Tightening the Complexity of Equivalence Problems for Commutative Grammars*

Christoph Haase[1] and Piotr Hofman[2]

1 **Laboratoire Spécification et Vérification (LSV), CNRS & ENS Cachan Université Paris-Saclay, France**
haase@lsv.ens-cachan.fr

2 **Laboratoire Spécification et Vérification (LSV), CNRS & ENS Cachan Université Paris-Saclay, France**
hofman@lsv.ens-cachan.fr

Abstract

Given two finite-state automata, are the Parikh images of the languages they generate equivalent? This problem was shown decidable in coNEXP by Huynh in 1985 within the more general setting of context-free commutative grammars. Huynh conjectured that a Π_2^P upper bound might be possible, and Kopczyński and To established in 2010 such an upper bound when the size of the alphabet is fixed. The contribution of this paper is to show that the language equivalence problem for regular and context-free commutative grammars is actually coNEXP-complete. In addition, our lower bound immediately yields further coNEXP-completeness results for equivalence problems for regular commutative expressions, reversal-bounded counter automata and communication-free Petri nets. Finally, we improve both lower and upper bounds for language equivalence for exponent-sensitive commutative grammars.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases language equivalence, commutative grammars, presburger arithmetic, semi-linear sets, petri nets

Digital Object Identifier 10.4230/LIPIcs.STACS.2016.41

1 Introduction

Language equivalence is one of the most fundamental decision problems in formal language theory. Classical results include PSPACE-completeness of deciding language equivalence for regular languages generated by non-deterministic finite-state automata (NFA) [4, p. 265], and the undecidability of language equivalence for languages generated by context-free grammars [12, p. 318].

Equivalence problems for formal languages which are undecidable over the free monoid may become decidable in the commutative setting. The problem then is to decide whether the Parikh images of two languages coincide. Given a word w over an alphabet Σ consisting of m alphabet symbols, the Parikh image of w is a vector in $\mathbb{N}^m$ counting in its i-th component how often the i-th alphabet symbol occurs in w. This definition can then be lifted to languages, and the Parikh image of a language consequently becomes a subset of $\mathbb{N}^m$, or, equivalently, a subset of $\Sigma^{\odot}$, the free commutative monoid generated by Σ. Parikh's theorem states that Parikh images of context-free languages are semi-linear sets. Since the latter are closed

* Supported by Labex Digicosme, Univ. Paris-Saclay, project VERICONISS.

under all Boolean operations [5], deciding equivalence between Parikh images of context-free languages is decidable.

When dealing with Parikh images of formal languages, it is technically more convenient to directly work with commutative grammars, which were introduced by Huynh in his seminal paper [13] and are "generating devices for commutative languages [that] use [the] free commutative monoid instead of [the] free monoid." In [13], Huynh studied the uniform word problem for various classes of commutative grammars; the complexity of equivalence problems for commutative grammars was subsequently investigated in a follow-up paper [14]. One of the main results in [14] is that the equivalence problem for regular and context-free commutative grammars is Π_2^{P}-hard and in coNEXP. Huynh remarks that a better upper bound might be possible and states as an open problem the question whether the equivalence problem for context-free commutative grammars is Π_2^{P}-complete [14, p. 117]. Some progress towards answering this question was made by Kopczyński and To, who showed that inclusion and *a fortiori* equivalence for regular and context-free commutative grammars are coNP-complete respectively Π_2^{P}-complete when the size of the alphabet is fixed [18, 17]. One of the main contributions of this paper is to answer Huynh's question negatively: we show that already for regular commutative grammars the equivalence problem is coNEXP-complete.

Our coNEXP lower bound is established by showing how to reduce validity in the coNEXP-complete Π_2-fragment of Presburger arithmetic [7, 8] (i.e. its $\forall^*\exists^*$-fragment) to language inclusion for regular commutative grammars. A reduction from this fragment of Presburger arithmetic has recently been used in [9] in order to show coNEXP-completeness of inclusion for integer vector addition systems with states ($\mathbb{Z}$-VASS), and this reduction is our starting point. Similarly to the standard definition of vector addition systems with states, $\mathbb{Z}$-VASS comprise a finite-state controller with a finite number of counters which, however, range over the integers. Consequently, counters can be incremented and decremented, may drop below zero, and the order in which transitions in $\mathbb{Z}$-VASS are taken may commute along a run—those properties are crucial to the hardness proof in [9]. The corresponding situation is different and technically challenging for regular commutative grammars. In particular, alphabet symbols can only be produced but not deleted, and, informally speaking, we cannot produce negative quantities of alphabet symbols.

A further contribution of our paper is to establish a new upper bound for the equivalence problem for exponent-sensitive commutative grammars, a generalisation of context-free commutative grammars where the left-hand sides of productions may contain an arbitrary number of some non-terminal symbol. Exponent-sensitive commutative grammars were recently introduced by Mayr and Weihmann in [21], who showed PSPACE-completeness of the word problem and membership in 2-EXPSPACE of the equivalence problem. Our hardness result implies that the equivalence problem is coNEXP-hard, and we also improve the 2-EXPSPACE-upper bound to co-2NEXP.

Finally, commutative grammars are closely related to Petri nets, cf. [13, 3, 27, 23]. We also discuss implications of our results to equivalence problems for various classes of Petri nets as well as regular commutative expressions [2] and reversal-bounded counter automata [16].

Due to space constraints, we only sketch some of the proofs in this paper. Full proofs can, however, be found in a technical report accompanying this paper which can be obtained from the authors' homepages.[1]

[1] http://www.mimuw.edu.pl/~ph209519/Papers/Stacs2016.pdf

2 Preliminaries

2.1 Commutative Grammars

Let $\Sigma = \{a_1, \ldots, a_m\}$ be a finite alphabet. The free monoid generated by Σ is denoted by Σ^*, and we denote by $\Sigma^\odot$ the free commutative monoid generated by Σ. We interchangeably use different equivalent ways in order to represent a word $w \in \Sigma^\odot$. For $1 \leq j \leq m$, let i_j be the number of times a_j occurs in w, we equivalently write w as $w = a_1^{i_1} a_2^{i_2} \cdots a_m^{i_m}$, $w = (i_1, i_2, \ldots, i_m) \in \mathbb{N}^m$ or $w : \Sigma \to \mathbb{N}$ with $w(a_j) = i_j$, whatever is most convenient. By $|w| = \sum_{1 \leq j \leq m} i_j$ we denote the length of w, and the representation size $\#w$ of w is $\sum_{1 \leq j \leq m} \lceil \log i_j \rceil$. Given $v, w \in \Sigma^\odot$, we sometimes write $v + w$ in order to denote the concatenation $v \cdot w$ of v and w. The empty word is denoted by ϵ, and as usual $\Sigma^+ \stackrel{\text{def}}{=} \Sigma^* \setminus \{\epsilon\}$ is the free semi-group and $\Sigma^\oplus \stackrel{\text{def}}{=} \Sigma^\odot \setminus \{\epsilon\}$ the free commutative semi-group generated by Σ. For $\Gamma \subseteq \Sigma$, $\pi_\Gamma(w)$ denotes the projection of w onto alphabet symbols from Γ.

A commutative grammar (sometimes just grammar in the following) is a tuple $G = (N, \Sigma, S, P)$, where

- N is the finite set of non-terminal symbols;
- Σ is a finite alphabet, the set of terminal symbols, such that $N \cap \Sigma = \emptyset$;
- $S \in N$ is the axiom; and
- $P \subseteq N^\oplus \times (N \cup \Sigma)^\odot$ is a finite set of productions.

The size of G, denoted by $\#G$, is defined as

$$\#G \stackrel{\text{def}}{=} |N| + |\Sigma| + \sum_{(V,W) \in P} |V| + |W|.$$

Note that commutative words in G are encoded in unary. Unless stated otherwise, we use this definition of the size of a commutative grammar in this paper.

Subsequently, we write $V \to W$ whenever $(V, W) \in P$. Let $D, E \in (N \cup \Sigma)^\odot$, we say D directly generates E, written $D \Rightarrow_G E$, iff there are $F \in (N \cup \Sigma)^\odot$ and $V \to W \in P$ such that $D = V + F$ and $E = F + W$. We write $\Rightarrow_G^*$ to denote the reflexive transitive closure of $\Rightarrow_G$, and if $U \Rightarrow_G^* V$ we say that U generates V. If G is clear from the context, we omit the subscript G. For $U \in N^\oplus$, the reachability set $\mathcal{R}(G, U)$ and the language $\mathcal{L}(G, U)$ generated by G starting at U are defined as

$$\mathcal{R}(G, U) \stackrel{\text{def}}{=} \{W \in (N \cup \Sigma)^\odot : U \Rightarrow^* W\} \qquad \mathcal{L}(G, U) \stackrel{\text{def}}{=} \mathcal{R}(G, U) \cap \Sigma^\odot.$$

The reachability set $\mathcal{R}(G)$ and the language $\mathcal{L}(G)$ of G are then defined as $\mathcal{R}(G) \stackrel{\text{def}}{=} \mathcal{R}(G, S)$ and $\mathcal{L}(G) \stackrel{\text{def}}{=} \mathcal{L}(G, S)$. The word problem is, given a commutative grammar G and $w \in \Sigma^\odot$, is $w \in \mathcal{L}(G)$? The main focus of our paper is on the complexity of deciding language inclusion and equivalence for commutative grammars: Given commutative grammars G, H, language inclusion is to decide $\mathcal{L}(G) \subseteq \mathcal{L}(H)$, and language equivalence is to decide $\mathcal{L}(G) = \mathcal{L}(H)$. Since our grammars admit non-determinism, language inclusion and equivalence are logarithmic-space inter-reducible.

By imposing restrictions on the set of productions, we obtain various classes of commutative grammars. Following [13, 21], given $G = (N, \Sigma, S, P)$, we say that G is

- of *type*-0 if there are no restrictions on P;
- *context-sensitive* if $|W| \geq |V|$ for each $V \to W \in P$;
- *exponent-sensitive* if $V \in \{\{U\}^\oplus : U \in N\}$ for each $V \to W \in P$;
- *context-free* if $V \in N$ for each $V \to W \in P$;
- *regular* if $V \in N$ and $W \in (N \cup \{\epsilon\}) \cdot \Sigma^\odot$ for each $V \to W \in P$.

Equivalence problems for commutative grammars were studied by Huynh, who showed that it is undecidable for context-sensitive and hence type-0 grammars, and Π_2^{P}-hard and in coNEXP for regular and context-free commutative grammars [14]. The main contribution of this paper is to prove the following theorem.

▶ **Theorem 1.** *The language equivalence problem for regular and context-free commutative grammars problem is* coNEXP*-complete.*

Exponent-sensitive grammars were only recently introduced by Mayr and Weihmann [21]. They showed that the word problem is PSPACE-hard, and that language equivalence is PSPACE-hard and in 2-EXPSPACE. The lower bounds require commutative words on the left-hand sides of productions to be encoded in binary. The second main contribution of our paper is to improve those results as follows.

▶ **Theorem 2.** *The language equivalence problem for exponent-sensitive commutative grammars is* coNEXP*-hard and in* co-2NEXP*.*

2.2 Presburger Arithmetic, Linear Diophantine Inequalities and Semi-Linear Sets

Let $\boldsymbol{u} = (u_1, \ldots, u_m)$, $\boldsymbol{v} = (v_1, \ldots, v_m) \in \mathbb{Z}^m$, the sum of $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined component-wise, i.e., $\boldsymbol{u} + \boldsymbol{v} = (u_1 + v_1, \ldots, u_m + v_m)$. Given $u \in \mathbb{Z}$, $\hat{\boldsymbol{u}}$ denotes the vector consisting of u in every component and any appropriate dimension. Let $1 \leq i \leq j \leq m$, we define $\pi_{[i,j]}(\boldsymbol{u}) \stackrel{\text{def}}{=} (u_i, \ldots, u_j)$. By $\|\boldsymbol{u}\|_\infty$ we denote the maximum norm of $\boldsymbol{u}$, i.e., $\|\boldsymbol{u}\|_\infty \stackrel{\text{def}}{=} \max\{|u_i| : 1 \leq i \leq n\}$. Let $M, N \subseteq \mathbb{Z}^m$ and $k \in \mathbb{Z}$, as usual $M + N$ is defined as $\{\boldsymbol{m} + \boldsymbol{n} : \boldsymbol{m} \in M,\ \boldsymbol{n} \in N\}$ and $k \cdot M \stackrel{\text{def}}{=} \{k \cdot \boldsymbol{m} : \boldsymbol{m} \in M\}$. Moreover, $\|M\|_\infty \stackrel{\text{def}}{=} \max\{\|\boldsymbol{z}\|_\infty : \boldsymbol{z} \in M\}$. The size $\#\boldsymbol{u}$ of $\boldsymbol{u}$ is $\#\boldsymbol{u} \stackrel{\text{def}}{=} \sum_{1 \leq i \leq m} \lceil \log |u_i| \rceil$, i.e., numbers are encoded in binary, and the size of M is $\#M \stackrel{\text{def}}{=} \sum_{\boldsymbol{u} \in M} \#\boldsymbol{u}$. For an $m \times n$ matrix A consisting of elements $a_{ij} \in \mathbb{Z}$, $\|A\|_{1,\infty} \stackrel{\text{def}}{=} \max\{\sum_{1 \leq j \leq n} |a_{ij}| : 1 \leq i \leq m\}$.

Presburger arithmetic is the is the first-order theory of the structure $\langle \mathbb{N}, 0, 1, +, \geq \rangle$. In this paper, atomic formulas of Presburger arithmetic are linear Diophantine inequalities of the form

$$\sum_{1 \leq i \leq n} a_i \cdot x_i \geq z_i,$$

where $a_i, z_i \in \mathbb{Z}$ and the x_i are first-order variables. Formulas of Presburger arithmetic can then be obtained in the usual way via positive Boolean combinations of atomic formulas and existential and universal quantification over first-order variables, i.e., according to the following grammar:

$$\phi ::= \forall \boldsymbol{x}.\phi \mid \exists \boldsymbol{x}.\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid t$$

Here, the $\boldsymbol{x}$ range over tuples of first-order variables, and t ranges over linear Diophantine inequalities as above. We assume that formulas of Presburger arithmetic are represented as a syntax tree, with no sharing of sub-formulas.

Given a formula ϕ of Presburger arithmetic with no free variables, validity is to decide whether ϕ holds with respect to the standard interpretation in arithmetic. By $\|\phi\|_\infty$ we denote the largest constant occurring in ϕ, and $|\phi|$ is the length of ϕ, i.e., the number of symbols required to write down ϕ, where constants are represented in unary. In analogy to matrices,

we define $\|\phi\|_{1,\infty} \stackrel{\text{def}}{=} \|\phi\|_\infty \cdot |\phi|$. Let $\psi(\boldsymbol{x})$ be a quantifier-free formula open in $\boldsymbol{x} = (x_1, \dots, x_m)$ and $\boldsymbol{x}^* = (x_1^*, \dots, x_m^*) \in \mathbb{N}^m$, we denote by $\psi[\boldsymbol{x}^*/\boldsymbol{x}]$ the formula obtained from ψ by replacing every x_i in ψ by x_i^*. Finally, given a quantifier-free Presburger formula ψ containing linear Diophantine inequalities $t_1, \dots, t_k$ and $b_1, \dots, b_k \in \{0, 1\}$, $\psi[b_1/t_1, \dots, b_k/t_k]$ denotes the Boolean formula obtained from ψ by replacing every t_i with b_i.

In this paper, we are in particular interested in the Π_2-fragment of Presburger arithmetic, i.e. the fragment in which formulas are restricted to a form $\phi = \forall \boldsymbol{x}.\exists \boldsymbol{y}.\psi(\boldsymbol{x}, \boldsymbol{y})$ where $\psi(\boldsymbol{x}, \boldsymbol{y})$ is quantifier free, for which the following is known.

▶ **Theorem 3** ([7, 8]). *Validity in the Π_2-fragment of Presburger arithmetic is* coNEXP*-complete (with hardness under polynomial-time many-one reductions).*

The sets of natural numbers definable in Presburger arithmetic are semi-linear sets [6]. Let $\boldsymbol{b} \in \mathbb{N}^m$ and $P = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_n\}$ be a finite subset of $\mathbb{N}^m$, define

$$\operatorname{cone}(P) \stackrel{\text{def}}{=} \{\lambda_1 \cdot \boldsymbol{p}_1 + \cdots + \lambda_n \cdot \boldsymbol{p}_n : \lambda_i \in \mathbb{N},\ 1 \leq i \leq n\}.$$

A linear set $L(\boldsymbol{b}, P)$ with base $\boldsymbol{b}$ and periods P is defined as $L(\boldsymbol{b}, P) \stackrel{\text{def}}{=} \boldsymbol{b} + \operatorname{cone}(P)$. A semi-linear set is a finite union of linear sets. For convenience, given a finite subset B of $\mathbb{N}^m$, we define $L(B, P) \stackrel{\text{def}}{=} \bigcup_{\boldsymbol{b} \in B} L(\boldsymbol{b}, P)$. The size of a semi-linear set $M = \bigcup_{i \in I} L(B_i, P_i) \subseteq \mathbb{N}^m$ is defined as

$$\#M \stackrel{\text{def}}{=} \sum_{i \in I} \#B_i + |B_i| \cdot \#P_i.$$

In particular, numbers are encoded in binary. Given a semi-linear set $N \subseteq \mathbb{N}^m$, $\#N$ is the minimum over the sizes of all semi-linear sets $M = \bigcup_{i \in I} L(\boldsymbol{b}_i, P_i)$ such that $N = M$.

A system of linear Diophantine inequalities D is a conjunction of linear inequalities over the same first-order variables $\boldsymbol{x} = (x_1, \dots, x_n)$, which we write in the standard way as $D : A \cdot \boldsymbol{x} \geq \boldsymbol{c}$, where A is a $m \times n$ integer matrix and $\boldsymbol{c} \in \mathbb{N}^m$. The size $\#D$ of D is the number of symbols required to write down D, where we assume binary encoding of numbers. The set of solutions of D is denoted by $[\![D]\!] \subseteq \mathbb{N}^n$. We say that D is feasible if $[\![D]\!] \neq \emptyset$. In [24, 1], bounds on the semi-linear representation of $[\![D]\!]$ are established. The following theorem is a consequence of Corollary 1 in [24] and Theorem 5 in [1].

▶ **Theorem 4.** *Let $D : A \cdot \boldsymbol{x} \geq \boldsymbol{c}$ be a system of linear Diophantine inequalities such that A is an $m \times n$ matrix. Then $[\![D]\!] = L(B, P)$ for $B, P \subseteq \mathbb{N}^n$ such that $|P| \leq \binom{m+n}{m}$ and*

$$\|B\|_\infty, \|P\|_\infty \leq (\|A\|_{1,\infty} + \|\boldsymbol{c}\|_\infty + 2)^{m+n}.$$

3 Lower Bounds

In this section, we establish the coNEXP-lower bound of Theorems 1 and 2. For the sake of a clear presentation, we will first describe the reduction for context-free commutative grammars, and then outline how the approach can be adapted to regular commutative grammars.

As stated in the introduction, we reduce from validity in the Π_2-fragment of Presburger arithmetic. To this end, let $\phi = \forall \boldsymbol{x}.\exists \boldsymbol{y}.\psi(\boldsymbol{x}, \boldsymbol{y})$ such that $\boldsymbol{x} = (x_1, \dots, x_m)$, $\boldsymbol{y} = (y_1, \dots, y_n)$, and ψ is a positive Boolean combination of atomic formulas $t_1, \dots, t_k$. For our reduction, we write atomic formulas of ψ as

$$t_i : \sum_{1 \leq j \leq m} (a_{i,j}^+ - a_{i,j}^-) \cdot x_j + z_i^+ - z_i^- \geq \sum_{1 \leq j \leq n} (b_{i,j}^+ - b_{i,j}^-) \cdot y_j, \tag{1}$$

where the $a^+_{i,j}, a^-_{i,j} \in \mathbb{N}$ are such that $a^+_{i,j} = 0$ or $a^-_{i,j} = 0$, and likewise the $b^+_{i,j}, b^-_{i,j} \in \mathbb{N}$ are such that $b^+_{i,j} = 0$ or $b^-_{i,j} = 0$, and the $z^+_i, z^-_i \in \mathbb{N}$ such that $z^+_i = 0$ or $z^-_i = 0$. Moreover, in the following we set $a_{i,j} \stackrel{\text{def}}{=} a^+_{i,j} - a^-_{i,j}$, $b_{i,j} \stackrel{\text{def}}{=} b^+_{i,j} - b^-_{i,j}$ and $z_i \stackrel{\text{def}}{=} z^+_i - z^-_i$.

▶ **Example 5.** Let $\phi = \forall x.\exists y.\psi(x,y)$ with $\psi(x,y) = (t_1 \wedge t_2) \vee (t_3 \wedge t_4)$ and

$$t_1 = x \geq 2 \cdot y \qquad t_2 = -x \geq -2 \cdot y \qquad t_3 = x + 1 \geq 2 \cdot y \qquad t_4 = -x - 1 \geq -2 \cdot y,$$

which expresses that every natural number is either even or odd. Here, for instance, $a^+_{2,1} = 0$, $a^-_{2,1} = 1$, $z^+_1 = z^-_1 = 0$, $b^+_{2,1} = 0$ and $b^-_{2,1} = 2$. Hence $a_{2,1} = -1$, $z_2 = 0$ and $b_{2,1} = -2$.

With no loss of generality and due to unary encoding of numbers in ϕ, we may assume that the following inequalities hold:

$$|\phi| \geq 2 + m + n + k \qquad\qquad |\phi| \geq \|\phi\|_\infty \tag{2}$$

We furthermore define a constant $c \in \mathbb{N}$, whose bit representation is polynomial in $|\phi|$, as

$$c \stackrel{\text{def}}{=} \min\{2^n : n \in \mathbb{N},\ 2^n \geq |\phi|^{3\cdot|\phi|+2} \cdot 2^{|\phi|}\}. \tag{3}$$

Let $\Sigma \stackrel{\text{def}}{=} \{t^+_1, t^-_1, \ldots, t^+_k, t^-_k\}$, we now show how to construct in logarithmic space context-free commutative grammars G, H over Σ such that $\mathcal{L}(G) \subseteq \mathcal{L}(H)$ iff ϕ is valid. The underlying idea is as follows: the language of G consists of all possible values of the left-hand sides of the inequalities t_i for every choice of $\boldsymbol{x}$, where the value of some t_i is represented by a word $w \in \Sigma^\odot$ via the difference $w(t^+_i) - w(t^-_i)$. For every $w \in \Sigma^\odot$ and $1 \leq i \leq k$, we misuse notation and define $w(t_i) \stackrel{\text{def}}{=} w(t^+_i) - w(t^-_i) \in \mathbb{Z}$; note that in particular $t_i \notin \Sigma$. The grammar H can then be defined in an analogous way and produces the values of the right-hand sides of H for a choice of $\boldsymbol{y}$, but can in addition simulate the Boolean structure of ψ in order to tweak those t_i for which, informally speaking, it cannot obtain a good value.

Before we define G, we remark that in context-free commutative grammars we can assume commutative words to be encoded in binary. This is not possible in regular grammars.

▶ Remark. For any class of commutative grammars containing context-free commutative grammars, it is with no loss of generality possible to assume binary encoding of commutative words, which has, for instance, been observed in [26]. For example, given a production $V \to a^{2^n}$, $n > 0$, we can introduce fresh non-terminal symbols $V_1, \ldots, V_n$ and replace $V \to a^{2^n}$ by $V \to V_1V_1$, $V_n \to a$ and $V_i \to V_{i+1}V_{i+1}$ for every $1 \leq i < n$. Clearly, the grammar obtained by this procedure generates the same language and only results in a sub-quadratic blow-up of the size of the resulting grammar.

Recall that we may represent commutative words of $\Sigma^\odot$ as vectors of natural numbers. We define:

$$u \stackrel{\text{def}}{=} (z^+_1, z^-_1, \ldots, z^+_k, z^-_k) \in \Sigma^\odot \qquad v_i \stackrel{\text{def}}{=} (a^+_{1,i}, a^-_{1,i}, \ldots, a^+_{k,i}, a^-_{k,i}) \in \Sigma^\odot \quad (1 \leq i \leq m) \tag{4}$$

where $a^+_{j,i}, a^-_{j,i}, z^+_j, z^-_j$ are defined in Equation (1).

The grammar G is constructed as $G \stackrel{\text{def}}{=} (N_G, \Sigma, S_G, P_G)$, where $N_G \stackrel{\text{def}}{=} \{S, X\}$ and P_G is defined as follows:

$$S_G \to X\hat{\boldsymbol{c}}u \qquad\qquad X \to \epsilon \qquad\qquad X \to X\hat{\boldsymbol{c}}v_i \qquad\qquad (1 \leq i \leq m)$$

Here, c is the constant from (3) whose addition ensures that the values of the t^+_i and t^-_i generated by G are large. Clearly, G can be constructed in logarithmic space even though c is exponential in $|\phi|$. The following lemma captures the essential properties of G.

▶ **Lemma 6.** *Let G be as above. The following hold:*

(i) *For every $\boldsymbol{x} \in \mathbb{N}^m$ there exists $w \in \mathcal{L}(G)$ such that for all $1 \leq i \leq k$,*

$$w(t_i) = \sum_{1 \leq j \leq m} (a^+_{i,j} - a^-_{i,j}) \cdot x_j + z^+_i - z^-_i.$$

(ii) *For every $w \in \mathcal{L}(G)$ there exists $\boldsymbol{x} \in \mathbb{N}^m$ such that for all $1 \leq i \leq k$,*

$$w(t_i) = \sum_{1 \leq j \leq m} (a^+_{i,j} - a^-_{i,j}) \cdot x_j + z^+_i - z^-_i \tag{5}$$

$$w(t^+_i) \geq c + z^+_i + \sum_{1 \leq j \leq m} c \cdot x_j \geq c \cdot (1 + \|\boldsymbol{x}\|_\infty) \tag{6}$$

$$w(t^-_i) \geq c + z^-_i + \sum_{1 \leq j \leq m} c \cdot x_j \geq c \cdot (1 + \|\boldsymbol{x}\|_\infty). \tag{7}$$

We now turn towards the construction of $H \stackrel{\text{def}}{=} (N_H, \Sigma, S_H, P_H)$ and define the set of non-terminals N_H and productions P_H of H in a step-wise fashion. Starting in S_H, H branches into three gadgets starting at the non-terminal symbols Y, F_ψ and I:

$$S_H \to Y F_\psi I$$

Here, Y is an analogue to X in G. Informally speaking, it allows for obtaining the right-hand sides of the inequalities t_i for a choice of $\boldsymbol{y} \in \mathbb{N}^n$. In analogy to G, we define

$$w_i \stackrel{\text{def}}{=} (b^+_{1,i}, b^-_{1,i}, \ldots, b^+_{k,i}, b^-_{k,i}) \in \Sigma^\odot \qquad (1 \leq i \leq n)$$

$$Y \to Y w_i \qquad (1 \leq i \leq n)$$

$$Y \to \epsilon$$

In contrast to X from G, note that Y does not add $\hat{\boldsymbol{c}}$ every time it loops. The following lemma is the analogue of H to Lemma 6 and can be shown along the same lines.

▶ **Lemma 7.** *Let Y be the non-terminal of H as defined above. The following hold:*

(i) *For every $\boldsymbol{y} \in \mathbb{N}^n$ there exists $w \in \mathcal{L}(H, Y)$ such that for all $1 \leq i \leq k$, $w(t^+_i) = \sum_{1 \leq j \leq n} b^+_{i,j} \cdot y_j$, $w(t^-_i) = \sum_{1 \leq j \leq n} b^-_{i,j} \cdot y_j$, and*

$$w(t_i) = \sum_{1 \leq j \leq n} (b^+_{i,j} - b^-_{i,j}) \cdot y_j.$$

(ii) *For every $w \in \mathcal{L}(H, Y)$ there exists $\boldsymbol{y} \in \mathbb{N}^n$ such that for all $1 \leq i \leq k$,*

$$w(t_i) = \sum_{1 \leq j \leq n} (b^+_{i,j} - b^-_{i,j}) \cdot y_j.$$

It is clear that the w_Y generated by Y may not be able to generate all t_i in a way that match all w generated by G (i.e., all choices of $\boldsymbol{x}$ made through G). For now, let us even assume that $w(t^+_i) \geq w_Y(t^+_i)$ and $w(t^-_i) \geq w_Y(t^-_i)$ holds for every $1 \leq i \leq k$. Later, we will show that if there is a good choice for $\boldsymbol{y}$, we can find a good $w_Y \in \mathcal{L}(H, Y)$ with this property. After generating w_Y, informally speaking, H should produce t^+_i and t^-_i in order match w, provided that ψ is valid.

In particular, the Boolean structure of ψ enables us to produce arbitrary quantities of some t_i. This is the duty of the gadget F_ψ which allows for assigning arbitrary values to some

atomic formulas t_i via gadgets R_{t_i} defined below. The gadget F_ψ recursively traverses the matrix formula ψ and invokes some R_γ whenever a disjunction is processed and a disjunct γ is evaluated to false:

$$F_{t_i} \to \epsilon \qquad F_{\alpha\wedge\beta} \to F_\alpha F_\beta \qquad F_{\alpha\vee\beta} \to F_\alpha R_\beta \qquad F_{\alpha\vee\beta} \to R_\alpha F_\beta \qquad F_{\alpha\vee\beta} \to F_\alpha F_\beta$$

The definition of R_γ for every subformula γ of ψ occurring in the syntax tree of ψ is now not difficult: we traverse γ until we reach a leaf t_i of the syntax tree of γ and then allow for generating an arbitrary number of alphabet symbols t_i^+ and t_i^-. Let $1 \leq i \leq k$, we define the following productions:

$$R_{t_i} \to \epsilon \qquad R_{t_i} \to R_{t_i} t_i^+ \qquad R_{t_i} \to R_{t_i} t_i^- \qquad R_{\alpha\wedge\beta} \to R_\alpha R_\beta \qquad R_{\alpha\vee\beta} \to R_\alpha R_\beta$$

Finally, it remains to provide a possibility to increase $w_Y(t_i)$ for those t_i that were not processed by some R_{t_i} in order to match w. For a good choice of w_Y, we certainly should have that for those t_i, the number of t_i generated by w_Y in H is at least as much as the number generated by G. Hence, in order to make w_Y agree with w on t_i, all we have to do to w_Y is to non-deterministically increment, i.e., produce, t_i^+ at least as often as t_i^-. This is the task of the gadget I of H, whose production rules are as follows:

$$I \to \epsilon \qquad I \to I t_i^+ t_i^- \qquad I \to I t_i^+ \qquad (1 \leq i \leq k)$$

The subsequent lemma, whose proof is immediate, states the properties of I formally.

▶ **Lemma 8.** $\mathcal{L}(H, I) = \left\{(n_1^+, n_1^-, \ldots, n_k^+, n_k^-) \in \Sigma^\odot : n_j^+ \geq n_j^-,\ 1 \leq j \leq k\right\}$.

This completes the construction of H. We now prove the correctness of our construction.

▶ **Lemma 9.** *Suppose $\mathcal{L}(G) \subseteq \mathcal{L}(H)$, then $\phi = \forall \boldsymbol{x}.\exists \boldsymbol{y}.\psi(\boldsymbol{x}, \boldsymbol{y})$ is valid.*

Proof. The idea underlying the proof is to show how to construct for every choice of $\boldsymbol{x} \in \mathbb{N}^m$ some $\boldsymbol{y} \in \mathbb{N}^n$ such that $\psi(\boldsymbol{x}, \boldsymbol{y})$ evaluates to true. By Lemma 6(i), for any $\boldsymbol{x} \in \mathbb{N}^m$ there exists some corresponding $w \in \mathcal{L}(G)$ and by assumption $w \in \mathcal{L}(H)$. In particular, w is composed of some $w_Y \in \mathcal{L}(H, Y)$ from which by Lemma 7(ii) some suitable $\boldsymbol{y} \in \mathbb{N}^n$ can be obtained. ◀

The converse direction is slightly more involved. Informally speaking, on the first sight one might be worried that H produces more t_i^+ or t_i^- than G which cannot be "erased." However, the addition of c in every component for every production applied by G together with Theorem 4 allows us to overcome this obstacle.

▶ **Lemma 10.** *Suppose $\phi = \forall \boldsymbol{x}.\exists \boldsymbol{y}.\psi(\boldsymbol{x}, \boldsymbol{y})$ is valid, then $\mathcal{L}(G) \subseteq \mathcal{L}(H)$.*

Proof. Let $w \in \mathcal{L}(G)$, by Lemma 6(ii) there exists $\boldsymbol{x}^* \in \mathbb{N}^m$ such that (5), (6) and (7) hold. By assumption, there is $\boldsymbol{y}^* \in \mathbb{N}^n$ such that $\psi(\boldsymbol{x}^*, \boldsymbol{y}^*)$ holds. Hence, there is $\xi : \{1, \ldots, k\} \to \{0, 1\}$ such that for all i where $\xi(i) = 1$,

$$\sum_{1 \leq j \leq m} a_{i,j} \cdot x_j^* + z_i \geq \sum_{1 \leq j \leq n} b_{i,j} \cdot y_j^*$$

and $\psi[\xi(1)/t_1, \ldots, \xi(k)/t_k]$ evaluates to true. With no loss of generality, write $\{i : \xi(i) = 1\} = \{1, \ldots, h\}$ for some $1 \leq h \leq k$. Consider the system $D : A \cdot (\boldsymbol{x}, \boldsymbol{y}) \geq \boldsymbol{z}$ of linear Diophantine inequalities over the unknowns $\boldsymbol{x}$ and $\boldsymbol{y}$, where

$$A \stackrel{\text{def}}{=} \begin{pmatrix} a_{1,1} & \cdots & a_{1,m} & -b_{1,1} & \cdots & -b_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{h,1} & \cdots & a_{h,m} & -b_{h,1} & \cdots & -b_{h,n} \end{pmatrix} \qquad \boldsymbol{z} \stackrel{\text{def}}{=} \begin{pmatrix} -z_1 \\ \vdots \\ -z_h \end{pmatrix}.$$

By assumption, D has a non-empty solution set. We have that A is a $h \times (m+n)$ matrix with $\|M\|_{1,\infty} \leq \|\psi\|_{1,\infty}$ and $\|\boldsymbol{z}\|_\infty \leq \|\psi\|_\infty$. By Theorem 4, there are $B, P \subseteq \mathbb{N}^{m+n}$ such that $[\![D]\!] = B + \mathrm{cone}(P)$. Consequently, $\boldsymbol{x}^* = \pi_{[1,m]}(\boldsymbol{b} + \lambda_1 \cdot \boldsymbol{p}_1 + \cdots + \lambda_\ell \cdot \boldsymbol{p}_\ell)$ for some $\boldsymbol{b} \in B$, $\boldsymbol{p}_i \in P$ and $\lambda_i \in \mathbb{N}$. In particular, since $|P| \leq \binom{h+m+n}{h} \leq 2^{|\phi|}$ we have

$$0 \leq \sum_{1 \leq i \leq \ell} \lambda_i \leq \|\boldsymbol{x}^*\|_\infty \cdot \ell \leq \|\boldsymbol{x}^*\|_\infty \cdot 2^{|\phi|}. \tag{8}$$

Now let

$$\boldsymbol{y}^\dagger \stackrel{\text{def}}{=} \pi_{[m+1,m+n]}(\boldsymbol{b} + \lambda_1 \cdot \boldsymbol{p}_1 + \cdots + \lambda_\ell \cdot \boldsymbol{p}_\ell).$$

We have $(\boldsymbol{x}^*, \boldsymbol{y}^\dagger)$ is a solution of D and henceforth $\psi[\boldsymbol{x}^*/\boldsymbol{x}, \boldsymbol{y}^\dagger/\boldsymbol{y}]$ evaluates to true. In fact, it is not difficult to show that

$$\|\boldsymbol{y}^\dagger\|_\infty \leq (1 + \|\boldsymbol{x}^*\|_\infty) \cdot \frac{c}{|\phi|^2}.$$

Combining the estimation of $\|\boldsymbol{y}^\dagger\|_\infty$ with (6) and (7) of Lemma 6, for every $1 \leq i \leq k$ we obtain

$$w(t_i^+), w(t_i^-) \geq c \cdot (1 + \|\boldsymbol{x}^*\|_\infty) \geq \|\boldsymbol{y}^\dagger\|_\infty \cdot |\phi|^2 \geq \|\boldsymbol{y}^\dagger\|_\infty \cdot \|\phi\|_\infty \cdot |\phi|. \tag{9}$$

By Lemma 7(i) there is $w_Y \in \mathcal{L}(H, Y)$ such that (9) yields

$$w(t_i^+) \geq \sum_{1 \leq j \leq n} \|\boldsymbol{y}^\dagger\|_\infty \cdot \|\phi\|_\infty \geq \sum_{1 \leq j \leq n} b_{i,j}^+ \cdot y_j^\dagger = w_Y(t_i^+)$$
$$w(t_i^-) \geq \sum_{1 \leq j \leq n} \|\boldsymbol{y}^\dagger\|_\infty \cdot \|\phi\|_\infty \geq \sum_{1 \leq j \leq n} b_{i,j}^- \cdot y_j^\dagger = w_Y(t_i^-).$$

Moreover, the construction of F_ψ is such that

$$\left\{w_F \in \Sigma^\odot : w_F(t_i^+) = w_F(t_i^-) = 0,\ \xi(i) = 1,\ 1 \leq i \leq k\right\} \subseteq \mathcal{L}(H, F_\psi).$$

Hence, we can find some $w_F \in \mathcal{L}(H, F_\psi)$ which allows us to adjust those t_i for which $\xi(i) = 0$. More formally, for $1 \leq i \leq k$ such that $\xi(i) = 0$,

$$(w_Y + w_F)(t_i^+) = w(t_i^+) \text{ and } (w_Y + w_F)(t_i^-) = w(t_i^-).$$

On the hand, for all $1 \leq i \leq k$ such that $\xi(i) = 1$,

$$(w_Y + w_F)(t_i^+) = w_Y(t_i^+) \text{ and } (w_Y + w_F)(t_i^+) = w_Y(t_i^+),$$

i.e., those t_i remain untouched by w_F.

Consequently, it remains to show that there is a suitable $w_I \in \mathcal{L}(H, I)$ such that we can adjust those t_i which were left untouched by w_F above. For all $1 \leq i \leq k$ such that $\xi(i) = 1$, since $\boldsymbol{y}^\dagger$ is a solution of D, we have

$$\begin{aligned}
& w(t_i) = w(t_i^+) - w(t_i^-) \geq w_Y(t_i^+) - w_Y(t_i^-) = w_Y(t_i) \\
\iff & w(t_i^+) - w_Y(t_i^+) \geq w(t_i^-) - w_Y(t_i^-) \\
\iff & \text{there are } m_i, n_i \in \mathbb{N} \text{ such that } w(t_i^+) = w_Y(t_i^+) + m_i + n_i \text{ and} \\
& w(t_i^-) = w_Y(t_i^-) + m_i.
\end{aligned}$$

But then Lemma 8 yields the required $w_I \in \mathcal{L}(H, I)$ such that $w_I(t_i^+) = m_i + n_i$, $w_I(t_i^-) = m_i$, and $w_I(t_j^+) = w_I(t_j^+) = 0$ for all j such that $\xi(j) = 0$.

Summing up, we have $w = w_Y + w_I + w_F$, and hence $w \in \mathcal{L}(H)$ as required. ◀

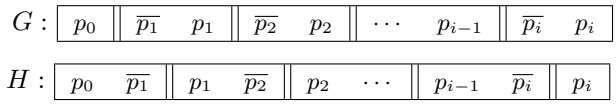

Figure 1 Illustration of the pairing of alphabet symbols. In G, we require that in each cell $w(p_{j+1}) = 2 \cdot w(\overline{p_j})$, and in H that $w(p_j) = w(\overline{p_j})$. Any word fulling these conditions has the property that $w(p_i) = 2^i \cdot w(p_0)$.

Lemmas 9 and 10 together with Theorem 3 yield the coNEXP-lower bound of Theorems 1 and 2 of the language inclusion problem for context-free and exponent-sensitive grammars, and hence coNEXP-hardness of the equivalence problem.

▶ Remark. It is not difficult to see that one can derive from G and H commutative context-free grammars G^e and H^e such that an even stronger statement holds:

$$\phi \text{ is valid} \iff \mathcal{R}(G^e) = \mathcal{R}(H^e).$$

3.1 Hardness for Regular Commutative Grammars

It remains to show how our reduction can be adapted in order to prove coNEXP-hardness of the equivalence problem for regular commutative grammars. Due to space constraints, we only sketch the main ideas, full details can be found in the technical report accompanying this paper.

As constructed above, neither G nor H are regular, the main problem being the following rules of G:

$$S_G \to X\hat{\boldsymbol{c}}u \qquad X \to X\hat{\boldsymbol{c}}v_i \qquad (1 \leq i \leq m).$$

Here, $\hat{\boldsymbol{c}}$ is a word of exponential length, cf. Equation (3), and, informally speaking, we cannot force a regular commutative grammar to generate an exponential quantity of an alphabet symbol. However, the interplay between G and H allows us to do so. The main idea is that in order to generate $\hat{\boldsymbol{c}}$ we use additional alphabet symbols $p_0, \ldots, p_i$ such that we require that number of occurrences of p_{j+1} is twice as much as p_j in a word w accepted by G for all $0 \leq j < i$, or otherwise this word is trivially accepted by H. With this approach we get that that if w witnesses $\mathcal{L}(G) \not\subseteq \mathcal{L}(H)$ then $w(p_i) = 2^i \cdot w(p_0)$, which is exactly what we need. In some more detail, the construction actually uses further additional alphabet symbols $\overline{p_1}, \ldots, \overline{p_i}$. Then, we enforce in G that $w(p_{j+1}) = 2 \cdot w(\overline{p_j})$, and in H that $w(p_j) = w(\overline{p_{j+1}})$. This can be achieved by accepting any word w in H for which $w(p_j) \neq w(\overline{p_{j+1}})$. Figure 1 illustrates this technique of pairing alphabet symbols p_j and $\overline{p_j}$.

Finally, the gadget F_ψ of H is also not regular. However, we can alternatively simulate ψ by a regular grammar in which conjunction in ψ corresponds to sequential composition and disjunction to branching.

4 Exponent-Sensitive Commutative Grammars

We now turn towards the equivalence problem for exponent-sensitive commutative grammars and sketch the proof of Theorem 2, i.e., show that language inclusion is coNEXP-hard and in co-2NEXP. The lower bound immediately follows from Theorem 1. Hence, it remains to provide a co-2NEXP upper bound, thereby improving the 2EXPSPACE upper bound from [21]. All formal details can be found in the technical report accompanying this paper.

It is sufficient to show that language inclusion between exponent-sensitive commutative grammars can be decided in co-2NEXP. To this end, we adapt an approach proposed by

Huynh used to show that language inclusion between context-free commutative grammars is in coNEXP [14]. Let G and H be exponent-sensitive commutative grammars. The starting point of Huynh's approach is to derive bounds on the size of a commutative word witnessing non-inclusion via the semi-linear representation of the reachability sets of G and H. For exponent-sensitive commutative grammars, in [22] $\mathcal{R}(G)$ and $\mathcal{R}(H)$ are shown semi-linear with a representation size doubly exponential in $\#G$ and in $\#H$, respectively, and this representation is also computable in doubly-exponential time. Given semi-linear sets M and N such that $M \setminus N$ is non-empty, Huynh shows in [15] that there is some $\boldsymbol{v} \in M \setminus N$ whose bit-size is polynomial in $\#M + \#N$. Consequently, if $\mathcal{L}(G) \not\subseteq \mathcal{L}(H)$ then the binary representation of some word $w \in \mathcal{L}(G) \setminus \mathcal{L}(H)$ has size bounded by $2^{2^{p(\#G+\#H)}}$ for some polynomial p. Since the word problem for exponent-sensitive commutative grammars is in PSPACE, deciding $\mathcal{L}(G) \subseteq \mathcal{L}(H)$ is in 2-EXPSPACE, as observed in [22, Thm. 5.5]. Now comes the second part of Huynh's approach into play. In [14], a Carathéodory-type theorem for semi-linear sets is established: given a linear set $M = L(\boldsymbol{b}, P) \subseteq \mathbb{N}^m$, Huynh shows that $M = \bigcup_{i \in I} L(\boldsymbol{b}_i, P_i)$, where $\boldsymbol{b}_i \in L(\boldsymbol{b}, P)$, each $\boldsymbol{b}_i$ has bit-size polynomial in $\#M$, and $P_i \subseteq P$ has full column rank and hence in particular $|P_i| \leq m$. The key point is that deciding membership in a linear set with such properties is obviously in P using Gaussian elimination, and that we can show that a semi-linear representation of $\mathcal{R}(G)$ and $\mathcal{R}(H)$ in which every linear set has those properties is computable in deterministic doubly-exponential time in $\#G$ and in $\#H$, respectively. Consequently, a co-2NEXP algorithm to decide $\mathcal{L}(G) \subseteq \mathcal{L}(H)$ can initially guess a word w whose representation is doubly-exponential in $\#G + \#H$, then compute the semi-linear representations of $\mathcal{R}(G)$ and $\mathcal{R}(H)$ in the special form of Huynh, and check in time polynomial in $\#w$ that w belongs to $\mathcal{L}(G)$ and not to $\mathcal{L}(H)$.

5 Applications to Further Equivalence Problems

Here, we discuss immediate corollaries of Theorem 1 for various other equivalence problems in formal language and automata theory. Due to space constraints, we cannot provide formal definitions of the objects we consider; they can be found in the references in the respective paragraphs.

In [2], Eilenberg and Schützenberger studied properties of regular languages in commutative monoids which are generated by regular commutative expressions. Such regular expressions are the same as standard regular expression which use the free commutative monoid instead of the free monoid. From Theorem 1, we obtain the following statement.

▶ **Theorem 11.** *Language equivalence between regular commutative expressions is* coNEXP*-complete.*

The upper bound can easily be obtained via a reduction to equivalence between regular commutative grammars. The lower bound follows from the observation that the construction outlined in Section 3.1 can be adjusted in a way such that the directed graph underlying the constructed regular commutative grammar does not contain nested cycles, and can hence be translated into an equivalent regular commutative expression of linear size.

Regular commutative grammars can also be viewed as 0-reversal-bounded counter automata in which every counter corresponds to an alphabet symbol. Reversal-bounded counter automata were introduced by Ibarra [16]. Along a run of a k-reversal bounded counter automaton, every counter may only change from incrementing to decrementing mode at most k times. Given two reversal-bounded counter automata with the same number of counters, equivalence is to decide whether their sets of counter values occurring in a final configuration is the same.

Table 1 Complexity of the word and the equivalence problem for classes of commutative grammars.

commutative grammar	word problem	language equivalence
type-0	EXPSPACE-h. [20], $\in \mathbf{F}_{\omega^3}$ [19]	undecidable [10]
context-sensitive	PSPACE-complete [13]	undecidable [14]
exponent-sensitive	PSPACE-complete [21]	coNEXP-h., $\in$ co-2NEXP
context-free	NP-complete [13, 3]	coNEXP-complete
regular		

▶ **Theorem 12.** *The equivalence problem for reversal-bounded counter automata is* coNEXP*-complete.*

The lower bound immediately follows from Theorem 1. For the upper bound, Hague and Lin [11] have shown that the set of counter values occurring in a final configuration is definable in existential Presburger arithmetic. Consequently, given two reversal-bounded counter automata whose reachability sets are defined by existential Presburger formulas $\varphi(\boldsymbol{x})$ and $\psi(\boldsymbol{x})$, respectively, they are equivalent iff $\phi \stackrel{\text{def}}{=} \forall \boldsymbol{x}.\varphi(\boldsymbol{x}) \leftrightarrow \psi(\boldsymbol{x})$ is valid. Since ϕ is a Π_2-sentence of Presburger arithmetic, Theorem 3 yields a coNEXP-upper bound for the equivalence problem.

Finally, it has, for instance, been observed in [3, 27, 23] that context-free commutative grammars can be seen as notational variants of communication-free Petri nets and basic parallel process nets (BPP-nets). In particular, language equivalence is logarithmic-space interreducible with reachability equivalence for such nets. Hence, Theorem 1 together with Remark 3 yields the following theorem.

▶ **Theorem 13.** *The equivalence problem for communication-free Petri nets and BPP-nets is* coNEXP*-complete.*

6 Conclusion

We showed that language inclusion and equivalence for regular and context-free commutative grammars are coNEXP-complete, resolving a long-standing open question posed by Huynh [14]. Our lower bound also carries over to the equivalence problem for exponent-sensitive commutative grammars, for which we could also improve the 2-EXPSPACE-upper bound [21] to co-2NEXP. The precise complexity of this problem remains an open problem of this paper. An overview over the complexity of word and equivalence problems for commutative grammars together with references to the literature is provided in Table 1.

One major open problem related to the problems discussed in this paper is weak bisimilarity between basic parallel processes. This problem is not known to be decidable and PSPACE-hard [25]. Unfortunately, it does not seem possible to adjust the construction of our coNEXP-lower bound to also work for weak bisimulation.

References

1 Eric Domenjoud. Solving systems of linear Diophantine equations: An algebraic approach. In *Mathematical Foundations of Computer Science (MFCS)*, pages 141–150, 1991.

2 Samuel Eilenberg and M.P. Schützenberger. Rational sets in commutative monoids. *Journal of Algebra*, 13(2):173–191, 1969.

3 Javier Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundamenta Informaticae*, 31(1):13–25, 1997. doi:10.3233/FI-1997-3112.

4 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

5 Seymour Ginsburg. *The mathematical theory of context free languages*. McGraw-Hill, 1966.

6 Seymour Ginsburg and Edwin H. Spanier. Bounded ALGOL-like languages. *Transactions of the American Mathematical Society*, pages 333–368, 1964.

7 Erich Grädel. Dominoes and the complexity of subclasses of logical theories. *Annals of Pure Applied Logic*, 43(1):1–30, 1989. doi:10.1016/0168-0072(89)90023-7.

8 Christoph Haase. Subclasses of Presburger arithmetic and the weak EXP hierarchy. In *Proceedings of the Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (CSL-LICS)*, pages 47:1–47:10. ACM, 2014. doi:10.1145/2603088.2603092.

9 Christoph Haase and Simon Halfon. Integer vector addition systems with states. In *Proceedings of the 8th International Workshop on Reachability Problems (RP)*, volume 8762 of *Lecture Notes in Computer Science*, pages 112–124. Springer, 2014. doi:10.1007/978-3-319-11439-2_9.

10 Michel Hack. The equality problem for vector addition systems is undecidable. *Theoretical Computer Science*, 2(1):77–95, 1976. doi:10.1016/0304-3975(76)90008-6.

11 Matthew Hague and Anthony Widjaja Lin. Model checking recursive programs with numeric data types. In *Proccedings of the 23rd International Conference on Computer Aided Verification (CAV)*, volume 6806 of *Lecture Notes in Computer Science*, pages 743–759. Springer, 2011. doi:10.1007/978-3-642-22110-1_60.

12 John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation – international edition (2. ed)*. Addison-Wesley, 2003.

13 Dung T. Huynh. Commutative grammars: The complexity of uniform word problems. *Information and Control*, 57(1):21–39, 1983. doi:10.1016/S0019-9958(83)80022-9.

14 Dung T. Huynh. The complexity of equivalence problems for commutative grammars. *Information and Control*, 66(1–2):103–121, 1985. doi:10.1016/S0019-9958(85)80015-2.

15 Dung T. Huynh. A simple proof for the Σ_2^p upper bound of the inequivalence problem for semilinear sets. *Elektronische Informationsverarbeitung und Kybernetik*, 22(4):147–156, 1986.

16 Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, 1978. doi:10.1145/322047.322058.

17 Eryk Kopczyński. Complexity of problems of commutative grammars. *Logical Methods in Computer Science*, 11(1), 2015. doi:10.2168/lmcs-11(1:9)2015.

18 Eryk Kopczyński and Anthony Widjaja To. Parikh images of grammars: Complexity and applications. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, (LICS)*, pages 80–89. IEEE Computer Society, 2010. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5570020, doi:10.1109/LICS.2010.21.

19 Jérôme Leroux and Sylvain Schmitz. Demystifying reachability in vector addition systems. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 56–67. IEEE, 2015. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7174833.

20 Richard Lipton. The reachability problem is exponential-space-hard. Technical report, Yale University, New Haven, CT, 1976.

21 Ernst W. Mayr and Jeremias Weihmann. Completeness results for generalized communication-free Petri nets with arbitrary edge multiplicities. In *Proceedings of the 7th International Workshop on Reachability Problems (RP)*, volume 8169 of *Lecture Notes in Computer Science*, pages 209–221. Springer, 2013. doi:10.1007/978-3-642-41036-9_19.

22 Ernst W. Mayr and Jeremias Weihmann. Completeness results for generalized communication-free Petri nets with arbitrary edge multiplicities. Technical Report TUM-I1335, Technische Universität München, 2013. URL: http://mediatum.ub.tum.de/node?id=1169599.

23 Ernst W. Mayr and Jeremias Weihmann. Complexity results for problems of communication-free Petri nets and related formalisms. *Fundamenta Informaticae*, 137(1):61–86, 2015. doi:10.3233/FI-2015-1170.

24 Loïc Pottier. Minimal solutions of linear Diophantine systems: Bounds and algorithms. In *Proceedings of the 4th International Conference on Rewriting Techniques and Applications (RTA)*, volume 488 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 1991. doi:10.1007/3-540-53904-2_94.

25 Jirí Srba. Complexity of weak bisimilarity and regularity for BPA and BPP. *Mathematical Structures in Computer Science*, 13(4):567–587, 2003. doi:10.1017/S0960129503003992.

26 Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In *5th Annual ACM Symposium on Theory of Computing, STOC*, pages 1–9. ACM, 1973.

27 Hsu-Chun Yen. On reachability equivalence for BPP-nets. *Theoretical Computer Science*, 179(1-2):301–317, 1997. doi:10.1016/S0304-3975(96)00147-8.