Model checking $\mathsf{FO}(\mathrm{R})$ over one-counter processes and beyond

Anthony Widjaja To

LFCS, School of Informatics, University of Edinburgh
anthony.w.to@ed.ac.uk

Abstract. One-counter processes are pushdown processes over a singleton stack alphabet (plus a stack-bottom symbol). We study the problems of model checking asynchronous products of one-counter processes against 1) first-order logic $\mathsf{FO}(\mathrm{R})$ with reachability predicate, 2) the finite variable fragments $\mathsf{FO}^k(\mathrm{R})$ ($k \geq 2$) of $\mathsf{FO}(\mathrm{R})$, 3) EF-logic which is a fragment of $\mathsf{FO}^2(\mathrm{R})$, and 4) all these logics extended with simple component-wise synchronizing predicates. We give a rather complete picture of their combined, expression, and data complexity. To this end, we show that these problems are poly-time reducible to two syntactic restrictions of Presburger Arithmetic, which are equi-expressive with first-order modulo counting theory of $(\mathbb{N}, <)$, for which we give optimal quantifier elimination procedures. In particular, these problems are all shown to be in PSPACE, which is in sharp contrast to the closely related problem of model checking $\mathsf{FO}(\mathrm{R})$ over pushdown processes (with one stack) which has nonelementary complexity. Finally, we apply our proof method to give a fixed automatic (and so rational) graph whose modal logic theory has nonelementary complexity, solving a recently posed open question.

1 Introduction

Pushdown automata (PDA) are a natural model for sequential programs with recursive calls and their model checking problems have been studied extensively. It is well-known that, over PDA, the problems of model checking first-order logic with reachability predicate $\mathsf{FO}(\mathrm{R})$ and monadic second-order logic MSO are decidable in nonelementary recursive time [20]. In fact, PDA (even with one control state) can easily generate natural numbers $(\mathbb{N}, +1)$ with a successor relation (a.k.a. S1S) and the infinite binary tree with two successor relations (a.k.a. S2S). Since the $\mathsf{FO}(\mathrm{R})$ theory of S2S and the MSO theory of S1S have nonelementary complexity [4, 24], the same lower bound can be deduced for PDA. In contrast, when considering modal and temporal logics — such as EF-logic, LTL, CTL, and μ-calculus — the complexity of model checking PDA is at most $\mathsf{EXPTIME}$ [3, 5, 27, 28].

One-counter processes (OCPs) are PDA over a singleton stack alphabet (plus a non-removable stack-bottom symbol). The problems of model checking basic modal logic, EF-logic, and μ-calculus over one-counter processes have been studied, and their complexity are lower than the corresponding problems for general

PDA [8, 10, 11, 23]. On the other hand, since OCPs can easily generate S1S, it follows from [24] that model checking MSO over OCPs is nonelementary. The complexity of model checking $\mathsf{FO}(\mathrm{R})$ over OCPs is, however, unknown. Unlike the case of general PDA, it is easy to show that OCPs cannot generate a graph isomorphic to S2S. Furthermore, the FO theory of S1S with linear order is only PSPACE-complete [7, 24].

PDA are well-known to be incapable of modeling concurrent programs. One common way to obtain concurrent behavior from pushdown processes is to consider *(finite) products* of graphs generated by PDA. An *asynchronous product* of k PDA $\mathcal{P}_1, \ldots, \mathcal{P}_k$ can be construed as a concurrent system with processes $\mathcal{P}_1, \ldots, \mathcal{P}_k$, each behaving independently (i.e. the processes do not interact). Therefore, reachability for an asychronous product of PDA can trivially be reduced to the reachability problems for each of its components. Note that taking synchronized products — the most general notion of products — of PDA (resp. OCPs) easily yield a model that is as powerful as Turing machines (resp. Minsky counter machines). There are several reasons for studying model checking problems over simple models of concurrent programs such as asynchronous products of PDA and OCPs. First, Wöhrle and Thomas [29] have recently shown that, when combined with logics such as $\mathsf{FO}(\mathrm{R})$ and EF-logic, asynchronous products are powerful enough for modeling *a finite amount of synchronization* (synchronization can be embedded in the formulas). Second, asynchronous products of PDA and OCPs are some of the most basic nontrivial concurrent models that are subsumed by more complex models such as ground tree rewrite systems [16], PAD [18], automatic graphs [2], and rational graphs [19]. Some open problems in the more general settings (such as the complexity of model checking EF-logic [18, 26]) seem difficult already in the restricted settings.

In this paper, we consider model checking problems of OCPs, as well as asynchronous products of OCPs (ΠOCPs), with respect to specifications in 1) $\mathsf{FO}(\mathrm{R})$, 2) the k-variable fragments $\mathsf{FO}^k(\mathrm{R})$ ($k \geq 2$) of $\mathsf{FO}(\mathrm{R})$, and 3) **EF**-logic which is a fragment of $\mathsf{FO}^2(\mathrm{R})$. We also study these logics extended with simple componentwise synchronizing unary predicates testing whether component i and j are the same elements, which we denote by $\mathsf{FO}_{\mathrm{S}}(\mathrm{R})$, $\mathsf{FO}^k_{\mathrm{S}}(\mathrm{R})$, and EF_{S}-logic. We give a rather complete picture of their *combined complexity* (i.e. inputs consist of systems and specifications), *expression complexity* (i.e. inputs consist only of specifications with a fixed system), and *data complexity* (i.e. inputs consist only of systems with a fixed specification).

	$\mathsf{FO}(\mathrm{R})$ $\mathsf{FO}^4(\mathrm{R})$	$\mathsf{FO}^2(\mathrm{R})$	EF-logic
Combined	PSPACE	PSPACE	in P^{NP} & $\mathsf{P}^{\mathsf{NP}[\log]}$-hard [8]
Expression	PSPACE	in P	in P [8]
Data	PH	PH	in P^{NP} & $\mathsf{P}^{\mathsf{NP}[\log]}$-hard [8]

Table 1. Results for OCPs.

	$\mathsf{FO}(\mathrm{R})$ $\mathsf{FO}^4(\mathrm{R})$ $\mathsf{FO}_{\mathrm{S}}(\mathrm{R})$	$\mathsf{FO}^2(\mathrm{R})$	EF-logic	EF_{S}-logic
Combined	PSPACE	PSPACE	PSPACE	PSPACE
Expression	PSPACE	in P	in P	PSPACE
Data	PH	PH	in P^{NP} & $\mathsf{P}^{\mathsf{NP}[\log]}$-hard	PH

Table 2. Results for ΠOCPs.

Our results are summarized in Table 1 and Table 2 together with the recent result from [8]. In particular, all our results are within PSPACE, in contrast to PDA whose expression complexity for $\mathsf{FO}(\mathrm{R})$ is nonelementary [4]. Our upper bounds are shown by first introducing two syntactic restrictions $\mathcal{L}$ and $\mathcal{L}'$ of Presburger Arithmetic, for which we give optimal quantifier elimination procedures, and showing that the ΠOCPs model checking problems are poly-time reducible to either $\mathcal{L}$ or $\mathcal{L}'$. Note that, to obtain a sharp upper bound, we cannot consider only OCPs (without products) and apply Feferman-Vaught type of composition methods (e.g. see [17, 22, 29]) as the resulting algorithm will run in time that is nonelementary in the formula size. Concerning our lower bound results, in contrast to the result in [8] that model checking EF-logic is in P^{NP}, data complexity of $\mathsf{FO}^2(\mathrm{R})$ over OCPs is already hard for every level of PH. On the other hand, the expression complexity of $\mathsf{FO}^2(\mathrm{R})$ over ΠOCPs is in P. This generalizes one of the key results in [8] that the expression complexity of EF-logic over OCPs (without products) is in P. However, for each $k > 3$, we can show that the expression complexity of $\mathsf{FO}^k(\mathrm{R})$ is PSPACE-complete already for OCPs. Also, notice that the combined complexity of EF-logic becomes PSPACE, which holds already for products of two OCPs. Finally, notice that adding simple synchronization relations to EF-logic causes the expression and data complexity to increase significantly. In fact, we shall use its proof method to give a fixed automatic (and so rational) graph whose modal logic theory has nonelementary complexity, answering a recently posed question in [1, 26].

The paper is organized as follows. We fix some necessary notations and definitions in Section 2. In Section 3 we define two syntactic restrictions $\mathcal{L}$ and $\mathcal{L}'$ of Presburger Arithmetic and prove that model checking problems for OCPs and ΠOCPs are poly-time reducible checking formulas in these restricted logics. In Section 4 we give optimal quantifier elimination procedures for $\mathcal{L}$ and $\mathcal{L}'$ and deduce optimal upper bounds for all problems in Table 1 and Table 2. We prove our lower bounds results for OCPs and ΠOCPs in Section 5. In Section 6 we give a fixed automatic graph whose modal logic has nonelementary complexity. Finally, we conclude in Section 7 with future work. Due to space constraints, most proofs have been relegated to the full version.

2 Preliminary

General Notations Let $\mathbb{Z}$ denote the set of integers. Let $\mathbb{N} = \mathbb{Z}_{\geq 0}$. For $i, j \in \mathbb{N}$, we use $[i, j]$ to denote $\{i, i+1, \ldots, j\}$. As usual, the notations (i, j), $(i, j]$, and

$[i,j)$ denote the subsets of $[i,j]$ with the appropriate endpoints omitted. We use $a+b\mathbb{N}$ to mean the arithmetic progression $\{a+bk : k \in \mathbb{N}\}$. Given n sets $S_1,\ldots,S_n$, their product $\Pi_{i=1}^n S_i$ is the set $\{(s_1,\ldots,s_n) : \forall i \in [1,n](s_i \in S_i)\}$. As usual, for a set S, we use S^* to denote the set of all finite strings over S. In the sequel, we use p_i to denote the ith prime number, e.g., $p_1 = 2$.

Computational Complexity We assume familiarity with complexity classes L, P, Σ_k^p, PH, NP, PSPACE and $\mathsf{EXPTIME}$ (see [12]). The class P^{NP} (resp. $\mathsf{P}^{\mathsf{NP}[\log]}$) consists of problems solvable by deterministic poly-time Turing machines with polynomially (resp. logarithmically) many calls to an NP oracle [13]. As usual, for each $n \in \mathbb{Z}_{>0}$, we use n-$\mathsf{EXPTIME}$ to denote the class of problems solvable in n-fold exponential time. A problem is said to be *elementary* if it is in n-$\mathsf{EXPTIME}$ for some $n \in \mathbb{Z}_{>0}$; otherwise, it is *nonelementary*. We assume familiarity with the alternating Turing machines (ATMs) [12]. Recall that the class of problems solvable by logspace (resp. poly-time) ATMs coincides with P (resp. PSPACE). An ATM with not-states (i.e. states that invert the outcome of the run) can be simulated by one without them without any extra space or time [12].

Graphs Let Σ be a finite set of *actions*. A Σ-labeled *graph* is a tuple $G = (V, \{E_a\}_{a\in\Sigma})$, where V is a set of *vertices* of G, and each $E_a \subseteq V \times V$ is a binary *edge relation* (a.k.a. transition relation) over V. Whenever Σ is clear from the context, we shall omit mention of Σ. We also denote E_a by $\rightarrow_a$, and write $v \rightarrow_a v'$ instead of $(v,v') \in \rightarrow_a$. For each $\Sigma' \subseteq \Sigma$, the transitive closure of $\left(\bigcup_{a\in\Sigma'} \rightarrow_a\right)$ is denoted by $\rightarrow^*_{\Sigma'}$. We also write $\rightarrow$ for $\rightarrow_\Sigma$.

Asynchronous products Let $\Sigma_1,\ldots,\Sigma_r$ be r pairwise disjoint sets of actions. Let Σ be their union. For each $i \in [1,r]$, let $G_i = (V_i, \{E_a\}_{a\in\Sigma_i})$ be a Σ_i-labeled graph. An *asynchronous product* of $G_1,\ldots,G_r$ is the graph $\Pi_{i=1}^r G_i := (V, \{\overline{E}_a\}_{a\in\Sigma})$, where $V := \Pi_{i=1}^r V_i$ and, whenever $a \in \Sigma_i$, $\overline{u} = (u_1,\ldots,u_r)$, and $\overline{v} = (v_1,\ldots,v_r)$, we have $(\overline{u},\overline{v}) \in \overline{E}_a$ iff $(u_i,v_i) \in E_a$ and $u_j = v_j$ for all $j \neq i$. Intuitively, the product is "asynchronous" as each edge relation in $\Pi_{i=1}^r G_i$ changes at most one component in each vertex of $\Pi_{i=1}^r G_i$, i.e., causing no interaction between different components. See [22, 29] for more details.

Other logical structures We define $\mathrm{S1S}_<$ to be the structure $(\mathbb{N},<)$, i.e., natural numbers with a (binary) linear order relation $<$. The structure $(\mathbb{N},+)$ consists of natural numbers with a ternary relation $+$ interpreted as additions over $\mathbb{N}$. See [12, 25] for more details.

One-counter processes A *one-counter process* (OCP) over an action alphabet Σ is a tuple $\mathcal{O} = (Q, \delta^+, \delta^0)$, where Q is a finite set of *control states*, $\delta^+ \subseteq Q\times\Sigma\times Q\times\{-1,0,1\}$ is a finite set of *non-zero transitions*, and $\delta^0 \subseteq Q\times\Sigma\times Q\times\{0,1\}$ is a finite set of *zero transitions*. Transitions of the form $(q,a,q',-1)$, $(q,a,q',0)$, and $(q,a,q',1)$ are, respectively, called *pop transitions*, *internal transitions*, and *push transitions*. The *size* $|\mathcal{O}|$ of $\mathcal{O}$ is defined as $|Q| + |\delta^0| + |\delta^+|$. The OCP $\mathcal{O}$ generates the graph $\mathcal{G}(\mathcal{O}) = (Q\times\mathbb{N}, \{E_a\}_{a\in\Sigma})$, where $((q,n),(q',n+k)) \in E_a$ iff either $n = 0$ and $(q,a,q',k) \in \delta^0$, or $n > 0$ and $(q,a,q',k) \in \delta^+$.

An *asynchronous product $\mathcal{O}$ of r OCPs* is simply a tuple of r OCPs $\mathcal{O}_1, \ldots, \mathcal{O}_r$ over pairwise disjoint action alphabets $\Sigma_1, \ldots, \Sigma_r$, whose control states need not be parwise disjoint. The product $\mathcal{O}$ has action labels $\Sigma := \Sigma_1 \cup \ldots \cup \Sigma_r$. Then, the graph $\mathcal{G}(\mathcal{O})$ generated by $\mathcal{O}$ is defined to be the Σ-labeled graph $\Pi_{i=1}^{r} \mathcal{G}(\mathcal{O}_i)$. The system $\mathcal{O}$ also defines another graph $\mathcal{G}_{\mathrm{S}}(\mathcal{O})$, which is simply $\mathcal{G}(\mathcal{O})$ expanded with the "synchronizing" edge relations $\{=_{i,j}\}_{1 \leq i \neq j \leq r}$ that are defined as

$$=_{i,j} := \{(\overline{c}, \overline{c}) : c_i = c_j\},$$

where $\overline{c} = (c_1, \ldots, c_r)$. In other words, the relation $=_{i,j}$ contains all self-loops in $\mathcal{G}(\mathcal{O})$ restricted to tuples, where ith and jth component agree. The graph $\mathcal{G}_{\mathrm{S}}(\mathcal{O})$ has action labels $\Sigma \cup \{(i,j)\}_{1 \leq i \neq j \leq r}$.

Logics We assume familiarity with first-order logic FO (see [15]). If the free variables of $\phi \in \mathsf{FO}$ are amongst $x_1, \ldots, x_n$, we may write $\phi(x_1, \ldots, x_n)$ instead of ϕ. Given a graph $G = (V, \{E_a\}_{a \in \Sigma})$ and a tuple $\overline{v} = (v_1, \ldots, v_n) \in V^n$, we write $G \models \phi[\overline{v}]$ to mean that ϕ is true in G over the valuation which assigns v_i to x_i. The same notations can be easily defined when dealing with $(\mathbb{N}, +)$. The *quantifier rank* of $\phi \in \mathsf{FO}$ is the maximum quantifier nesting depth in ϕ.

The k-variable first-order logic FO^k is the restriction of FO to formulas using at most k variables. Over Σ-labeled graphs, the logic FO(R) (resp. FO^k(R)) is the extension of FO (resp. FO^k) with binary relations $R_{\Sigma'}$ (for each $\Sigma' \subseteq \Sigma$) interpreted as the transitive closure relation $\rightarrow^*_{\Sigma'}$ (see [29]). Denote R_{Σ} by R. In the sequel, we often use $\rightarrow^*_{\Sigma'}$ to denote $R_{\Sigma'}$.

Formulas in the basic modal logic ML over Σ-labeled graphs are built from the following grammar: $\phi, \psi ::= \top \mid \neg\psi \mid \phi \vee \psi \mid \langle a \rangle \phi$ $(a \in \Sigma)$. Given a graph $G = (V, \{E_a\}_{a \in \Sigma})$ and each $\phi \in \mathrm{ML}$, define a set $[\![\phi]\!]_G \subseteq V$ as follows:

(1) $[\![\top]\!]_G = V$; (2) $[\![\neg\phi]\!]_G = V - [\![\phi]\!]_G$

(3) $[\![\phi \vee \psi]\!]_G = [\![\phi]\!]_G \cup [\![\psi]\!]_G$

(4) $[\![\langle a \rangle \phi]\!]_G = \{u \in V : \exists v \in V (u \rightarrow_a v \text{ and } v \in [\![\phi]\!]_G)\}$

As usual, use $\bot$, $\phi \wedge \psi$, and $[a]\phi$ to denote $\neg\top$, $\neg(\neg\phi \vee \neg\psi)$, and $\neg\langle a \rangle \neg\phi$, respectively. The logic ML(R) is the extension of ML with reachability modalities $\langle R_{\Sigma'} \rangle$ (for each $\Sigma' \subseteq \Sigma$), where $[\![\langle R_{\Sigma'} \rangle \phi]\!]_G := \{u \in V : \exists v \in V (u \rightarrow^*_{\Sigma'} v \text{ and } v \in [\![\phi]\!]_G)\}$. For the purpose of this paper, the EF-logic is the logic ML(R). [This is a slightly more general logic than the commonly considered EF-logic, which is more convenient to work with in our cases. However, all our results will hold as well for the restricted EF-logic.] There is an easy standard translation (see [15]) from formulas in ML (resp. ML(R)) to formulas in FO^2 (resp. FO^2(R)) with one free variable. Over ΠOCPs, we shall use FO_{S}(R), $\mathsf{FO}^k_{\mathrm{S}}$(R), ML_{S}(R), and the EF_{S}-logic to denote the logics FO(R), FO^k(R), ML(R), and the EF-logic with synchronizing predicates $\{=_{i,j}\}$, interpreted over graphs of the form $\mathcal{G}_{\mathrm{S}}(\mathcal{O})$.

The model checking problems for any of the above logic L over OCPs (resp. ΠOCPs) can be defined in the obvious way, i.e., with respect to the graph $\mathcal{G}(\mathcal{O})$ generated by the input OCP (resp. ΠOCPs). The input formulas are permitted to have free variables, which are to be interpreted as configurations in $\mathcal{G}(\mathcal{O})$, where numbers are *represented in binary*. Also, if L is FO_{S}(R), $\mathsf{FO}^k_{\mathrm{S}}$(R), ML_{S}(R), or the EF_{S}-logic, the interpretation is over $\mathcal{G}_{\mathrm{S}}(\mathcal{O})$.

The *first-order modulo counting logic* $\mathsf{FO}_{\mathrm{MOD}}$ extends FO with the modulo counting quantifiers $\exists^{p,q}$, for each $q \in \mathbb{Z}_{>0}$ and $p \in [0,q)$. In this paper, we consider $\mathsf{FO}_{\mathrm{MOD}}$ only over $(\mathbb{N},<)$. The semantics of $\mathsf{FO}_{\mathrm{MOD}}$ is defined over $(\mathbb{N},<)$ as follows: $(\mathbb{N},<) \models \exists^{p,q} x \phi(x,\overline{b})$ iff the number $l := |\{a \in \mathbb{N} : (\mathbb{N},<) \models \phi(a,\overline{b})\}|$ is either infinite or finite and $l \equiv p \pmod q$. See [21] for more details.

Alternation rank Given an FO formula ϕ, push all the negations to atomic propositions level. The alternation rank $\mathrm{AL}(\phi)$ of ϕ is then defined as the maximum number of alternations of operators in $\{\forall,\wedge\}$ and operators in $\{\exists,\vee\}$ over all paths from the root to the leaves in the parse tree of ϕ.

Gödel encoding For the purpose of this paper, we define the Gödel function $\mathfrak{G} : \mathbb{Z}_{>0} \to \{0,1\}^\omega$ mapping positive integers to infinite binary words as follows: if $n = \Pi_{i>0} p_i^{j_i}$, where $j_i \in \mathbb{N}$ and p_i the ith prime, then define $\mathfrak{G}(n) = j_1' j_2' \ldots$, where $j_i' = 0$ if $j_i = 0$ and $j_i' = 1$ if $j_i > 0$.

3 The logics $\mathcal{L}$ and $\mathcal{L}'$

We define our first syntactic restriction $\mathcal{L}$ of Presburger Arithmetic, to which we will reduce the model checking of $\mathsf{FO}(\mathrm{R})$ over ΠOCPs.

Definition 1. *The syntax of the logic $\mathcal{L}$ is as follows. Atomic propositions are of the form:*

- $x \sim y + c$, *where* $\sim \in \{\leq, \geq, =\}$,
- $x \sim c$, *where* $\sim \in \{\leq, \geq, =\}$,
- $x \equiv y + c \pmod d$, *where* $c \in [0, d-1]$, *and*
- $x \equiv c \pmod d$, *where* $c \in [0, d-1]$.

Here, x and y can take any variables, while c and d are constant natural numbers, given in binary representations. We then close the logic under boolean combinations, and existential and universal quantifications. The semantics is given directly from Presburger Arithmetic. The expression $x \equiv y + c \pmod d$ is to be interpreted as the Presburger formula $\exists z(\ x = y + c + dz \ \vee \ x + dz = y + c\ )$.

Intuitively, the logic $\mathcal{L}$ is the fragment of Presburger Arithmetic that permits only inequality tests, addition with constants, and modulo tests. We now impose some further syntactic restrictions to our logic $\mathcal{L}$, to which model checking $\mathsf{FO}^2(\mathrm{R})$ over ΠOCPs is still poly-time reducible.

Definition 2. *Define the logic $\mathcal{L}'$ as follows. The only variables allowed are x_i and y_i, where $i \in \mathbb{Z}_{>0}$. The atomic propositions of $\mathcal{L}'$ are given as follows for each $i \in \mathbb{Z}_{>0}$:*

- $x_i \sim y_i + c$ *and* $y_i \sim x_i + c$,
- $x_i \sim c$ *and* $y_i \sim c$,
- $x_i \equiv y_i + c \pmod d$ *and* $y_i \equiv x_i + c \pmod d$, *and*
- $x_i \equiv c \pmod d$ *and* $y_i \equiv c \pmod d$.

Here, c and d are constant natural numbers given in binary. We then close the logic under boolean combinations, and existential and universal quantifications.

The logic $\mathcal{L}'$ allows only two variables x_i and y_i to be related. In fact, if we only allow x_1 and y_1 as variables, then $\mathcal{L}'$ coincides with FO^2 fragment of $\mathcal{L}$.

We shall briefly discuss the expressive power of $\mathcal{L}$ in terms of subsets of $\mathbb{N}^k$ that can be defined in the logics. It can be shown that $\mathcal{L}$ coincides with the $\mathsf{FO}_{\mathrm{MOD}}$ theory over $(\mathbb{N}, <)$. In fact, [21] shows that $\mathsf{FO}_{\mathrm{MOD}}$ theory over $(\mathbb{N}, <)$ admits a quantifier elimination, when the vocabulary is expanded with congruence tests. Therefore, $\mathcal{L}$ subsumes $\mathsf{FO}_{\mathrm{MOD}}$ over $(\mathbb{N}, <)$. To show that $\mathcal{L} \subseteq \mathsf{FO}_{\mathrm{MOD}}(\mathbb{N}, <)$, observe that expressions of the form $x \sim y + c$ can easily be replaced by equivalent FO formulas over $(\mathbb{N}, <)$. Also, the atomic formula $x \equiv y + c \pmod d$ can be defined as $\bigwedge_{a=0}^{d-1}(y \equiv a \pmod d \leftrightarrow x \equiv a + c \pmod d)$, and congruence tests $x \equiv a \pmod d$ can be defined in $\mathsf{FO}_{\mathrm{MOD}}$ over $(\mathbb{N}, <)$ as $\exists^{a,d} y(y < x)$. The expressive power of $\mathsf{FO}_{\mathrm{MOD}}$ over $(\mathbb{N}, <)$ was shown in [21] to be strictly in between FO over $(\mathbb{N}, <)$ and Presburger Arithmetic. For example, it was shown that Presburger formulas of the form $x = 2y$ is not definable in $\mathsf{FO}_{\mathrm{MOD}}$ over $(\mathbb{N}, <)$. Finally, we shall emphasize that the proof in [21] of quantifier elimination for $\mathsf{FO}_{\mathrm{MOD}}$ over $(\mathbb{N}, <)$ expanded with congruence tests is nonconstructive.

The *membership problem of the logic* $\mathcal{L}$ is as follows: given $\phi(\overline{x}) \in \mathcal{L}$, where $\overline{x} = (x_1, \ldots, x_n)$ and a tuple $\overline{a} \in \mathbb{N}^n$ in binary, decide whether $\mathbb{N} \models \phi(\overline{a})$. The membership problem for $\mathcal{L}'$ can be defined similarly. We now state a proposition, which can be proved easily (but somewhat tedious) using the result in [8, Lemma 4.6].

Proposition 3. *There is a poly-time reduction from the problem of model checking* $\mathsf{FO}_{\mathrm{S}}(\mathrm{R})$ *(resp.* $\mathsf{FO}^2(\mathrm{R})$*) over* ΠOCPs *to the membership problem for* $\mathcal{L}$ *(resp.* $\mathcal{L}'$*). Furthermore, the alternation rank of the output formula in* $\mathcal{L}$ *(resp.* $\mathcal{L}'$*) is the same as the alternation rank of the input formula in* $\mathsf{FO}_{\mathrm{S}}(\mathrm{R})$ *(resp.* $\mathsf{FO}^2(\mathrm{R})$*) up to addition by a small constant.*

4 Upper bounds

In this section, we shall show that the combined and data complexity of $\mathsf{FO}_{\mathrm{S}}(\mathrm{R})$ over ΠOCPs are, respectively, in PSPACE and PH. We then show that the expression complexity of $\mathsf{FO}^2(\mathrm{R})$ is in P. To deduce a P^{NP} upper bound for data complexity of EF-logic over ΠOCPs, it suffices to invoke the Feferman-Vaught type of composition method for EF-logic [22] and use the P^{NP} algorithm for model checking EF-logic over OCPs from [8]. Observe that these will give the claimed upper bounds in Table 1 and Table 2.

4.1 Combined and data complexity of $\mathsf{FO}_{\mathrm{S}}(\mathrm{R})$

Theorem 4. *The combined and data complexity* $\mathsf{FO}_{\mathrm{S}}(\mathrm{R})$ *over* ΠOCPs *are in* PSPACE *and in* PH*, respectively.*

By Proposition 3, to deduce this theorem it suffices to prove the following proposition.

Proposition 5. *The membership problem of $\mathcal{L}$-formulas is in* PSPACE. *Moreover, fixing the alternation rank of input formulas, the problem is in* PH.

The proof is done via a quantifier elimination technique (e.g. see [12] for an overview). Loosely speaking, our proof can be thought of as an extension of Ehrenfeucht-Fraïssé games on linear orders (e.g. see [15]) with modulo tests. We first define an equivalence relation $\equiv^k_{p,m}$ on tuples of natural numbers.

Definition 6. *Given two $(k+1)$-tuples $\overline{a} = (a_0, \ldots, a_k), \overline{b} = (b_0, \ldots, b_k)$ of natural numbers such that $a_0 = b_0 = 0$ and two numbers $p, m > 0$, we write $\overline{a} \equiv^k_{p,m} \overline{b}$ iff for all $i, j \in [0, k]$ the following statements hold:*

1. *$|a_i - a_j| < pm$ implies $|a_i - a_j| = |b_i - b_j|$,*
2. *$|b_i - b_j| < pm$ implies $|a_i - a_j| = |b_i - b_j|$,*
3. *$|a_i - a_j| \geq pm$ iff $|b_i - b_j| \geq pm$,*
4. *$a_i \equiv b_i \pmod{p}$,*
5. *$a_i \leq a_j$ iff $b_i \leq b_j$.*

It is easy to see that, given $m' \geq m > 0$, we have $\overline{a} \equiv^k_{p,m'} \overline{b}$ implies $\overline{a} \equiv^k_{p,m} \overline{b}$. Similarly, if $p | p'$, then $\overline{a} \equiv^k_{p',m} \overline{b}$ implies $\overline{a} \equiv^k_{p,m} \overline{b}$. The following lemma can be used to eliminate a quantifier.

Lemma 7. *Given two $(k+1)$-tuples $\overline{a} = (a_0, \ldots, a_k), \overline{b} = (b_0, \ldots, b_k)$ of natural numbers such that $a_0 = b_0 = 0$ and two numbers $p, m > 0$, if $\overline{a} \equiv^k_{p,3m} \overline{b}$, then for all $a' \in \mathbb{N}$, there exists $b' \in \mathbb{N}$ such that $\overline{a}, a' \equiv^{k+1}_{p,m} \overline{b}, b'$.*

Let us consider only tuples $\overline{a} = (a_0, \ldots, a_k)$ of natural numbers satisfying $a_0 = 0$. Given an $\equiv^k_{p,3m}$-equivalence class C and an $\equiv^{k+1}_{p,m}$-equivalence class C', we say that C' is *consistent with* C if there exist a tuple $\overline{a} = (a_0, \ldots, a_k)$ of natural numbers and a number $a' \in \mathbb{N}$ such that $a_0 = 0$, $\overline{a} \in C$, and $(\overline{a}, a') \in C'$. The following lemma shows that we need not consider large numbers when eliminating a quantifier.

Lemma 8. *Let $\overline{a} = (a_0, \ldots, a_k)$ be a tuple of natural numbers and C be its $\equiv^k_{p,3m}$-equivalence class. Then, every $\equiv^{k+1}_{p,m}$-equivalence class has a representative in the set $\{(\overline{a}, a') : 0 \leq a' \leq \max(\overline{a}) + pm + p\}$.*

Define $r(0, m) := m$ and $r(n+1, m) := 3r(n, m)$, for $n \in \mathbb{N}$. By induction, we have $r(n, m) = 3^n m$. Let us now define the notion of *offsets* and *periods* of formulas in $\mathcal{L}$. If ϕ are atomic formulas of the form $x \sim y + c$, $x \sim c$, $x \equiv y + c \pmod{d}$, or $x \equiv c \pmod{d}$, then *offsets of* ϕ are defined to be the integer c. If ϕ is not an atomic formula, then its *offset* is the largest offset of atomic subformulas of ϕ. If ϕ are atomic formulas of the form $x \sim y + c$ or $x \sim c$, then its *period* is defined to be 1. If ϕ are atomic formulas of the form $x \equiv y + c \pmod{d}$ or $x \equiv c \pmod{d}$, then its *period* is defined to be d. Otherwise, if ϕ is not an atomic formula, its *period* is defined to be the least common multiple of the periods of each of its atomic subformulas. For $p, m \in \mathbb{Z}_{>0}$, define $\mathcal{L}_{p,m}$ to be formulas in $\mathcal{L}$, whose periods divide p and whose offsets are smaller than m.

Lemma 9. *Let $p, m \in \mathbb{Z}_{>0}$. Suppose $\overline{a} = (a_0, \ldots, a_k), \overline{b} = (b_0, \ldots, b_k)$ are tuples of natural numbers satisfying $a_0 = b_0 = 0$ and $\overline{a} \equiv^k_{p,r(n,m)} \overline{b}$. Then, given a formula $\phi(x_1, \ldots, x_k)$ in $\mathcal{L}_{p,m}$ of quantifier rank n,*

$$(\mathbb{N}, +) \models \phi(a_1, \ldots, a_k) \Leftrightarrow (\mathbb{N}, +) \models \phi(b_1, \ldots, b_k).$$

We are now ready to prove Proposition 5.

Proof (of Proposition 5). We now give a poly-time ATM M which checks whether $(\mathbb{N}, +) \models \phi(a_1, \ldots, a_n)$ for given a formula $\phi(x_1, \ldots, x_n)$ and a $n+1$-tuple $\overline{a} = (a_0, \ldots, a_n)$, where $a_0 = 0$. First, push all the negations downward to the atomic propositions level, which can be done easily. Suppose that p and m be, respectively, the period and offset of the input formula. Now if ϕ is an atomic proposition (i.e. inequality, or modulo tests), it is easy to see that M can check it in poly-time. If ϕ is $\psi \vee \psi'$ (resp. $\psi \wedge \psi'$), then existentially (resp. universally) guess ψ or ψ' and check the guessed formula. If ϕ is of the form $\exists x \psi(\overline{y}, x)$ (resp. $\forall x \psi(\overline{y}, x)$) and has quantifier rank k, then M existentially (resp. universally) guesses a number a_{n+1} not exceeding $\max(\overline{a}) + pr(k, m) + p \leq \max(\overline{a}) + p3^k m + p$ and check whether $(\mathbb{N}, +) \models \psi(\overline{a}, a_{n+1})$. The upper bound for a_{n+1} is sufficient due to Lemma 8.

To analyze the running time of M, notice that the maximum number that M can guess on any of its run on input ϕ of quantifier rank h and a tuple $\overline{a}$ of natural numbers (in binary) is $\max(\overline{a}) + \Sigma_{j=0}^{h}(pr(j, m) + p) \leq \max(\overline{a}) + p(h+1)3^h m + p(h+1)$, which can be represented using polynomially many bits. [Note that p and m are represented in binary and so the guessed number is polynomial in $\log(p)$ and $\log(m)$.] This implies that membership of $\mathcal{L}$-formulas is in PSPACE. Finally, notice that the number of alternations used by M corresponds to the alternation rank of ϕ. Therefore, considering only formulas of fixed alternation rank, the membership problem for $\mathcal{L}$-formulas is in PH. □

4.2 Expression complexity of $\mathsf{FO}^2(\mathrm{R})$

Theorem 10. *The expression complexity of $\mathsf{FO}^2(\mathrm{R})$ over ΠOCPs is in* P.

Define $\mathcal{L}'_{p,m}$ to be the set of all formulas in $\mathcal{L}'$ whose periods divide p and whose offsets do not exceed m. Let $\mathcal{L}'_{p,m}(n)$ to be the set of all formulas in $\mathcal{L}'_{p,m}$ that use only variables in $\{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_n\}$. For all fixed $p, m, n \in \mathbb{Z}_{>0}$, the *membership problem of* $\mathcal{L}'_{p,m}(n)$ is as follows: given $\phi(\overline{x}, \overline{y}) \in \mathcal{L}'_{p,m}(n)$ and two tuples $\overline{a}, \overline{b} \in \mathbb{N}^n$ of numbers in binary representation, decide whether $(\mathbb{N}, +) \models \phi(\overline{a}, \overline{b})$. By Proposition 3, Theorem 10 follows from the following proposition.

Proposition 11. *For fixed $p, m, n \in \mathbb{Z}_{>0}$, the membership problem of $\mathcal{L}'_{p,m}(n)$ is in* P.

This proposition can also be proved via quantifier elimination. The intuition that we can obtain a poly-time algorithm is from a two-pebble Ehrenfeucht-Fraïssé

games over linear orders (see [15]), which can only distinguish small linear orders (i.e. only linear in the quantifier rank of the $\mathsf{FO}^2(\mathrm{R})$ formula). The proof is similar to the case for $\mathsf{FO}(\mathrm{R})$, but is much more tedious.

5 Lower bounds

In order to facilitate our lower bound proofs in this section, we shall define a 2-player game, called *the buffer game*, which we shall prove to be PSPACE-complete. First, let $\mathcal{L}_{\mathrm{DIV}}$ be the set of quantifier-free $\mathcal{L}$-formulas in 3-CNF (i.e. in CNF and each clause has exactly three literals) with one free variable x, whose atomic propositions are of the form $x \equiv 0 \pmod{p}$ where p is a prime number. The buffer game is played by Player $\exists$ and Player $\forall$. An arena of the buffer game is a tuple $(\overline{v}, k, \phi)$, where $\overline{v}$ is a finite and strictly increasing sequence of positive integers, k is the number of integers in $\overline{v}$, and ϕ a formula of $\mathcal{L}_{\mathrm{DIV}}$. The buffer game with arena $(\overline{v}, k, \phi)$, where $\overline{v} = (v_1, \ldots, v_k)$, has $k+1$ rounds and is played as follows. Each round r defines a *positive* number m_r, which represents the current buffer value. At round 0, Player $\exists$ chooses a number m_0 to be written to the buffer. Suppose that $0 < r \leq k$, and $m_0, \ldots, m_{r-1}$ are the buffer values chosen from the previous rounds. At even (resp. odd) round r, Player $\exists$ (resp. Player $\forall$) rewrites the buffer by a number $m_r \geq m_{r-1}$ of his choosing such that $m_r \equiv m_{r-1} \pmod{\Pi_{j=1}^{v_r} p_j}$, i.e., $m_r = m_{r-1} + c\left(\Pi_{j=1}^{v_r} p_j\right)$ for some $c \in \mathbb{N}$. In particular, by Chinese remainder theorem, this condition implies that, for each $1 \leq j \leq v_r$, $p_j | m_r$ iff $p_j | m_{r-1}$. In other words, each player is not allowed to "overwrite" some divisibility information in the buffer. Player $\exists$ *wins* if $(\mathbb{N}, +) \models \phi(m_k)$. Otherwise, Player $\forall$ *wins*. The problem `BUFFER` is defined as follows: given an arena $(\overline{v}, k, \phi)$ of the buffer game, where *each number is represented in unary*, decide whether Player $\exists$ has a winning strategy. For each $n \in \mathbb{N}$, we define the problem $\texttt{BUFFER}_n$ to be the restriction of the problem `BUFFER` which takes only an input arena of the form $(\overline{v}, n, \phi)$.

Lemma 12. *The problem* `BUFFER` *is* PSPACE*-complete. The problem* $\texttt{BUFFER}_k$ *is* Σ^p_{k+1}*-complete.*

Loosely speaking, by applying Gödel encoding one can encode each truth valuation for boolean formulas into a number. Therefore, boolean formulas can be reduced to statements about divisibility. Furthermore, a *block* of $\exists$ (resp. $\forall$) quantifers in a quantified boolean formula can be reduced into a choice of number at a single round in the buffer game for Player $\exists$ (resp. Player $\forall$).

We now use the buffer game to prove our first lower bound result for the problem of model checking OCPs.

Proposition 13. *Combined complexity of* $\mathsf{FO}^2(\mathrm{R})$ *on OCPs is* PSPACE*-hard. For every* $k \in \mathbb{N}$*, there is a fixed formula* ϕ_k *of* $\mathsf{FO}^2(\mathrm{R})$ *with* $k + c$ *quantifier alternations, for some small constant* $c \in \mathbb{N}$*, such that checking* ϕ_k *over OCPs is* Σ^p_k*-hard.*

To prove this theorem, we first state a standard lemma, whose proof can be found in [8, 11] (similar proof techniques have been used earlier in [14]).

Lemma 14. *Given a $\mathcal{L}_{\mathrm{DIV}}$-formula ϕ, we can compute in polynomial time an OCP $\mathcal{O}$ with a fixed set Γ of action symbols and an initial state q_I such that, for each positive integer m, it is the case that $\mathcal{G}(\mathcal{O}), (q_I, m) \models \alpha$ iff $(\mathbb{N}, +) \models \phi(m)$, where α is a small fixed EF formula.*

The crucial idea in the proof of the above lemma is that both divisibility and indivisibility tests of the form $p|x$ or $p\not|x$ can be reduced to a certain reachability question for an appropriate OCP $\mathcal{O}$ by embedding a cycle of length p in $\mathcal{O}$.

Proof (sketch of Proposition 13). We give a poly-time reduction from BUFFER. Given an arena $\mathcal{A} = (\overline{v}, k, \phi)$, we compute an $\mathsf{FO}^2(\mathrm{R})$ sentence ϕ', and a OCP $\mathcal{O} = (Q, \delta^+, \delta^0)$ such that Player $\exists$ has a winning strategy in $\mathcal{A}$ iff $\mathcal{G}(\mathcal{O}) \models \phi'$. Let $\overline{v} = (v_1, \ldots, v_k)$. As we shall see, ϕ' depends only on k and has quantifier rank $k + c$ for some small constant $c \in \mathbb{N}$, which by Lemma 12 will prove the desired lower bound for data complexity.

We now run the algorithm given by Lemma 14 on input ϕ to compute a OCP $\mathcal{O}_1 = (D, \delta_1^+, \delta_1^0)$ with initial state $q_I \in D$. The key now is to build on top of $\mathcal{O}_1$ and the fixed formula α (which can be thought of as an $\mathsf{FO}^2(\mathrm{R})$ formula) so as to encode the initial guessing of numbers.

The structure of our output OCP $\mathcal{O}$ can be visualized as

$$B_0 \to B_1 \ldots \to B_k \to \mathcal{O}_1.$$

The number $k+1$ of blocks B_i in $\mathcal{O}$ corresponds to the number of rounds played in the buffer game. The initial state is in block B_0. Our output $\mathsf{FO}^2(\mathrm{R})$ formula will have $k + 1$ leading (alternating) quantifiers so as to ensure that each player moves in their designated rounds. One variable will be used for storing the last buffer value from the previous round, while the other is used for storing the buffer value after the designated player has made his move. We now describe how to ensure that at each round i ($i > 0$) the player can only add numbers that are in the set $H_i := \{c\left(\Pi_{j=1}^{v_i} p_j\right) : c \in \mathbb{N}\}$. Define the function $g : \mathbb{Z}_{>0} \to \mathbb{Z}_{>0}$ as $g(s) := \Pi_{j=1}^{s} p_j$. Note that g grows exponentially in s, which is why we cannot simply embed a cycle of length $g(v_i)$ in B_i, for each $i \in [1, k]$. On the other hand, notice that H_i is $\mathbb{Z} - L_i$, where $L_i := \left(\bigcup_{1 \le j \le v_i} \bigcup_{a \in (0, p_j)} a + \mathbb{N}p_j\right) \cup \mathbb{Z}_{<0}$.

In turn, L_i can be characterized as the set of weights of paths in a small finite graph G_i from a vertex s to a vertex t, where the *weight* of a path is the sum of the weights of its edges (which we shall allow to be only either -1,0, or 1). In fact, G_i will have $O(\Sigma_{j=1}^{v_i} p_j)$ vertices, which is polynomial in v_i. For example, the set $(1 + 2\mathbb{N}) \cup (1 + 3\mathbb{N}) \cup (2 + 3\mathbb{N}) \cup \mathbb{Z}_{<0}$ corresponds to the weights of $s \to^* t$ paths in the graph in Figure 1.

Furthermore, the graph G_i can be thought of as an OCP. Adding the self-loop transitions $(s, \mathtt{loop}_s, s, 0)$ and $(t, \mathtt{loop}_t, t, 0)$ on states s and t, the binary relation $\{((s, a), (t, a + b)) : b \in H_i\}$ can then be expressed in $\mathsf{FO}^2(\mathrm{R})$ as $\neg(x \to^* y) \wedge E_{\mathtt{loop}_s}(x, x) \wedge E_{\mathtt{loop}_t}(y, y)$. Therefore, we shall embed the modified OCP G_i into B_i, where t will be the entry state for block B_{i+1} of $\mathcal{O}$. [B_{k+1} shall be interpreted as $\mathcal{O}_1$.]

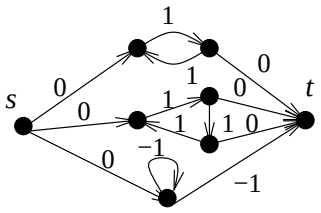

Fig. 1. The $s \to^* t$ path-weights in this graph equals $(1+2\mathbb{N}) \cup (1+3\mathbb{N}) \cup (2+3\mathbb{N}) \cup \mathbb{Z}_{<0}$.

Finally, using this idea, it is not difficult to compute the desired $\mathsf{FO}^2(\mathrm{R})$ sentence by mimicking the $k+1$ rounds of the game by using at most $k+c$ alternating quantifiers (using only the variables x and y). The end buffer value m, which needs to be checked against ϕ, can be checked against α instead. □

We can also apply Lemma 12 to prove the following lower bound.

Proposition 15. *The combined complexity of model checking **EF**-logic over an asynchronous product of two one-counter processes is* PSPACE*-hard.*

Intuitively, instead of simulating each alternation in the buffer game as values in the two variables x and y, we can simulate them as values in two different counters. We can make sure that the divisibility information is not "overwritten" by encoding it as a non-fixed formula.

We saw in the previous section that the expression complexity of $\mathsf{FO}^2(\mathrm{R})$ over ΠOCPs is in P. In contrast, we can show that this is not the case for $\mathsf{FO}^4(\mathrm{R})$ even over OCPs (without products).

Proposition 16. *The expression complexity of* $\mathsf{FO}^4(\mathrm{R})$ *(without equality relation) over OCPs is* PSPACE*-hard.*

The fixed graph is in fact $(\mathbb{N}, <)$. The proof adapts the technique in [9] of succinctly encoding addition arithmetic on large numbers using the successor relations and linear order $<$ with only four variables.

We already saw that the data complexity of EF-logic over ΠOCPs is P^{NP}. In contrast, we can show the following proposition.

Proposition 17. *For each $k \in \mathbb{N}$, there is a fixed* EF_{S}*-logic formula ϕ_k such that model checking ϕ_k over* ΠOCPs *is Σ^p_k-hard.*

Intuitively, by using the synchronization constraints, one can faithfully simulate two variables x and y in any given $\mathsf{FO}^2(\mathrm{R})$ formula as values of two different counters. This idea can easily be adapted for showing the following proposition by appealing to Proposition 16.

Proposition 18. *The expression complexity of* EF_{S}*-logic over* ΠOCPs *is hard for* PSPACE.

6 Modal logic over automatic graphs

Automatic graphs [2] are those graphs $G = (V, \{E_a\}_{a\in\Sigma})$, where V is a regular subset of Σ^* for some finite alphabet Σ, and each edge relation $E_a \subseteq \Sigma^* \times \Sigma^*$ is recognizable by a synchronous transducer over Σ. We briefly recall the definition of synchronous transducers — see [2] for more details. A *synchronous transducer* R over Σ is a finite word-automaton over the product alphabet $\Sigma_\perp \times \Sigma_\perp$, where $\Sigma_\perp := \Sigma \cup \{\perp\}$ and $\perp \notin \Sigma$. Given $v, w \in \Sigma^*$ where $v = a_1 \dots a_n$ and $w = b_1 \dots b_m$, define $v \otimes w$ to be the word $c_1 \dots c_k$ over $\Sigma_\perp \times \Sigma_\perp$, where $k = \max(n, m)$ and

$$c_i = \begin{cases} (a_i, b_i) & \text{if } i \leq \min(n, m), \\ (\perp, b_i) & \text{if } n < i \leq m, \\ (a_i, \perp) & \text{if } m < i \leq n. \end{cases}$$

The edge relation definable by R consists of each ordered pair $(v, w) \in \Sigma^* \times \Sigma^*$ such that the word $v \otimes w$ is accepted by R (in the usual automata sense). Rational graphs [19] are similar to automatic graphs, but use a more general notion of transducers.

The FO (resp. ML) theories of automatic (resp. rational) graphs are known to be decidable, e.g., see [2, 1]. In fact, the infinite binary tree S2S with a linear order is automatic [2], which implies that model checking FO over automatic graphs is nonelementary. Recently, the authors of [26] and [1] asked whether the complexity of model checking ML over, respectively, automatic graphs and rational graphs is nonelementary. We shall show that this is the case in the stronger sense by establishing a *fixed* automatic graph whose ML theory is nonelementary. Since automatic graphs are rational [19], the same can be said about rational graphs. Our proof uses the proof method for Proposition 17. Due to space limit, we shall only define a graph $\mathfrak{T}$ whose modal logic theory we claim to be nonelementary. Its proof can be found in the full version. Furthermore, one can easily check that the graph $\mathfrak{T}$ is automatic.

We denote by $\mathrm{S2S}_< := (\{0,1\}^*, \mathrm{succ}_0, \mathrm{succ}_1, \prec)$ the infinite binary tree with a descendant relation, i.e., $\mathrm{succ}_0 := \{(w, w0) : w \in \{0,1\}^*\}$, $\mathrm{succ}_1 := \{(w, w1) : w \in \{0,1\}^*\}$, and $\prec := \{(w, wv) : w, v \in \{0,1\}^*\}$. Although the FO theory of $\mathrm{S2S}_<$ was proved to be nonelementary in [4], it is not easy to see whether FO^k suffices from the proof. Nevertheless, one can easily show that FO^4 suffices as follows. Using Stockmeyer's well-known results [24] that equivalence of star-free regular expressions is nonelementary, one can immediately deduce that FO^3 theory over all finite linear orders with a unary predicate is nonelementary, since there is a linear-time translation from star-free regular expressions to equivalent FO^3 formulas (e.g. see [6]). One may then use the linear-time reduction given in [4] from FO theory of binary strings to FO theory of $\mathrm{S2S}_<$, which incurs only an extra variable. So, we have the following proposition.

Proposition 19. *The* FO^4 *theory of* $\mathrm{S2S}_<$ *is nonelementary.*

Now define the graph

$$\mathfrak{T} := \langle \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*; \\ \{\mathrm{succ}_0^i\}_{i=1}^4, \{\mathrm{succ}_1^i\}_{i=1}^4, \{\prec_i\}_{i=1}^4, \{=_{i,j}\}_{1\leq i<j\leq 4}, \{\mathsf{G}_i\}_{i=1}^4 \rangle.$$

where the edge relations are defined as follows:

- $\mathrm{succ}_0^i := \{(\overline{w}, \overline{w'}) : w_i' = w_i 0 \text{ and } \forall j \neq i(w_j = w_j')\}$. This relation takes the ith component to its left child.
- $\mathrm{succ}_1^i := \{(\overline{w}, \overline{w'}) : w_i' = w_i 1 \text{ and } \forall j \neq i(w_j = w_j')\}$. This relation takes the ith component to its right child.
- $\prec_i := \{(\overline{w}, \overline{w'}) : w_i \prec w_i' \text{ and } \forall j \neq i(w_j = w_j')\}$. This relation takes the ith component to its descendant.
- $=_{i,j} := \{(\overline{w}, \overline{w'}) : w_i = w_j \text{ and } \forall k(w_k = w_k')\}$. This relation simply loops if the ith component equals the jth component.
- $\mathsf{G}_i := \{(\overline{w}, \overline{w'}) : \forall j \neq i(w_j = w_j')\}$. This relation takes the ith component to any other word (i.e. global modality).

Proposition 20. *The graph $\mathfrak{T}$ is automatic and its* ML *theory is nonelementary.*

Theorem 21. *There exists a fixed automatic (and so rational) graph whose* ML *theory is nonelementary.*

7 Future work

We conclude now with several future work. We would like to determine the expression complexity of $\mathsf{FO}^3(\mathrm{R})$ over OCPs and ΠOCPs. Our lower bound proof for $\mathsf{FO}^4(\mathrm{R})$ does not hold since $\mathsf{FO}^3(\mathrm{R})$ cannot succinctly encode arithmetic of large numbers using only successors and linear orders [9]. We would also like to study the combined complexity of EF-logic over asynchronous products of PDA. It is known to be PSPACE-hard [27] and decidable (by applying the compositional method [22]), but no better upper or lower bound is known. [In fact, it seems not clear how to adapt the proof of PSPACE upper bound for PDA to asynchronous products.] On the other hand, it follows from the proof of Theorem 21 that model checking EF_{S}-logic over asynchronous products of PDA is nonelementary.

Acknowledgments: The author thanks Shunichi Amano, Stefan Göller, Leonid Libkin, and anonymous reviewers for their helpful comments. The author is supported by EPSRC grant E005039 and ORSAS Award.

References

1. W. Bekker and V. Goranko. Symbolic model checking of tense logics on rational Kripke models. *To appear in proceedings of ILC'07.*
2. A. Blumensath and E. Grädel. Automatic structures. In *LICS '00*, pages 51–62.
3. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR '97*, pages 135–150.
4. K. J. Compton and C. W. Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *Ann. Pure Appl. Logic*, 48(1):1–79, 1990.

5. J. Esparza, A. Kucera, and S. Schwoon. Model checking LTL with regular valuations for pushdown systems. *Inf. Comput.*, 186(2):355–376, 2003.
6. K. Etessami, M. Y. Vardi, and T. Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2):279–295, 2002.
7. J. Ferrante and C. W. Rackoff. *The Computational Complexity of Logical Theories*, volume 718. Springer-Verlag, 1979.
8. S. Göller, R. Mayr, and A. W. To. On the computational complexity of verifying one-counter processes. *To appear in LICS*, 2009.
9. M. Grohe and N. Schweikardt. The succinctness of first-order logic on linear orders. *Logical Methods in Computer Science*, 1(1), 2005.
10. P. Jancar and Z. Sawa. A note on emptiness for alternating finite automata with a one-letter alphabet. *Inf. Process. Lett.*, 104(5):164–167, 2007.
11. 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.
12. D. C. Kozen. *Theory of Computation*. Springer-Verlag, 2006.
13. M. W. Krentel. The complexity of optimization problems. *J. Comput. Syst. Sci.*, 36(3):490–509, 1988.
14. A. Kučera. Efficient verification algorithms for one-counter processes. In *ICALP '00*, pages 317–328.
15. L. Libkin. *Elements Of Finite Model Theory (Texts in Theoretical Computer Science. An EATCS Series)*. SpringerVerlag, 2004.
16. C. Löding. Reachability problems on regular ground tree rewriting graphs. *Theory Comput. Syst.*, 39(2):347–383, 2006.
17. J. A. Makowsky. Algorithmic uses of the Feferman-Vaught Theorem. *Ann. Pure Appl. Logic*, 126(1-3):159–213, 2004.
18. R. Mayr. Decidability of model checking with the temporal logic EF. *Theor. Comput. Sci.*, 256(1-2):31–62, 2001.
19. C. Morvan. On rational graphs. In *FOSSACS '00*, pages 252–266, 2000.
20. D. E. Muller and P. E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theor. Comput. Sci.*, 37:51–75, 1985.
21. P. Péladeau. Logically defined subsets of $\mathbb{N}^k$. *Theor. Comput. Sci.*, 93(2):169–183, 1992.
22. A. Rabinovich. On compositionality and its limitations. *ACM Trans. Comput. Logic*, 8(1):4, 2007.
23. O. Serre. Parity games played on transition graphs of one-counter processes. In L. Aceto and A. Ingólfsdóttir, editors, *FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2006.
24. L. J. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD thesis, Department of Electrical Engineering, MIT, 1974.
25. W. Thomas. Constructing infinite graphs with a decidable MSO-theory. In *MFCS '03*, pages 113–124.
26. A. W. To and L. Libkin. Recurrent reachability analysis in regular model checking. In *LPAR '08*, pages 198–213.
27. I. Walukiewicz. Model checking CTL properties of pushdown systems. In *FSTTCS '00*, pages 127–138.
28. I. Walukiewicz. Pushdown processes: games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001.
29. S. Wöhrle and W. Thomas. Model checking synchronized products of infinite transition systems. *Logical Methods in Computer Science*, 3(4), 2007.