TIGHTENING THE COMPLEXITY OF EQUIVALENCE PROBLEMS FOR COMMUTATIVE GRAMMARS[*]

CHRISTOPH HAASE AND PIOTR HOFMAN

Abstract. We show that the language equivalence problem for regular and context-free commutative grammars is coNEXP-complete. In addition, our lower bound immediately yields further coNEXP-completeness results for equivalence problems for communication-free Petri nets and reversal-bounded counter automata. Moreover, we improve both lower and upper bounds for language equivalence for exponent-sensitive commutative grammars.

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) [GJ79, p. 265], and the undecidability of language equivalence for languages generated by context-free grammars [HMU03, 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 under all Boolean operations [Gin66], 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 [Huy83] and are "generating devices for commutative languages [that] use [the] free commutative monoid instead of [the] free monoid." In [Huy83], 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 [Huy85]. One of the main results in [Huy85] 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 [Huy85, p. 117]. Some progress towards answering this question

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

was made by Kopczyński and To, who showed that inclusion and *a fortiori* equivalence for regular and context-free commutative grammars is Π_2^{P}-complete when the size of the alphabet is fixed [KT10, Kop15]. 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 [Grä89, Haa14] (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 [HH14] 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 [HH14]. 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 [MW13a], 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 very closely related to Petri nets, c.f. [Huy83, Esp97, Yen97, MW15]. We also discuss implications of our results to equivalence problems for various classes of Petri nets as well as reversal-bounded counter automata [Iba78].

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 the 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 $\Sigma^+ \overset{\text{def}}{=} \Sigma^* \setminus \{\epsilon\}$ is the free semi-group and $\Sigma^\oplus \overset{\text{def}}{=} \Sigma^\odot \setminus \{\epsilon\}$ 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 subsequently) 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
- $P \subset N^{\oplus} \times (N \cup \Sigma)^{\odot}$ 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 $w \in \mathcal{L}(G)$? Our main focus in this paper is, however, 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 interreducible.

By imposing restrictions on the set of productions, we obtain various classes of commutative grammars. Following [Huy83, MW13a], given $G = (N, \Sigma, S, P)$, we say that G is

- of *type*-0 if there are no restrictions on P;
- *context-sensitive* if $|V| \geq |W|$ 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 [Huy85]. 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 [MW13a]. 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 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.

Remark 3. For context-free commutative grammars, it is with no loss of generality possible to assume binary encoding of commutative words, which has, for instance, been remarked in [KT10]. 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 $V \to V_1 V_1$, $V_n \to a$ 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.

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:

$$\phi ::= \forall \boldsymbol{x}.\phi \mid \exists \boldsymbol{x}.\phi \mid \phi \land \phi \mid \phi \lor \phi \mid t$$

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 $\boldsymbol{x} = (x_1, \ldots, x_m)$ and $\boldsymbol{x}^* = (x_1^*, \ldots, x_m^*) \in \mathbb{N}^m$, 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 $t_1, \ldots, t_k$ and $b_1, \ldots, b_k \in \{0,1\}$, $\psi[b_1/t_1, \ldots, b_k/t_k]$ 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, for which the following is known[1].

Proposition 4 ([Grä89, Haa14])**.** Validity in the Π_2-fragment of Presburger arithmetic is coNEXP-complete.

The sets of natural numbers definable in Presburger arithmetic are semi-linear sets [GS64]. Let $\boldsymbol{b} \in \mathbb{N}^m$ and $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ be a finite subset of $\mathbb{N}^m$, define

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

A linear set $L(\boldsymbol{b}, P)$ with base $\boldsymbol{b}$ and periods P is defined as $L(\boldsymbol{b}, P) \stackrel{\text{def}}{=} \boldsymbol{b} + \mathrm{cone}(P)$. 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 $\boldsymbol{x} = (x_1, \ldots, x_n)$, which we write in the standard way as $D : A \cdot \boldsymbol{x} \ge \boldsymbol{c}$, 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 [Pot91, Dom91], bounds on the semi-linear representation of $[\![D]\!]$ are established. The following proposition is a consequence of Corollary 1 in [Pot91] and Theorem 5 in [Dom91].

Proposition 5 ([Pot91, Dom91])**.** Let $D : A \cdot \boldsymbol{x} \ge \boldsymbol{c}$ 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 $|P| \le \binom{n}{m}$ and

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

3. Lower Bounds

In this section, we establish the coNEXP-lower bound of Theorems 1 and 2. This is shown by establishing coNEXP-hardness of language inclusion for regular commutative grammars. However, for the sake of a clear presentation, we will first describe the reduction for context-free commutative grammars, and then show how the approach can be adapted to regular commutative grammars. Finally, we also show that even reachability equivalence is coNEXP-hard.

[1] The hardness result is stated under polynomial-time many-one reductions in [Haa14] .

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 $\boldsymbol{x} = (x_1, \ldots, x_m)$, $\boldsymbol{y} = (y_1, \ldots, y_n)$, and ψ is a positive Boolean combination of atomic formulas $t_1, \ldots, t_k$. For our reduction, we write atomic formulas of ψ as

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

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 $a_{i,j} \overset{\text{def}}{=} a_{i,j}^+ - a_{i,j}^-$, $b_{i,j} \overset{\text{def}}{=} b_{i,j}^+ - b_{i,j}^-$ and $z_i \overset{\text{def}}{=} z_i^+ - z_i^-$.

Example 6. Consider the formula ϕ that we will use as our running example such that $\phi = \forall x.\exists y.\psi(x, y)$, where $\psi(x, y) = (t_1 \wedge t_2) \vee (t_3 \wedge t_4)$ and

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

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:

$$(1) \qquad |\phi| \geq 2 + m + n + k \qquad\qquad |\phi| \geq \|\phi\|_\infty$$

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

$$(2) \qquad c \overset{\text{def}}{=} \min\left\{2^n \geq |\phi|^{3 \cdot |\phi| + 2} \cdot 2^{|\phi|} : n \in \mathbb{N}\right\}.$$

Let $\Sigma \overset{\text{def}}{=} \{t_1^+, t_1^-, \ldots, t_k^+, t_k^-\}$, 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 $w(t_i) \overset{\text{def}}{=} w(t_i^+) - w(t_i^-) \in \mathbb{Z}$; 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. We explain the reduction in further detail in due course, but for now give a small example and then turn towards the formal definition of G.

Example 7. Let ϕ be our running example. We have that $\Sigma = \{t_1^+, t_1^-, \ldots, t_4^+, t_4^-\}$. Our goal is to define a context-free commutative grammar G such that,

$$\mathcal{L}(G) = \{(c, c, c, c, c+1, c, c, c+1) + i \cdot (c+1, c, c, c+1, c+1, c, c, c+1) \in \Sigma^\odot : i \in \mathbb{N}\}.$$

For any $w \in \mathcal{L}(G)$, there is then some $x \in \mathbb{N}$ such that $w(t_1) = x$, $w(t_2) = -x$, $w(t_3) = x + 1$ and $w(t_4) = -x - 1$, those values which correspond to the possible values of the left-hand sides of the atomic formulas of ϕ for any choice of x. ◊

Recall that we may represent commutative words of $\Sigma^{\odot}$ as vectors of natural numbers, we define:

$$u \overset{\text{def}}{=} (z_1^+, z_1^-, \ldots, z_k^+, z_k^-) \in \Sigma^{\odot} \tag{3}$$

$$v_i \overset{\text{def}}{=} (a_{1,i}^+, a_{1,i}^-, \ldots, a_{k,i}^+, a_{k,i}^-) \in \Sigma^{\odot} \qquad (1 \leq i \leq m) \tag{4}$$

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

$$\begin{array}{lll} S_G \to X\hat{\boldsymbol{c}}u & X \to X\hat{\boldsymbol{c}}v_i & (1 \leq i \leq m) \\ X \to \epsilon & & \end{array}$$

Here, c is the constant from (2) whose addition ensures that the values of the t_i^+ and t_i^- generated by G are large. Moreover, recall that it follows from Remark 3 that G can be constructed in logarithmic space even though c is exponential in $|\phi|$. The following lemma captures the essential properties of G.

Lemma 8. 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 $1 \leq i \leq k$,

$$w(t_i) = \sum_{1 \leq j \leq m} (a_{i,j}^+ - a_{i,j}^-) \cdot x_j + z_i^+ - z_i^- \tag{5}$$

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

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

Proof. Regarding (i), let $\boldsymbol{x} = (x_1, \ldots, x_m) \in \mathbb{N}^m$ and consider the following derivation of G:

$$S_G \Rightarrow Xu \Rightarrow Xv_1u \Rightarrow^* Xv_1^{x_m}u \Rightarrow^* Xv_1^{x_1} \cdots v_{m-1}^{x_{m-1}}u \Rightarrow^* v_1^{x_1} \cdots v_{m-1}^{x_{m-1}}v_m^{x_m}u = w.$$

For every $1 \leq i \leq k$ we have

$$\begin{aligned} w(t_i^+) &= v_1(t_i^+) \cdot x_1 + \cdots + v_m(t_i^+) \cdot x_m + u(t_i^+) \\ &= (a_{i,1}^+ + c) \cdot x_1 + \cdots + (a_{i,m}^+ + c) \cdot x_1 + z_i^+ + c \\ &= \sum_{1 \leq j \leq m} (a_{i,j}^+ + c) \cdot x_j + z_i^+ + c. \end{aligned}$$

In the same way, we obtain

$$w(t_i^-) = \sum_{1 \leq j \leq m} (a_{i,j}^- + c) \cdot x_j + z_i^- + c,$$

whence

$$w(t_i) = w(t_i^+) - w(t_i^-) = \sum_{1 \leq j \leq m} (a_{i,j}^+ - a_{i,j}^-) \cdot x_j + z_i^+ - z_i^-.$$

Regarding (ii), by the construction of G, any $w \in \mathcal{L}(G)$ is of the form

$$w = v_1^{x_1} \cdots v_{m-1}^{x_{m-1}} v_m^{x_m} u.$$

Define $\boldsymbol{x} \stackrel{\text{def}}{=} (x_1, \ldots, x_m)$ and let $1 \leq i \leq k$, Equation (5) follows as in (i). Moreover, since $u(t_i^+) = c + z_i^+$ and $v_j(t_i^+) \geq c$ for every $1 \leq j \leq m$, we obtain inequality (6). The same argument allows for deriving (7). □

This completes the construction of G and the proof of the relevant properties of G. 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

$$\begin{aligned} w_i &\stackrel{\text{def}}{=} (b_{1,i}^+, b_{1,i}^-, \ldots, b_{k,i}^+, b_{k,i}^-) \in \Sigma^{\odot} && (1 \leq i \leq n)\\ Y &\to Y w_i && (1 \leq i \leq n)\\ Y &\to \epsilon \end{aligned}$$

In contrast to X from G, note that Y does not add $\boldsymbol{c}$ every time it loops. The following lemma is the analogue of H to Lemma 8 and can be shown along the same lines.

Lemma 9. 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 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:

$$\begin{aligned} F_{t_i} &\to \epsilon & F_{\alpha \wedge \beta} &\to F_\alpha F_\beta\\ F_{\alpha \vee \beta} &\to F_\alpha R_\beta & F_{\alpha \vee \beta} &\to R_\alpha F_\beta\\ F_{\alpha \vee \beta} &\to F_\alpha F_\beta \end{aligned}$$

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:

$$\begin{array}{ccc} R_{t_i} \to \epsilon & R_{t_i} \to R_{t_i} t_i^+ & R_{t_i} \to R_{t_i} t_i^- \\ R_{\alpha\wedge\beta} \to R_\alpha R_\beta & R_{\alpha\vee\beta} \to R_\alpha R_\beta & \end{array}$$

Example 10. Continuing our running example, H includes among others the following rules:

$$\begin{array}{rr} F_{(t_1\wedge t_2)\vee(t_3\wedge t_4)} \to F_{(t_1\wedge t_2)} R_{(t_3\wedge t_4)} & F_{(t_1\wedge t_2)\vee(t_3\wedge t_4)} \to R_{(t_1\wedge t_2)} F_{(t_3\wedge t_4)} \\ F_{(t_1\wedge t_2)\vee(t_3\wedge t_4)} \to F_{(t_1\wedge t_2)} F_{(t_3\wedge t_4)} & R_{(t_1\wedge t_2)} \to R_{t_1} R_{t_2} \\ R_{t_1} \to R_{t_1} t_1^+ & R_{t_2} \to R_{t_2} t_2^+ \\ R_{t_1} \to R_{t_1} t_1^- & R_{t_2} \to R_{t_2} t_2^- \\ R_{t_1} \to \epsilon & R_{t_2} \to \epsilon \end{array}$$

In particular,

$$\begin{aligned} \mathcal{L}(H, F_\psi) = \{(n_1^+, n_1^-, n_2^+, n_2^-, 0, 0, 0, 0) \in \Sigma^\odot : n_1^+, n_1^-, n_2^+, n_2^- \in \mathbb{N}\} \cup \\ \cup \{(0, 0, 0, 0, n_3^+, n_3^-, n_4^+, n_4^-) \in \Sigma^\odot : n_3^+, n_3^-, n_4^+, n_4^- \in \mathbb{N}\}, \end{aligned}$$

i.e., F_ψ may produce arbitrary quantities of either t_1 and t_2, or t_3 and t_4, reflecting the Boolean structure of ψ. ♢

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 have $w(t_i) \geq w_Y(t_i)$ for those t_i. 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 11. $\mathcal{L}(H, I) = \{(n_1^+, n_1^-, \ldots, n_k^+, n_k^-) \in \Sigma^\odot : n_j^+ \geq n_j^-,\ 1 \leq j \leq k\}$.

This completes the construction of H. Before we prove two lemmas that establish the correctness of our construction, let us illustrate the necessity of I with the help of an example.

Example 12. For ϕ of our running example, suppose that x is even and hence

$$\begin{aligned} w = (c + (1+c)\cdot x,\ c + c\cdot x,\ c + c\cdot x,\ c + (1+c)\cdot x, \\ c + 1 + (1+c)\cdot x,\ c + c\cdot x,\ c + c\cdot x,\ c + 1 + (1+c)\cdot x) \in \mathcal{L}(G). \end{aligned}$$

For $y = x/2$, we clearly have

$$w_Y = (2\cdot y, 0, 0, 2\cdot y, 2\cdot y, 0, 0, 2\cdot y) \in \mathcal{L}(H, Y).$$

The value of $w_Y(t_1)$ and $w_Y(t_2)$ already matches $w(t_1)$ and $w(t_2)$, respectively, however $w_Y(t_1^+) \neq w(t_1^+)$, etc. But then we find

$$w_I = (2\cdot y + 1)\cdot(c, c, c, c, 0, 0, 0, 0) \in \mathcal{L}(H, I).$$

Moreover, we have

$$w_F = (0,0,0,0,1,0,0,1) + (2 \cdot y + 1) \cdot (0,0,0,0,c,c,c,c) \in \mathcal{L}(H, F_\psi),$$

and consequently $w_Y + w_F + w_I = w \in \mathcal{L}(H)$. $\Diamond$

Lemma 13. Suppose $\mathcal{L}(G) \subseteq \mathcal{L}(H)$, then $\phi = \forall \boldsymbol{x}.\exists \boldsymbol{y}.\psi(\boldsymbol{x}, \boldsymbol{y})$ is valid.

Proof. Let $\boldsymbol{x} \in \mathbb{N}^m$, we show how to construct $\boldsymbol{y} \in \mathbb{N}^n$ such that $\psi(\boldsymbol{x}, \boldsymbol{y})$ evaluates to true. By Lemma 8(i), there exists $w \in \mathcal{L}(G)$ such that for all $1 \leq i \leq k$,

$$w(t_i) = \sum_{1 \leq j \leq m} (a_{i,j}^+ - a_{i,j}^-) \cdot x_j + z_i^+ - z_i^-, \tag{8}$$

and since $\mathcal{L}(G) \subseteq \mathcal{L}(H)$, $w \in \mathcal{L}(H)$. By definition of H, there are $w_Y, w_F, w_I \in \Sigma^\odot$ such that

- $w_Y \in \mathcal{L}(H, Y)$, $w_I \in \mathcal{L}(H, I)$, $w_F \in \mathcal{L}(H, F_\psi)$; and
- $w = w_Y + w_F + w_I$.

By Lemma 9(ii), there exists $\boldsymbol{y} \in \mathbb{N}^n$ such that for all $1 \leq i \leq k$,

$$w_Y(t_i) = \sum_{1 \leq j \leq n} (b_{i,j}^+ - b_{i,j}^-) \cdot y_j. \tag{9}$$

We claim that $\boldsymbol{y}$ has the desired properties, i.e., that $w(t_i) \geq w_Y(t_i)$ for all inequalities necessary to make $\psi(\boldsymbol{x}, \boldsymbol{y})$ evaluate to true. To this end, consider the derivation tree of w_F and define a mapping $\xi : \{1, \ldots, k\} \to \{0, 1\}$ such that $\xi(i) \overset{\text{def}}{=} 0$ if the non-terminal R_{t_i} occurs in the derivation tree and $\xi(i) \overset{\text{def}}{=} 1$ if F_{t_i} occurs in it. By the construction of F_ψ, this mapping is well-defined, and moreover it also implies that $\psi[\xi(t_1)/t_1, \ldots, \xi(t_k)/t_k]$ evaluates to true. So it remains to show that for all t_i such that $\xi(i) = 1$, i.e., we have

$$w(t_i) = w(t_i^+) - w(t_i^-) \geq w_Y(t_i^+) - w_Y(t_i^-) = w_Y(t_i).$$

Since for all such i we have $w_F(t_i^+) = w_F(t_j^-) = 0$, it follows that $w(t_i^+) = w_Y(t_i^+) + w_I(t_i^+)$ and $w(t_i^-) = w_Y(t_i^-) + w_I(t_i^-)$, hence

$$w(t_i^+) - w(t_i^-) = (w_Y(t_i^+) - w_Y(t_i^-)) + (w_I(t_i^+) - w_I(t_i^-)).$$

By Lemma 11, $w_I(t_i^+) - w_I(t_i^-) \geq 0$, and consequently $w(t_i) \geq w_Y(t_i)$ as required by (8) and (9). $\square$

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 reduction step made by G together with Proposition 5 allows us to overcome this obstacle.

Lemma 14. 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 8(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

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

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 Proposition 5, there are $B, P \subseteq \mathbb{N}^{m+n}$ such that $[\![D]\!] = B + \text{cone}(P)$. 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 $|P| \leq \binom{m+n}{h} \leq 2^{|\phi|}$ we have

$$0 \leq \sum_{1 \leq i \leq \ell} \lambda_i \leq \|\boldsymbol{x}^*\|_\infty \cdot \ell \leq \|\boldsymbol{x}^*\|_\infty \cdot 2^{|\phi|}. \tag{10}$$

Now let

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

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. Moreover,

$$\begin{aligned}
\|\boldsymbol{y}^\dagger\|_\infty &\leq \|\boldsymbol{b}\|_\infty + \|\lambda_1 \cdot \boldsymbol{p}_1 + \cdots + \lambda_\ell \cdot \boldsymbol{p}_\ell\|_\infty \\
&\leq \|B\|_\infty + \sum_{1 \leq i \leq \ell} \lambda_i \cdot \|P\|_\infty \\
&\leq \|B\|_\infty + \|\boldsymbol{x}^*\|_\infty \cdot 2^{|\phi|} \cdot \|P\|_\infty && \text{(by (10))} \\
&\leq \left(1 + \|\boldsymbol{x}^*\|_\infty \cdot 2^{|\phi|}\right) \cdot (\|A\|_{1,\infty} + \|\boldsymbol{z}\|_\infty + 2)^{h+m+n} && \text{(by Prop. 5)} \\
&\leq \left(1 + \|\boldsymbol{x}^*\|_\infty \cdot 2^{|\phi|}\right) \cdot ((m+n+1) \cdot \|\phi\|_\infty + 2)^{k+m+n} \\
&\leq \left(1 + \|\boldsymbol{x}^*\|_\infty \cdot 2^{|\phi|}\right) \cdot |\phi|^{3 \cdot |\phi|} \\
&\leq (1 + \|\boldsymbol{x}^*\|_\infty) \cdot |\phi|^{3 \cdot |\phi|} \cdot 2^{|\phi|} && \text{(by (1))} \\
&\leq (1 + \|\boldsymbol{x}^*\|_\infty) \cdot \frac{c}{|\phi|^2} && \text{(by (2))}
\end{aligned}$$

Combining the estimation of $\|\boldsymbol{y}^\dagger\|_\infty$ with (6) and (7) of Lemma 8, for every $1 \leq i \leq k$ we obtain

$$w(t_i^+), w(t_i^-) \geq c \cdot (1 + \|\boldsymbol{x}^*\|_\infty) \geq \|\boldsymbol{y}^\dagger\|_\infty \cdot |\phi|^2 \geq \|\boldsymbol{y}^\dagger\|_\infty \cdot \|\phi\|_\infty \cdot |\phi|. \tag{11}$$

By Lemma 9(i) there is $w_Y \in \mathcal{L}(H, Y)$ such that (11) yields

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

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 11 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. □

Lemmas 13 and 14 together with Proposition 4 yield the coNEXP-lower bound of Theorems 1 and 2 of the language inclusion problem for context-free and exponent-sensitive grammars. In order to show hardness of the equivalence problem, we merge H into G, i.e., define

$$G^e \stackrel{\text{def}}{=} (N_G \cup N_H \cup \{S\}, \Sigma, S, P_G \cup P_H \cup \{S \to S_G, S \to S_H\}).$$

It is now clear that ϕ is valid iff $\mathcal{L}(G^e) = \mathcal{L}(H)$. Finally, if we, in addition, redefine H as

$$H^e \stackrel{\text{def}}{=} (\{S_G, X\} \cup N_H \cup \{S\}, \Sigma, P_H \cup \{S \to S_G, S \to X S_H, X \to \epsilon\})$$

then G^e and H^e have the same set of non-terminals $N \stackrel{\text{def}}{=} N_G \cup N_H \cup \{S\} = \{S_G, X\} \cup N_H \cup \{S\}$, and even a stronger statement holds:

$$\phi \text{ is valid} \iff \mathcal{R}(G^e) = \mathcal{R}(H^e). \tag{12}$$

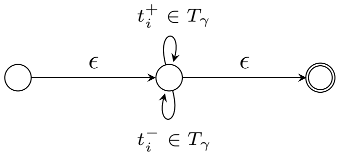
3.1. **Hardness for Regular Commutative Grammars.** It remains to show how the reduction developed so far can be adapted in order to prove coNEXP-hardness of the equivalence problem for regular commutative grammars. As constructed above, neither G nor H are regular. In this section, we show how to obtain regular commutative grammars G^r and H^r from G and H such that $\mathcal{L}(G^r) \subseteq \mathcal{L}(H^r)$ iff $\mathcal{L}(G) \subseteq \mathcal{L}(H)$.

It is actually not difficult to see that H can be made regular. By the construction of H, both gadgets starting in Y and I are regular, but F_ψ is not, and also the initial production $S_H \to Y F_\psi I$ is not regular. The latter can be fixed by additionally adding productions $Y \to F_\psi$ and $F_\psi \to I$ to the set of productions P, and replacing $S_H \to Y F_\psi I$ with $S_H \to Y$. It thus remains to make F_ψ regular. The non-regularity of the latter is due to the fact that we use branching provided by context-free commutative grammars in order to simulate the Boolean structure of ψ. However, as we show now, it is possible to serialise F_ψ.

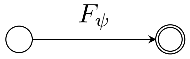
As a first step, we discuss the serialisation of R_γ for all subformulas γ occurring in the syntax-tree of ψ. Recall that the task of R_γ is to generate arbitrary amounts of alphabet symbols t_i^+ and t_i^- for all t_i occurring in γ. We define the set T_γ collecting all t_i^+ and t_i^- corresponding to the inequalities appearing in γ:

$$T_\gamma \overset{\text{def}}{=} \begin{cases} \{t_i^+, t_i^-\} & \text{if } \gamma = t_i \\ T_\alpha \cup T_\beta & \text{if } \gamma = \alpha \wedge \beta \text{ or } \gamma = \alpha \vee \beta. \end{cases}$$

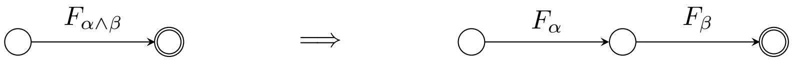
We can now redefine R_γ to be the regular grammar corresponding to the following NFA, for which clearly $\mathcal{L}(R_\gamma) = T_\gamma^\odot$ holds:



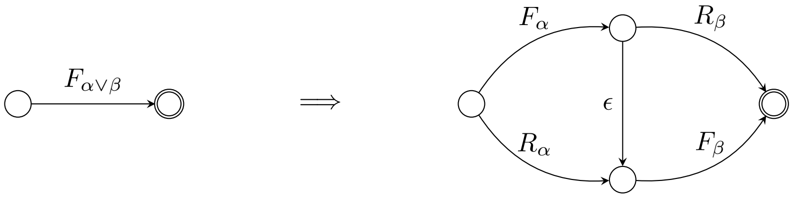
Next, we describe an inductive procedure that when completed yields an NFA that corresponds to a regular grammar whose language is equivalent to $\mathcal{L}(H, F_\psi)$. The procedure constructs in every iteration an NFA with a unique incoming and outgoing state and labels every transition with the gadget that should replace this transition in the next iteration, or with an alphabet letter if no more replacement is required. The initial such NFA is the following:



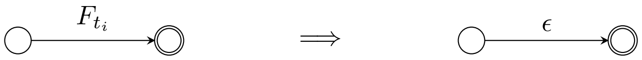
In the induction step, the rewriting of a transition labelled with F_γ depends on the logical connective. A conjunction $\gamma = \alpha \wedge \beta$ is replaced by sequential composition:



Thus, the outgoing state of the gadget F_α connects to the incoming state of the gadget F_β. In the case of a disjunction $\gamma = \alpha \vee \beta$, the transition is rewritten into three paths that, informally speaking, correspond to possible truth assignments to the subformulas F_α and F_β. If the inequalities appearing in α are allowed to receive arbitrary values, the transition labelled with $F_{\alpha\vee\beta}$ is replaced by the sequential composition of two gadgets, R_α and F_β; the other cases are treated in the same way:



Once some F_{t_i} is reached, it gets replaced by the empty word:



Moreover, if R_γ is reached it gets replaced by the gadget described earlier. From this construction, it is clear that the regular grammar that can be obtained from

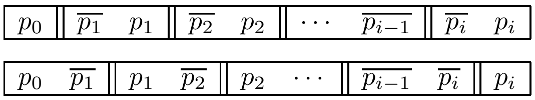

p_0 | $\overline{p_1}$ p_1 | $\overline{p_2}$ p_2 | $\cdots$ $\overline{p_{i-1}}$ | $\overline{p_i}$ p_i
p_0 $\overline{p_1}$ | p_1 $\overline{p_2}$ | p_2 $\cdots$ | $\overline{p_{i-1}}$ $\overline{p_i}$ | p_i

FIGURE 1. Illustration of the pairing of alphabet symbols in C_ℓ (above) and C_r (below).

the resulting NFA exactly generates $\mathcal{L}(H, F_\psi)$, and that in the following we may assume that H is regular.

We now turn towards showing how G can be made regular. Even though the structure of G already appears to be regular, note that we use the construction of Remark 3 in order to encode the constant c in binary. This is not possible in the case of regular commutative grammar, at least not in an obvious way. However, we can use an interplay between G and H in order to, informally speaking, force G to produce alphabet symbols in exponential quantities. To this end, we introduce additional alphabet symbols and define $\Gamma_i \overset{\text{def}}{=} \{p_0, \overline{p_1}, p_1, \ldots, \overline{p_i}, p_i\}$ for every $i \in \mathbb{N}$. Before formally providing the construction in Lemma 15 below, let us discuss how we can achieve our goal on an informal level. Suppose we wish to produce a word $w \in \Gamma_i^\odot$ such that $w(p_i) = 2^i \cdot p_0$. One way to obtain a language that contains such a word is to pair alphabet symbols $\overline{p_j}$ and p_j and to produce two symbols p_j every time some $\overline{p_j}$ is non-deterministically produced. The pairing is illustrated in the top of Figure 1, and, more formally, such a language can be generated by the following regular grammar: $C_\ell \overset{\text{def}}{=} (\{S_\ell\}, \Gamma_i, S_\ell, P_\ell)$, where

$$\begin{aligned} S_\ell &\to \epsilon \\ S_\ell &\to S_\ell p_0 \\ S_\ell &\to S_\ell \overline{p_j} p_j p_j && (0 \leq j \leq i). \end{aligned}$$

Clearly, we can find some $w \in \mathcal{L}(C_\ell)$ such that

$$w(p_i) = 2 \cdot w(\overline{p_i}) = 2 \cdot w(p_{i-1}) = 2^2 \cdot w(\overline{p_{i-1}}) = \cdots = 2^i \cdot w(p_0).$$

Such a w implicitly requires another pairing, namely that $w(\overline{p_{j+1}}) = w(p_j)$ for all $0 \leq j < i$, which is illustrated in the bottom of Figure 1. If we can rule out all words of $\mathcal{L}(C_\ell)$ that violate this pairing, we obtain a language containing the desired $w \in \Gamma_i^\odot$ such that $w(p_i) = 2^i \cdot w(p_0)$. This is the task of the regular grammar C_r constructed in the following lemma.

Lemma 15. For every $i \in \mathbb{N}$, there are logarithmic-space computable regular commutative grammars C_ℓ and C_r such that:

(i) $\mathcal{L}(C_\ell) = \{w \in \Gamma_i^\odot : w(p_j) = 2 \cdot w(\overline{p_j})$ for every $1 \leq j \leq i\}$; and
(ii) $\mathcal{L}(C_r) = \{w \in \Gamma_i^\odot : w(p_j) \neq w(\overline{p_{j+1}})$ for some $0 \leq j < i\}$.

In particular, for every $v \in \mathcal{L}(C_\ell) \setminus \mathcal{L}(C_r)$, $v(p_i) = 2^i \cdot v(p_0)$.

Proof. Regarding Part (i), clearly C_ℓ as defined above has the desired properties. In order to prove Part (ii), we define $C_r \overset{\text{def}}{=} (N_r, \Gamma_i, S_r, P_r)$, where $N_r \overset{\text{def}}{=} \{S_r\} \cup$

$\{N_j, \overline{N_{j+1}} : 0 \leq j < i\}$ and

$$\begin{array}{lll}
S_r \to S_r p_i & & \\
S_r \to N_j p_j & S_r \to \overline{N_{j+1}}\ \overline{p_{j+1}} & (0 \leq j < i) \\
N_j \to N_j p_j & \overline{N_{j+1}} \to \overline{N_{j+1}}\overline{p_{j+1}} & (0 \leq j < i) \\
N_j \to N_j p_j \overline{p_{j+1}} & \overline{N_{j+1}} \to \overline{N_{j+1}} p_j \overline{p_{j+1}} & (0 \leq j < i) \\
N_j \to N_j p_g & \overline{N_j} \to \overline{N_j} p_g & (0 \leq g, j < i, g \neq j) \\
N_j \to N_j \overline{p_{g+1}} & \overline{N_j} \to \overline{N_j}\overline{p_{g+1}} & (0 \leq g, j < i, g \neq j) \\
N_j \to \epsilon & \overline{N_j} \to \epsilon & (0 \leq j < i).
\end{array}$$

Informally speaking, after non-deterministically producing alphabet symbols p_i starting from S_r, we can then non-deterministically choose an index $0 \leq j < i$ such that either $p_j > \overline{p_{j+1}}$ (when switching to N_j) or $\overline{p_{j+1}} > p_j$ (when switching to $\overline{N_j}$), and for any choice of j all other alphabet symbols p_g and $\overline{p_{g+1}}$ such that $g \neq j$ can be produced in arbitrary quantities. It is easily checked that $\mathcal{L}(C_r)$ has the desired properties. Now, we have

$$\begin{aligned}
& v \in \mathcal{L}(C_\ell) \setminus \mathcal{L}(C_r) \\
\Longleftrightarrow\ & v \in \mathcal{L}(C_\ell) \cap \overline{\mathcal{L}(C_r)} \\
\Longleftrightarrow\ & v \in \mathcal{L}(C_\ell) \cap \{w \in \Gamma_i^\odot : w(p_j) = w(\overline{p_{j+1}}) \text{ for all } 0 \leq j < i\} \\
\Longleftrightarrow\ & v \in \{w \in \Gamma_i^\odot : w(p_{j+1}) = 2 \cdot w(\overline{p_{j+1}}) \text{ and } w(p_j) = w(\overline{p_{j+1}}) \text{ for all } 0 \leq j < i\} \\
\Longrightarrow\ & v \in \{w \in \Gamma_i^\odot : w(p_{j+1}) = 2 \cdot w(p_j) \text{ for all } 0 \leq j < i\} \\
\Longrightarrow\ & v(p_i) = 2^i \cdot v(p_0).
\end{aligned}$$

□

Let $\Sigma = \{t_1^+, t_1^-, \ldots, t_k^+, t_k^-\}$ be as defined in the previous section. The following corollary is an immediate consequence of Lemma 15 and enables us to construct an exponential number of t_i^+ and t_i^-.

Corollary 16. For every $i \in \mathbb{N}$, there are logarithmic-space computable regular commutative grammars C_ℓ^Σ and C_r^Σ over $\Sigma \cup \Gamma_i$ such that

(i) $\mathcal{L}(C_\ell^\Sigma) = \mathcal{L}(C_\ell) \cdot \Sigma^\odot \cap \{w \in (\Sigma \cup \Gamma_i)^\odot : w(t_j^+) = w(t_j^-) = w(p_i)\}$; and
(ii) $\mathcal{L}(C_r^\Sigma) = \mathcal{L}(C_r) \cdot \Sigma^\odot$.

where C_ℓ and C_r are defined as in Lemma 15.

Recall that H already is a regular commutative grammar, and let c be the constant from (2) and $j \overset{\text{def}}{=} \log c$. We can now define regular versions G^r and H^r over $\Sigma \cup \Gamma_j$ of G and H, respectively, such that $\mathcal{L}(G^r) \subseteq \mathcal{L}(H^r)$ iff $\mathcal{L}(G) \subseteq \mathcal{L}(H)$. Let u and v_i be defined as in (3) and (4), and let C_ℓ^Σ and C_r^Σ be as defined in Corollary 16 for the alphabet Γ_j, the axiom of G^r is S_G^r and the transitions of G^r are given by

$$\begin{array}{lll}
S_G^r \to X p_0 u & X \to X p_0 v_j & (1 \leq j \leq m) \\
X \to C_\ell^\Sigma & &
\end{array}$$

Moreover, H^r is the regular commutative grammar such that

$$\mathcal{L}(H^r) = \mathcal{L}(C_r^\Sigma) \cup \mathcal{L}(H) \cdot \Gamma_j^\odot.$$

The correctness of the construction can be seen as follows. Let $w \in \mathcal{L}(G^r)$, we distinguish two cases:

(i) If $w(p_i) \neq c \cdot w(p_0)$ then $w \in \mathcal{L}(C_r^\Sigma) \subseteq \mathcal{L}(H^r)$.
(ii) Otherwise, $w(p_i) = c \cdot w(p_0)$ and $w \notin \mathcal{L}(C_r^\Sigma)$. Consequently, $w \in \mathcal{L}(H^r)$ iff $w \in \mathcal{L}(H) \cdot \Gamma_j^{\odot}$ thus $\pi_\Sigma(w) \in \mathcal{L}(H)$. But we know that $\pi_\Sigma(w) \in \mathcal{L}(G)$.

Concluding, we have $\mathcal{L}(G^r) \subseteq \mathcal{L}(H^r)$ if $\mathcal{L}(G) \subseteq \mathcal{L}(H)$. The implication in the opposite direction is obvious, which completes our proof.

4. Improved Complexity Bounds for Language Equivalence for Exponent-Sensitive Commutative Grammars

In this section, we turn towards the equivalence problem for exponent-sensitive commutative grammars and prove Theorem 2. Hardness for coNEXP of this problem directly follows from Theorem 1, since regular commutative grammars are a subclass of exponent-sensitive grammars. Hence, here we show that the problem can be decided in co-2NEXP, thereby improving the 2EXPSPACE upper bound from [MW13a]. As stated in Section 2, commutative words on the left-hand sides of the productions of exponent-sensitive commutative grammar as defined in [MW13a] are encoded in binary.

It is sufficient to show that inclusion between exponent-sensitive commutative grammars can be decided in co-2NEXP. To this end, we follow an approach proposed by Huynh used to show that inclusion of context-free commutative grammars is in coNEXP [Huy85]. 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, $\mathcal{R}(G)$ and $\mathcal{R}(H)$ are shown semi-linear with a representation size doubly exponential in $\#G + \#H$ in [MW13b], 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 [Huy86] 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 [MW13b, Thm. 5.5]. Now comes the second part of Huynh's approach into play. In [Huy85], 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 obviously is 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 $\#G + \#H$. 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 polynomial time in $\#w$ that w belongs to $\mathcal{L}(G)$ and not to $\mathcal{L}(H)$. We now proceed with the formal details.

Subsequently, let $s \stackrel{\text{def}}{=} \#G$, $t \stackrel{\text{def}}{=} \#H$ and $\mathcal{L}(G), \mathcal{L}(H) \subseteq \Sigma^{\odot}$. We begin with stating the relevant facts about the semi-linear representation of the reachability set of exponent-sensitive commutative grammars. The subsequent proposition is derived from [MW13b, Lem. 5.4], which is stated in terms of generalised communication-free Petri nets, but as argued in the proof of [MW13b, Thm. 6.1], there is a logarithmic-space reduction from exponent-sensitive commutative grammars to such Petri nets which preserves reachability sets, and hence allows us to apply [MW13b, Lem. 5.4].

Proposition 17 ([MW13b])**.** There exists a fixed polynomial p such that the reachability set $\mathcal{R}(G) = \bigcup_{i \in I} L(\boldsymbol{b}_i, Q_i)$ is computable in $\mathsf{DTIME}(2^{2^{\mathsf{poly}(s)}})$ such that for every $i \in I$,

- $|I| \leq 2^{2^{p(s)}}$ and $|Q_i| \leq 2^{p(s)}$; and
- $\#\boldsymbol{b}_i \leq p(s)$ and $\#\boldsymbol{q} \leq p(s)$ for every $\boldsymbol{q} \in Q_i$.

Next, we introduce Huynh's decomposition of linear sets as described above. The following proposition is a consequence and a summary of Proposition 2.6 and Lemmas 2.7 and 2.8 in [Huy85].

Proposition 18 ([Huy85])**.** Let $M = L(\boldsymbol{b}, Q)$ be a linear set. There is a fixed polynomial p such that $M = \bigcup_{i \in I} M_i$ and for every $i \in I$, $M_i = L(\boldsymbol{b}_i, Q_i)$ with

- $\boldsymbol{b}_i \in L(\boldsymbol{b}, Q)$ and $\#\boldsymbol{b}_i \leq p(\#M)$; and
- $Q_i \subseteq Q$ is has full column rank and $|Q_i| = \operatorname{rank}(Q)$.

Subsequently, for a given $M = L(\boldsymbol{b}, Q)$, whenever $\bigcup_{i \in I} L(\boldsymbol{b}_i, Q_i)$ has the properties described in Proposition 18, we say that it is the Huynh representation of M.

Lemma 19. Let $M = L(\boldsymbol{b}, Q)$ be a linear set. The Huynh representation of M can be computed $\mathsf{DTIME}(2^{\mathsf{poly}(\#M)})$.

Proof. Let p be the polynomial from Proposition 18. First, we compute the set of $\boldsymbol{b}_i$ as follows: we enumerate all candidates $\boldsymbol{b}_i$ such that $\#\boldsymbol{b}_i \leq p(\#M)$, there is at most an exponential number of them. For every candidate we check if $\boldsymbol{b}_i \in L(\boldsymbol{b}, Q)$, which can be done in NP. Next, we enumerate all subsets $Q_i \subseteq Q$ of full column rank and cardinality $\operatorname{rank}(Q)$, again there are at most exponentially many of them. Finally, we output the all possible combinations of the $\boldsymbol{b}_i$ with the Q_i. □

Lemma 20. The Huynh representation of $\mathcal{R}(G)$ can be computed in $\mathsf{DTIME}(2^{2^{\mathsf{poly}(s)}})$.

Proof. First, we apply Proposition 17 in order to compute a semi-linear representation $\bigcup_{i \in I} L(\boldsymbol{b}_i, Q_i)$ of $\mathcal{R}(G)$ such that $|I| \leq 2^{2^{p(s)}}$, $|Q_i| \leq 2^{p(s)}$, and $\#\boldsymbol{b}_i \leq p(s)$ and $\#\boldsymbol{q} \leq p(s)$ for every $\boldsymbol{q} \in Q_i$ for some fixed polynomial p. By Lemma 19, from every $M_i = L(\boldsymbol{b}_i, Q_i)$ we can compute an equivalent Huynh representation $N_i = \bigcup_{j \in J_i} L(\boldsymbol{c}_{i,j}, R_{i,j})$ of M_i in $\mathsf{DTIME}(2^{\mathsf{poly}(\#M_i)}) = \mathsf{DTIME}(2^{2^{\mathsf{poly}(s)}})$. Thus, the overall procedure also runs in $\mathsf{DTIME}(2^{2^{\mathsf{poly}(s)}})$. □

As the final ingredient, we state Huynh's result that whenever inclusion between two semi-linear sets does not hold then there exists a witness of polynomial bit-size.

Proposition 21 ([Huy86])**.** Let $M, N \subseteq \mathbb{N}^m$ be semi-linear sets. There is a fixed polynomial p such that whenever $M \not\subseteq N$ then there exists some $\boldsymbol{v} \in M \setminus N$ such that $\#\boldsymbol{v} \leq p(\#M + \#N)$.

We are now fully prepared to prove the main statement of this section, which immediately yields the upper bound for Theorem 2.

Proposition 22. Deciding $\mathcal{L}(G) \subseteq \mathcal{L}(H)$ is in $\mathsf{co\text{-}2NEXP}$.

Proof. We describe a $\mathsf{co\text{-}2NEXP}$-algorithm. First, by combining Proposition 17 with Proposition 21, if $\mathcal{L}(G) \not\subseteq \mathcal{L}(H)$ then there is some $w \in \Sigma^{\odot}$ such that $\#w \leq 2^{2^{p(s+t)}}$ for some fixed polynomial p. The algorithm non-deterministically chooses such a w. Now the algorithm computes the Huynh representations of $\mathcal{R}(G)$ and $\mathcal{R}(H)$ in $\mathsf{DTIME}(2^{2^{\mathsf{poly}(s+t)}}) = \mathsf{DTIME}(\mathsf{poly}(\#w))$. For every linear set $M = L(\boldsymbol{b}, Q)$ in the Huynh representation of $\mathcal{R}(G)$ and $\mathcal{R}(H)$, $\#\boldsymbol{b} \leq p(s+t)$, $\#\boldsymbol{q} \leq p(s+t)$ for all $\boldsymbol{q} \in Q$ and some fixed polynomial p, and Q has full column rank and hence $|Q| \leq |\Sigma|$. Thanks to those properties, $w \in L(\boldsymbol{b}, Q)$ can be decided in $\mathsf{DTIME}(\mathsf{poly}(\#M))$ using Gaussian elimination. Consequently, checking $w \in \mathcal{L}(G) \setminus \mathcal{L}(H)$ can be performed in $\mathsf{DTIME}(\mathsf{poly}(\#w))$. □

5. Applications to Equivalence Problems for Classes of Petri Nets, BPPs and Reversal-Bounded Counter Automata

Here, we discuss immediate corollaries of Theorems 1 and 2 for equivalence problems for various classes of Petri nets, basic parallel process nets (BPP-nets) and reversal-bounded counter automata.

It has, for instance, been observed in [Esp97, Yen97, MW15] that context-free commutative grammars can be seen as notational variants of communication-free Petri nets and BPP-nets. This allows for transferring results on standard decision problems between these formalisms. We do not formally introduce communication-free Petri nets and BPP-nets here. Informally speaking, in those nets non-terminal and terminal symbols of commutative context-free grammars correspond to places in those nets, where every transition can remove a token from at most one place, and where tokens cannot be removed from places corresponding to terminal symbols. The equivalence problem for communication-free Petri nets and BPP-nets is to decide whether the set of reachable markings of two given nets coincides. In particular, this requires that both nets have the same set of places. Via a reduction to the equivalence problem for context-free commutative grammars, it is possible to obtain a coNEXP-upper bound for the equivalence problem of communication-free Petri nets and BPP-nets [MW15]. On the other hand, Theorem 1 together with the strengthened construction given in (12) yields a matching lower bound.

Theorem 23. The equivalence problem for communication-free Petri nets and BPP-nets is coNEXP-complete.

As already briefly mentioned in Section 4, exponent-sensitive commutative grammars are closely related to so-called generalised communication-free Petri nets, and in fact inter-reducible with them [MW13b, Thm. 6.1]. Similarly to communication-free Petri nets, transitions of generalised communication-free Petri nets may only remove tokens from one place, however they are allowed to remove an arbitrary number of them, not just one. For the sake of completeness, let us state Theorem 2 in terms of generalised communication-free Petri nets.

Theorem 24. The equivalence problem for generalised communication-free Petri nets is coNEXP-hard and in $\mathsf{co\text{-}2NEXP}$.

We now turn towards the equivalence problem for reversal-bounded counter automata. For our purposes, it is sufficient to introduce reversal-bounded counter automata on an informal level, formal definitions can be found in [Iba78], see also [Iba14] for a recent survey on decision problems for reversal-bounded counter automata. A counter automaton comprises a finite-state controller with a finite number of counters ranging over the natural numbers that can be incremented, decremented or tested for zero along a transition. A classical result due to Minsky states that reachability in counter automata is undecidable already in the presence of two counters [Min61]. One way of overcoming this problem is to bound the number of times a counter is allowed to switch between increasing and decreasing mode. Consider, for instance, a counter of some counter automaton whose values along a run are $0, 1, 2, 3, 5, 4, 4, 3, 3, 4, 5, 6$. On this run, the counter reverses its mode twice, once from incrementing to decrementing mode (when decrementing from 5 to 4), and one more time from decrementing to incrementing mode (when, thereafter, incrementing from 3 to 4). In this example, the number of reversals of the counter is bounded by two. A k-reversal bounded counter automaton is a counter automaton whose counters are only allowed to have at most k reversals along a run.

Ibarra [Iba78] has shown that the sets of reachable configurations of reversal-bounded counter automata are effectively semi-linear. Hague and Lin [HL11] showed that from a reversal-bounded counter automaton one can construct in polynomial time an open existential Presburger formula $\varphi(\boldsymbol{x})$ defining the set of reachable configurations, i.e., the sets of counter values with which a target control state is reached starting in an initial configuration. The equivalence problem for reversal bounded counter automata over the same number of counters is to decide whether their reachability sets are the same.

An application of the result of Hague and Lin combined with Proposition 4 immediately yields a coNEXP-upper bound for the equivalence problem. Given two reversal-bounded counter automata whose reachability sets are defined by existential Presburger formulas $\varphi(\boldsymbol{x})$ and $\psi(\boldsymbol{x})$, respectively, their reachability set is equivalent iff $\phi \overset{\text{def}}{=} \forall \boldsymbol{x}.\phi(\boldsymbol{x}) \leftrightarrow \psi(\boldsymbol{x})$ is valid. Since ϕ is a Π_2-sentence of Presburger arithmetic, Proposition 4 yields a coNEXP-upper bound for the equivalence problem. On the other hand, Theorem 1 gives that equivalence is already coNEXP-hard for regular commutative grammars. Now the latter can immediately be simulated by a 0-reversal bounded counter automaton by introducing one counter for each alphabet symbol in Σ, treating non-terminal symbols as control states, and incrementing the counter corresponding to some $a \in \Sigma$ whenever a production $V \to Wa$ is simulated. Consequently, we have proved the following theorem.

Theorem 25. The equivalence problem for reversal-bounded counter automata is coNEXP-complete, and in particular coNEXP-hard for 0-reversal bounded counter automata with no zero tests whose constants are encoded in unary.

6. Conclusion

In this paper, 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 [Huy85]. Our lower bound also carries over to the equivalence problem for exponent-sensitive commutative grammars, for which we could also improve the $\mathsf{2\text{-}EXPSPACE}$-upper bound [MW13a] to $\mathsf{co\text{-}2NEXP}$. The precise complexity of this problem remains an open problem of this paper. An

com. grammar	word problem	language equivalence
type-0	EXPSPACE-h. [Lip76], $\in \mathbf{F}_{\omega^3}$ [LS15]	undecidable [Hac76]
cont.-sensitive	PSPACE-complete [Huy83]	undecidable [Huy85]
exp.-sensitive	PSPACE-complete [MW13a]	coNEXP-h., $\in$ co-2NEXP
context-free	NP-complete [Huy83, Esp97]	coNEXP-complete
regular		

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

overview over the complexity of word and equivalence problems for commutative grammars together with references to the literature is provided in Table 1.

It is interesting to note the non-monotonic behaviour of regular grammars with respect to the complexity of the equivalence problem. In the non-commutative setting, language equivalence is PSPACE-complete, and hardness even holds when the number of alphabet symbols is fixed. In contrast, in the commutative setting language equivalence is Π_2^{P}-complete when the number of alphabet symbols is fixed [KT10, Kop15], and coNEXP-complete for an alphabet of arbitrary size as shown in this paper.

One major open problem related to the problems discussed in this paper is weak bisimilartiy between basic parallel processes. This problem is not known to be decidable and PSPACE-hard [Srb03]. Unfortunately, it does not seem possible to adjust the construction of our coNEXP-lower bound to also work for weak bisimulation.

Acknowledgements. We would like to thank Matthew Hague for clarifying some questions regarding reversal-bounded counter automata.

References

[Dom91] Eric Domenjoud, *Solving systems of linear Diophantine equations: An algebraic approach*, Mathematical Foundations of Computer Science, MFCS, 1991, pp. 141–150.

[Esp97] Javier Esparza, *Petri nets, commutative context-free grammars, and basic parallel processes*, Fundamenta Informaticae **31** (1997), no. 1, 13–25.

[Gin66] Seymour Ginsburg, *The mathematical theory of context free languages*, McGraw-Hill, 1966.

[GJ79] M. R. Garey and David S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, W. H. Freeman, 1979.

[Grä89] Erich Grädel, *Dominoes and the complexity of subclasses of logical theories*, Annals of Pure Applied Logic **43** (1989), no. 1, 1–30.

[GS64] Seymour Ginsburg and Edwin H. Spanier, *Bounded ALGOL-like languages*, Transactions of the American Mathematical Society (1964), 333–368.

[Haa14] Christoph Haase, *Subclasses of Presburger arithmetic and the weak EXP hierarchy*, 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) (Thomas A. Henzinger and Dale Miller, eds.), ACM, 2014, pp. 47:1–47:10.

[Hac76] Michel Hack, *The equality problem for vector addition systems is undecidable*, Theoretical Computer Science **2** (1976), no. 1, 77–95.

[HH14] Christoph Haase and Simon Halfon, *Integer vector addition systems with states*, Proceedings of the 8th International Workshop on Reachability Problems (RP) (Joël Ouaknine, Igor Potapov, and James Worrell, eds.), Lecture Notes in Computer Science, vol. 8762, Springer, 2014, pp. 112–124.

[HL11] Matthew Hague and Anthony Widjaja Lin, *Model checking recursive programs with numeric data types*, Proccedings of the 23rd International Conference on Computer Aided Verification (CAV) (Ganesh Gopalakrishnan and Shaz Qadeer, eds.), Lecture Notes in Computer Science, vol. 6806, Springer, 2011, pp. 743–759.

[HMU03] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman, *Introduction to automata theory, languages, and computation - international edition (2. ed)*, Addison-Wesley, 2003.

[Huy83] Dung T. Huynh, *Commutative grammars: The complexity of uniform word problems*, Information and Control **57** (1983), no. 1, 21–39.

[Huy85] ———, *The complexity of equivalence problems for commutative grammars*, Information and Control **66** (1985), no. 1–2, 103–121.

[Huy86] ———, *A simple proof for the Σ_2^p upper bound of the inequivalence problem for semilinear sets*, Elektronische Informationsverarbeitung und Kybernetik **22** (1986), no. 4, 147–156.

[Iba78] Oscar H. Ibarra, *Reversal-bounded multicounter machines and their decision problems*, Journal of the ACM **25** (1978), no. 1, 116–133.

[Iba14] ———, *Automata with reversal-bounded counters: A survey*, Proceedings of the 16th International Workshop on Descriptional Complexity of Formal Systems (DCFS) (Helmut Jürgensen, Juhani Karhumäki, and Alexander Okhotin, eds.), Lecture Notes in Computer Science, vol. 8614, Springer, 2014, pp. 5–22.

[Kop15] Eryk Kopczyński, *Complexity of problems of commutative grammars*, Logical Methods in Computer Science **11** (2015), no. 1.

[KT10] Eryk Kopczyński and Anthony Widjaja To, *Parikh images of grammars: Complexity and applications*, Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, IEEE Computer Society, 2010, pp. 80–89.

[Lip76] Richard Lipton, *The reachability problem is exponential-space-hard*, Tech. report, Yale University, New Haven, CT, 1976.

[LS15] Jérôme Leroux and Sylvain Schmitz, *Reachability in vector addition systems demystified*, Proceedings of the 30th Annual ACM/IEEE Symposium on Logic In Computer Science (LICS), IEEE, 2015, To appear.

[Min61] Marvin L. Minsky, *Recursive unsolvability of post's problem of "tag" and other topics in the theory of turing machines*, The Annals of Mathematics **74** (1961), no. 3, 437–455.

[MW13a] Ernst W. Mayr and Jeremias Weihmann, *Completeness results for generalized communication-free Petri nets with arbitrary edge multiplicities*, Proceedings of the 7th International Workshop on Reachability Problems (RP) (Parosh Aziz Abdulla and Igor Potapov, eds.), Lecture Notes in Computer Science, vol. 8169, Springer, 2013, pp. 209–221.

[MW13b] ———, *Completeness results for generalized communication-free Petri nets with arbitrary edge multiplicities*, Tech. Report TUM-I1335, Technische Universität München, 2013.

[MW15] ———, *Complexity results for problems of communication-free Petri nets and related formalisms*, Fundamenta Informaticae **137** (2015), no. 1, 61–86.

[Pot91] Loïc Pottier, *Minimal solutions of linear Diophantine systems: Bounds and algorithms*, Proceedings of the 4th International Conference on Rewriting Techniques and Applications (RTA) (Ronald V. Book, ed.), Lecture Notes in Computer Science, vol. 488, Springer, 1991, pp. 162–173.

[Srb03] Jirí Srba, *Complexity of weak bisimilarity and regularity for BPA and BPP*, Mathematical Structures in Computer Science **13** (2003), no. 4, 567–587.

[Yen97] Hsu-Chun Yen, *On reachability equivalence for BPP-nets*, Theoretical Computer Science **179** (1997), no. 1-2, 301–317.

Laboratoire Spécification et Vérification (LSV), CNRS & ENS de Cachan, France