# Tightening the Complexity of Equivalence Problems for Commutative Grammars*

**Christoph Haase and Piotr Hofman**

**Laboratoire Spécification et Vérification (LSV), CNRS & ENS Cachan**
**Université Paris-Saclay, France**
**{haase,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 $\Pi_2^{\mathsf{P}}$ 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.xxx.yyy.p

## 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 $\Sigma$ 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 $\Sigma$. Parikh's theorem states that Parikh images of context-free languages are semi-linear sets. Since the latter are closed 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 sem-

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

inal 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 $\Pi_2^{\mathsf{P}}$-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 $\Pi_2^{\mathsf{P}}$-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 $\Pi_2^{\mathsf{P}}$-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 $\Pi_2$-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].

We only sketch some of the proofs in the main part of this paper. Full proofs can be found in the appendix.

## 2 Preliminaries

### 2.1 Commutative Grammars.

Let $\Sigma = \{a_1, \ldots, a_m\}$ be a finite alphabet. The free monoid generated by $\Sigma$ is denoted by $\Sigma^*$, and we denote by $\Sigma^\odot$ the free commutative monoid generated by $\Sigma$. 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 $\epsilon$, 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 $\Sigma$. For $\Gamma \subseteq \Sigma$, $\pi_\Gamma(w)$ denotes the projection of $w$ onto alphabet symbols from $\Gamma$.

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;
- $\Sigma$ 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

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

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

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

The reachability set $\mathcal{R}(G)$ and the language $\mathcal{L}(G)$ of $G$ are then defined as $\mathcal{R}(G) \overset{\text{def}}{=} \mathcal{R}(G, S)$ and $\mathcal{L}(G) \overset{\text{def}}{=} \mathcal{L}(G, S)$. 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 $\Pi_2^{\mathsf{P}}$-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 $\pi_{[i,j]}(\boldsymbol{u}) \overset{\text{def}}{=} (u_i, \ldots, u_j)$. By $\|\boldsymbol{u}\|_\infty$ we denote the maximum norm of $\boldsymbol{u}$, i.e., $\|\boldsymbol{u}\|_\infty \overset{\text{def}}{=} \max\{|u_i| : 1 \leq i \leq n\}$. 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 $k \cdot M \overset{\text{def}}{=} \{k \cdot \boldsymbol{m} : \boldsymbol{m} \in M\}$. Moreover, $\|M\|_\infty \overset{\text{def}}{=} \max\{\|\boldsymbol{z}\|_\infty : \boldsymbol{z} \in M\}$. The size $\#\boldsymbol{u}$ of $\boldsymbol{u}$ is $\#\boldsymbol{u} \overset{\text{def}}{=} \sum_{1 \leq i \leq m} \lceil \log |u_i| \rceil$, i.e., numbers are encoded in binary, and the size of $M$ is $\#M \overset{\text{def}}{=} \sum_{\boldsymbol{u} \in M} \#\boldsymbol{u}$. For an $m \times n$ matrix $A$ consisting of elements $a_{ij} \in \mathbb{Z}$, $\|A\|_{1,\infty} \overset{\text{def}}{=} \max\{\sum_{1 \leq j \leq n} |a_{ij}| : 1 \leq i \leq m\}$.

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 $\phi$ of Presburger arithmetic with no free variables, validity is to decide whether $\phi$ holds with respect to the standard interpretation in arithmetic. By $\|\phi\|_\infty$ we denote the largest constant occurring in $\phi$, and $|\phi|$ is the length of $\phi$, i.e., the number of symbols required to write down $\phi$, where constants are represented in unary. In analogy to matrices, we define $\|\phi\|_{1,\infty} \overset{\text{def}}{=} \|\phi\|_\infty \cdot |\phi|$. 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 $\psi$ by replacing every $x_i$ in $\psi$ by $x_i^*$. Finally, given a quantifier-free Presburger formula $\psi$ 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 $\psi$ by replacing every $t_i$ with $b_i$.

In this paper, we are in particular interested in the $\Pi_2$-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.

▶ **Proposition 3** ([7, 8])**.** *Validity in the $\Pi_2$-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 $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ be a finite subset of $\mathbb{N}^m$, define

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

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

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

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 $D : A \cdot \boldsymbol{x} \geq \boldsymbol{c}$, 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 proposition is a consequence of Corollary 1 in [24] and Theorem 5 in [1]. A formal proof can be found in Appendix A.

▶ **Proposition 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 $\Pi_2$-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 $\psi$ is a positive Boolean combination of atomic formulas $t_1, \ldots, t_k$. For our reduction, we write atomic formulas of $\psi$ 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 $a_{i,j} \overset{\text{def}}{=} a^+_{i,j} - a^-_{i,j}$, $b_{i,j} \overset{\text{def}}{=} b^+_{i,j} - b^-_{i,j}$ and $z_i \overset{\text{def}}{=} z^+_i - z^-_i$.

▶ **Example 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 $\phi$, 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

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

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 $\Sigma$ such that $\mathcal{L}(G) \subseteq \mathcal{L}(H)$ iff $\phi$ 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 $\psi$ 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 6. 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:

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

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

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

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

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, whose proof can be found in Appendix B.1, captures the essential properties of $G$.

▶ **Lemma 7.** *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 $1 \le i \le k$,*

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

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

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

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

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

We now turn towards the construction of $H \overset{\text{def}}{=} (N_H, \Sigma, S_H, P_H)$ 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 $Y$, $F_\psi$ 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

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

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

$$Y \to \epsilon$$

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 7 and can be shown along the same lines.

▶ **Lemma 8.** *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 $1 \le i \le k$, $w(t^+_i) = \sum_{1\le j\le n} b^+_{i,j} \cdot y_j$, $w(t^-_i) = \sum_{1\le j\le n} b^-_{i,j} \cdot y_j$, and*

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

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

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

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 $w(t^+_i) \ge w_Y(t^+_i)$ and $w(t^-_i) \ge w_Y(t^-_i)$ holds for every $1 \le i \le k$. 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 $\psi$ is valid.

In particular, the Boolean structure of $\psi$ enables us to produce arbitrary quantities of some $t_i$. This is the duty of the gadget $F_\psi$ which allows for assigning arbitrary values to some atomic formulas $t_i$ via gadgets $R_{t_i}$ defined below. The gadget $F_\psi$ recursively traverses the matrix formula $\psi$ and invokes some $R_\gamma$ whenever a disjunction is processed and a disjunct $\gamma$ 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 $R_\gamma$ for every subformula $\gamma$ of $\psi$ occurring in the syntax tree of $\psi$ is now not difficult: we traverse $\gamma$ until we reach a leaf $t_i$ of the syntax tree of $\gamma$ 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 9.** $\mathcal{L}(H, I) = \{(n_1^+, n_1^-, \ldots, n_k^+, n_k^-) \in \Sigma^\odot : n_j^+ \geq n_j^-,\ 1 \leq j \leq k\}$.

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

▶ **Lemma 10.** *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 7(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 8(ii) some suitable $\boldsymbol{y} \in \mathbb{N}^n$ can be obtained. All details can be found in Appendix B.2. ◀

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 Proposition 4 allows us to overcome this obstacle.

▶ **Lemma 11.** *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 7(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 Proposition 4, 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 $|P| \leq \binom{h+m+n}{h} \leq 2^{|\phi|}$ 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

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

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, in Appendix B.3 we 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 7, 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 8(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 $F_\psi$ 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 9 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. ◀

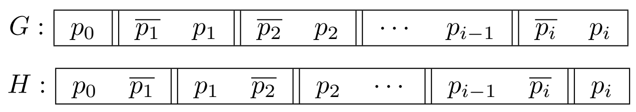


**Figure 1** Illustration of the pairing of alphabet symbols. In $G$, we require that in each cell $w(p_{j+1}) = 2 \cdot \overline{p_j}$, and in $H$ that $w(p_j) = w(\overline{p_j})$. Any word fulling these conditions has the property that $w(p_i) = 2^i \cdot w(p_0)$.

Lemmas 10 and 11 together with Proposition 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 12. In the appendix, we show that we 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 are provided in Appendix C.

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

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

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 $F_\psi$ of $H$ is also not regular. However, we can alternatively simulate $\psi$ by a regular grammar in which conjunction in $\psi$ 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]. Due to space constraints, all formal details are deferred to Appendix D.

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 13.** *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

| 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 | | |

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

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.

▶ **Theorem 14.** *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 $\phi$ is a $\Pi_2$-sentence of Presburger arithmetic, Proposition 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 12 yields the following theorem.

▶ **Theorem 15.** *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.

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.

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.

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.

10 Michel Hack. The equality problem for vector addition systems is undecidable. *Theoretical Computer Science*, 2(1):77–95, 1976.

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.

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.

14 Dung T. Huynh. The complexity of equivalence problems for commutative grammars. *Information and Control*, 66(1–2):103–121, 1985.

15 Dung T. Huynh. A simple proof for the $\Sigma_2^p$ 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.

17 Eryk Kopczyński. Complexity of problems of commutative grammars. *Logical Methods in Computer Science*, 11(1), 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.

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.

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.

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.

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.

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.

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

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.

## A Missing Proofs from Section 2

▶ **Proposition 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*

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

**Proof.** First, the bounds on the max-norm follow from Corollary 1 in [24]. It is shown there that the set of *integer* solutions to $D$ is bounded by

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

In order to restrict to non-negative solutions, we can add $n$ additional constraints, which yields the desired bounds on $\|B\|_\infty$ and $\|A\|_{1,\infty}$.

Second, the paper by Domenjoud [1] allows for deriving bounds on $|P|$ as follows. In his paper, he shows how to compute sets of minimal solutions $P'$ to homogeneous systems of *equations* $D' : A' \cdot \boldsymbol{x}' = 0$, where $A'$ is an $m' \times n'$ matrix of rank $r'$. Any solution $\boldsymbol{x}'$ to $D'$ can be obtained as a non-negative linear combination of elements of the set of minimal solutions $P'$. In particular, Theorem 5 in [1] allows for deriving that there exists bijection between elements in $P'$ and subsets of the set of columns of $A'$ of rank $r'$. Hence, $|P'| \leq \binom{n'}{r'}$.

In order to derive the desired bounds on $P$, we proceed as follows. Let $A' \stackrel{\text{def}}{=} (A \mid -I)$, where $I$ is the $m \times m$ unit matrix. Now $P$ can be obtained from the set of minimial solutions to the system of *equations* $D' : A' \cdot \boldsymbol{x}' = \boldsymbol{0}$ by projecting onto the first $n$ components. Note that $rank(A') = m$, hence $|P| \leq \binom{m+n}{m}$ as shown in [1]. ◀

## B Missing Proofs from Section 3

### B.1 Proof of Lemma 7

▶ **Lemma 7.** *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 $1 \leq i \leq k$,*

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

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

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

**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 $\boldsymbol{x} \overset{\text{def}}{=} (x_1, \ldots, x_m)$ 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). ◀

## B.2 Proof of Lemma 10

▶ **Lemma 10.** *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 7(i), there exists $w \in \mathcal{L}(G)$ such that for all $1 \leq i \leq k$,

$$w(t_i) = \sum_{1 \leq j \leq m} (a_{i,j}^+ - a_{i,j}^-) \cdot x_j + z_i^+ - z_i^-, \tag{10}$$

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

$$w_Y(t_i) = \sum_{1 \leq j \leq n} (b_{i,j}^+ - b_{i,j}^-) \cdot y_j. \tag{11}$$

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 $F_\psi$, 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 9, $w_I(t_i^+) - w_I(t_i^-) \geq 0$, and consequently $w(t_i) \geq w_Y(t_i)$ as required by (10) and (11). ◀

## B.3 Further Details to the Proof of Lemma 11

▶ **Lemma 11.** *Suppose $\phi = \forall \boldsymbol{x}.\exists \boldsymbol{y}.\psi(\boldsymbol{x}, \boldsymbol{y})$ is valid, then $\mathcal{L}(G) \subseteq \mathcal{L}(H)$.*

**Proof.** We deferred showing that

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

This can be derived as follows:

$$\begin{aligned}
\|\boldsymbol{y}^\dagger\|_\infty &\leq \|\boldsymbol{b}\|_\infty + \|\lambda_1 \cdot \boldsymbol{p}_1 + \cdots + \lambda_\ell \cdot \boldsymbol{p}_\ell\|_\infty \\
&\leq \|B\|_\infty + \sum_{1 \leq i \leq \ell} \lambda_i \cdot \|P\|_\infty \\
&\leq \|B\|_\infty + \|\boldsymbol{x}^*\|_\infty \cdot 2^{|\phi|} \cdot \|P\|_\infty && \text{(by (8))} \\
&\leq \left(1 + \|\boldsymbol{x}^*\|_\infty \cdot 2^{|\phi|}\right) \cdot (\|A\|_{1,\infty} + \|\boldsymbol{z}\|_\infty + 2)^{h+m+n} && \text{(by Prop. 4)} \\
&\leq \left(1 + \|\boldsymbol{x}^*\|_\infty \cdot 2^{|\phi|}\right) \cdot ((m+n+1) \cdot \|\phi\|_\infty + 2)^{k+m+n} \\
&\leq \left(1 + \|\boldsymbol{x}^*\|_\infty \cdot 2^{|\phi|}\right) \cdot |\phi|^{3 \cdot |\phi|} \\
&\leq (1 + \|\boldsymbol{x}^*\|_\infty) \cdot |\phi|^{3 \cdot |\phi|} \cdot 2^{|\phi|} && \text{(by (2))} \\
&\leq (1 + \|\boldsymbol{x}^*\|_\infty) \cdot \frac{c}{|\phi|^2} && \text{(by (3))}
\end{aligned}$$

◀

## B.4 Further Details to Remark 12

In order to show hardness of the equivalence problem for context-free commutative grammars, we merge $H$ into $G$, i.e., define

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

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

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

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

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

## C Missing Proofs from Section 3.1

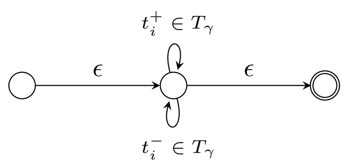
Here, we provide full details on how the reduction developed in Section 3 can be adapted in order to prove coNEXP-hardness of the equivalence problem for regular commutative grammars. As constructed, neither $G$ nor $H$ are regular. In this section, we show how to obtain regular commutative grammars $G^r$ 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 $F_\psi$ 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 $F_\psi$ 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 $\psi$. However, as we show now, it is possible to serialise $F_\psi$.

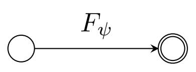
As a first step, we discuss the serialisation of $R_\gamma$ for all subformulas $\gamma$ occurring in the syntax-tree of $\psi$. Recall that the task of $R_\gamma$ is to generate arbitrary amounts of alphabet symbols $t_i^+$ and $t_i^-$ for all $t_i$ occurring in $\gamma$. We define the set $T_\gamma$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

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

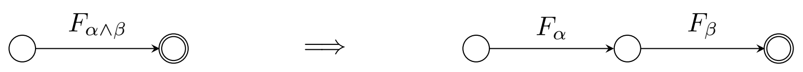
We can now redefine $R_\gamma$ to be the regular grammar corresponding to the following NFA, for which clearly $\mathcal{L}(R_\gamma) = T_\gamma^\odot$ holds:



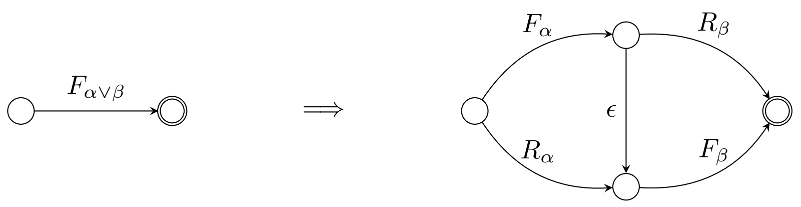
Next, we describe an inductive procedure that when completed yields an NFA that corresponds to a regular grammar whose language is equivalent to $\mathcal{L}(H, F_\psi)$. The procedure constructs in every iteration an NFA with a unique incoming and outgoing state and labels every transition with the 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:



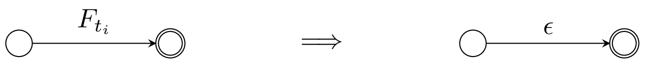
In the induction step, the rewriting of a transition labelled with $F_\gamma$ depends on the logical connective. A conjunction $\gamma = \alpha \wedge \beta$ is replaced by sequential composition:



Thus, the outgoing state of the gadget $F_\alpha$ connects to the incoming state of the gadget $F_\beta$. 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 $F_\alpha$ and $F_\beta$. If the inequalities appearing in $\alpha$ are allowed to receive arbitrary values, the transition labelled with $F_{\alpha\vee\beta}$ is replaced by the sequential composition of two gadgets, $R_\alpha$ and $F_\beta$; 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 $R_\gamma$ 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 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 6 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 $\Gamma_i \stackrel{\text{def}}{=} \{p_0, \overline{p_1}, p_1, \ldots, \overline{p_i}, p_i\}$ for every $i \in \mathbb{N}$. Before formally providing the construction in Lemma 16 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: $C_\ell \stackrel{\text{def}}{=} (\{S_\ell\}, \Gamma_i, S_\ell, P_\ell)$, where

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

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 $0 \le j < i$, 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 16.** *For every $i \in \mathbb{N}$, there are logarithmic-space computable regular commutative grammars $C_\ell$ 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 $1 \le j \le i\}$; and*
*(ii) $\mathcal{L}(C_r) = \{w \in \Gamma_i^{\odot} : w(p_j) \neq w(\overline{p_{j+1}})$ for some $0 \le j < i\}$.*
*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 $C_\ell$ as defined above has the desired properties. In order to prove Part (ii), we define $C_r \stackrel{\text{def}}{=} (N_r, \Gamma_i, S_r, P_r)$, where $N_r \stackrel{\text{def}}{=} \{S_r\} \cup \{N_j, \overline{N_{j+1}} : 0 \le j < i\}$

and

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

Informally speaking, after non-deterministically producing alphabet symbols $p_i$ starting from $S_r$, we can then non-deterministically choose an index $0 \le j < i$ 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 $g \ne j$ can be produced in arbitrary quantities. It is easily checked that $\mathcal{L}(C_r)$ has the desired properties. Now, we have

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

◀

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 16 and enables us to construct an exponential number of $t_i^+$ and $t_i^-$.

▶ **Corollary 17.** *For every $i \in \mathbb{N}$, there are logarithmic-space computable regular commutative grammars $C_\ell^\Sigma$ and $C_r^\Sigma$ 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 $C_\ell$ and $C_r$ are defined as in Lemma 16.*

Recall that $H$ already is a regular commutative grammar, and let $c$ be the constant from (3) and $j \stackrel{\text{def}}{=} \log c$. 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 (4), and let $C_\ell^\Sigma$ and $C_r^\Sigma$ be as defined in Corollary 17 for the alphabet $\Gamma_j$, the axiom of $G^r$ is $S_G^r$ and the transitions of $G^r$ are given by

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

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.

## D Missing Proofs from Section 4

Here, we present the full details and show that language inclusion for exponent-sensitive commutative grammars is in co-2NEXP. Let $G$ and $H$ be exponent-sensitive commutative grammars, and let $s \overset{\text{def}}{=} \#G$, $t \overset{\text{def}}{=} \#H$ 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 [22, Lem. 5.4], which is stated in terms of generalised communication-free Petri nets, but as argued in the proof of [22, 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 [22, Lem. 5.4].

▶ **Proposition 18** ([22]). *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* 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 [14].

▶ **Proposition 19** ([14]). *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* $|Q_i| = \mathrm{rank}(Q)$.

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 19, we say that it is the Huynh representation of $M$.

▶ **Lemma 20.** *Let $M = L(\boldsymbol{b}, Q)$ be a linear set. The Huynh representation of $M$ can be computed* DTIME$(2^{\mathsf{poly}(\#M)})$.

**Proof.** Let $p$ be the polynomial from Proposition 19. 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 $\mathrm{rank}(Q)$, 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 21.** *The Huynh representation of $\mathcal{R}(G)$ can be computed in* DTIME$(2^{2^{\mathsf{poly}(s)}})$.

**Proof.** First, we apply Proposition 18 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 20, 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 DTIME$(2^{\mathsf{poly}(\#M_i)})$ = DTIME$(2^{2^{\mathsf{poly}(s)}})$. Thus, the overall procedure also runs in 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 22** ([15]). *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 23.** *Deciding $\mathcal{L}(G) \subseteq \mathcal{L}(H)$ is in* co-2NEXP.

**Proof.** We describe a co-2NEXP-algorithm. First, by combining Proposition 18 with Proposition 22, 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 $\mathsf{DTIME}(\mathsf{poly}(\#w))$. ◀