$t(q', r') < \max(\psi(q', j', c, b), t(q', r') + L(q', r')).$$

Next, we aim at defining the set

$$\{n \in \mathbb{N} \mid \exists n' \in \mathbb{N} \colon (q, n) \rightarrow_0 (q', n'), (q', n') \models \varphi_j\}$$

in terms of an extended MMA formula. For this, assume that there is a path $\pi$ from $(q, n)$ to $(q', n')$ in $T(\mathbb{O})$ such that $(q', n') \models \varphi_j$, where $n' \in \mathbb{N}$. We distinguish three (not necessarily distinct) cases. Either (1) $\pi$ is positive and $n \leq n'$, (2) $\pi$ is positive and $n \geq n'$, or (3) $\pi$ 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. $\pi$ is positive and $n \leq n'$. Then $n \geq \Delta(q, q'')$ and $n' - n \in \Delta^0_1(q', q'')$ for some $q'' \in Q$ by Lemma 9. Thus, $(n' - n) \in a + b\mathbb{N}$ for some arithmetic progression $a + b\mathbb{N} \subseteq \Delta^0_1(q', q'')$. 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) or (ii) or (iii) infinitely 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

$$\inf(q'', j'', c, b) \land \geq \nabla(q, q'') \land \equiv (c - a) \mod b.$$
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 \( \varphi \), we can compute in polynomial time for each control location \( q \) of \( \mathcal{O} \) an MMA dag-formula \( \alpha(q) \) such that \( \alpha(q) \equiv (T(\mathcal{O}), (q, n)) \models \varphi \).

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

**Corollary 11** The following problem is in \( \text{P}^{\text{NP}} \):

**INPUT:** A one-counter process \( \mathcal{O} = (Q, q_0, \delta_{>0}) \), a state \( (q, n) \in Q \times \mathbb{N} \) with \( n \) given in binary, and an EF dag-formula \( \varphi \).

**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 \( \mathcal{O} = (Q, q_0, \delta_{>0}) \) the following problem is in \( \text{P} \):

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

**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 \( \text{P}^{\text{NP}} \) lower bound for model checking EF logic dag-formulas. Then we prove a \( \text{P}^{\text{NP}[\log]} \) 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 \( \text{P}^{\text{NP}} \) lower bound for the problem of model checking EF logic dag-formulas. The problem we will reduce from is the well-known \( \text{P}^{\text{NP}} \)-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 \( i \in [n] \):

\[
\sigma(x_i) = 1 \iff \exists Z_i F_i(x_1, \ldots, x_{i-1}, Z_i).
\]

Notice that imposing the constraint in (1) ensures that there exists a unique assignment \( \sigma \). 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 \( \text{P}^{\text{NP}} \)-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 \( \varphi \) that mimic the structure of the given sequence of formulas. By using a well-known G"odel encoding technique that was also used in [9, 11], we can view each natural \( m \in \mathbb{N} \) as the truth assignment \( \alpha_m : \{x_1, \ldots, x_n\} \to \{0, 1\} \), 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 \( \varphi \) 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 \( F_j \)'s, 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 \( 3\text{-CNF} \), 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 \( \text{P}^{\text{NP}[\log]} \)-complete. A \( \text{P}^{\text{NP}[\log]} \) 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 \( \text{P}^{\text{NP}[\log]} \)-hardness [27, Theorem 5.2] (see also [21, Lemma 7]). The reduction for the following theorem can be achieved via G"odel encoding as in the proof of Theorem 13.

**Theorem 14** There exists a fixed EF formula \( \varphi \) such that model checking \( \varphi \) over one-counter nets is \( \text{P}^{\text{NP}[\log]} \)-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 \text{min} and \text{max} operators is hard for \( \text{P}^{\text{NP}[\log]} \). The reason for this is that the \text{min} and \text{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 \text{min-free} and \text{max-free} is \( \text{P}^{\text{NP}} \). 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 \( \text{w-bisimilarity} \) checking problem. We show that the general problem is \( \text{P}^{\text{NP}} \)-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 $P^{NP[log]}$-hard.

We extend our EF logic with the following operator $EF(\tau^*)$, which we will also abbreviate as $EF'$. The meaning is that for a transition system $T = (S, \{\tau_a | a \in \Sigma\})$ and a state $s \in S$ we have $T, s \models EF' \phi$ if and only if there exists $s' \in S$ such that $s \xrightarrow{\tau} s'$ and $T, s' \models \phi$. Note that our $P^{NP}$ upper bound easily carries over to this extended EF logic by simply restricting ourselves to $\tau$-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 $T_1 = (S_1, \{\frac{1}{2^a} | a \in \Sigma\})$ be a transition system and $T_2 = (S_2, \{\frac{1}{a} | a \in \Sigma\})$ be a finite transition system with $k$ states. Then, given any state $s_2 \in S_2$, we can construct a dag-formula $\varphi_{s_2, T_2}$ 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 $P^{NP[log]}$; 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 $\rightarrow$ 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 subformulas $F_k$ 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 $P^{NP[log]}$ 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 $\mu$-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 $\mu$-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 $\mu$-calculus, seems to be worth investigating, lying somewhere between $\text{P}^{\text{NP}[\log]}$ 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 $\text{P}^{\text{NP}[\log]}$ and PNP. 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


