Theoretical Computer Science 735 (2018) 147–161

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Context-free commutative grammars with integer counters and resets

Dmitry Chistikov [a,1], Christoph Haase [b,*,2], Simon Halfon [b]

[a] Max Planck Institute for Software Systems (MPI-SWS), Germany
[b] LSV, CNRS & ENS Cachan, Université Paris-Saclay, France

ARTICLE INFO

Article history:
Received 4 February 2016
Accepted 10 June 2016
Available online 27 June 2016

Keywords:
Context-free commutative grammars
Communication-free Petri nets
Reset nets
Vector addition systems with states
Presburger arithmetic
Subset sum

ABSTRACT

We study the computational complexity of reachability, coverability and inclusion for extensions of context-free commutative grammars with integer counters and reset operations on them. Those grammars can alternatively be viewed as an extension of communication-free Petri nets. Our main results are that reachability and coverability are inter-reducible and both NP-complete. In particular, this class of commutative grammars enjoys semi-linear reachability sets. We also show that the inclusion problem is, in general, coNEXP-complete and already Π_2^{P}-complete for grammars with only one non-terminal symbol. Showing the lower bound for the latter result requires us to develop a novel Π_2^{P}-complete variant of the classic subset sum problem.

1. Introduction

This paper studies the computational complexity of certain decision problems for extensions of context-free commutative grammars with integer counters and reset operations on them. The motivation for our work comes from the close relationship of such grammars with subclasses of Petri nets. For presentational purposes, we begin with introducing the decision problems we consider in terms of Petri nets.

Petri nets, or equivalently Vector Addition Systems with States (VASS), are a prominent and appealing class of infinite-state systems, from both theoretical and practical perspectives. On the one hand, their high level of abstraction allows them to be used as a mathematical model with well-defined semantics in a wide range of application domains, in particular but not limited to the verification of concurrent programs, see, e.g., [1]. On the other hand, for half a century Petri nets have provided a pool of challenging and intricate decision problems and questions about their structural properties. One of the most important and well-known instances is the question about the computational complexity of the reachability problem for Petri nets, which has attracted the attention of generations of researchers without, however, having been fully resolved.

A Petri net comprises a finite set of places with a finite number of transitions. Places may contain a finite number of tokens, and a transition can consume tokens from places, provided sufficiently many are present, and then add a finite number of tokens to some places. In the VASS setting, places are referred to as counters and we will often use these terms interchangeably in this paper. A configuration of a Petri net is a marking of its places, which is just a function $\boldsymbol{m}\colon \text{Places} \to \mathbb{N}$

* Corresponding author.
E-mail address: haase@lsv.ens-cachan.fr (C. Haase).

[1] Sponsored in part by the ERC Synergy award ImPACT. Present address: Department of Computer Science, University of Oxford, UK.
[2] Supported by Labex Digicosme, Univ. Paris-Saclay, project VERICONISS.

http://dx.doi.org/10.1016/j.tcs.2016.06.017
0304-3975/

or, equivalently, a vector of natural numbers whose components are indexed by elements from Places. The most prominent decision problems for Petri nets are reachability, coverability and inclusion. Given configurations $\boldsymbol{m}$ and $\boldsymbol{n}$ of a Petri net $\mathcal{A}$, reachability is to decide whether there is a sequence of transitions of $\mathcal{A}$ whose effect transforms $\boldsymbol{m}$ into $\boldsymbol{n}$. Coverability asks whether there is a transition sequence from $\boldsymbol{m}$ to a configuration that is "above" $\boldsymbol{n}$, i.e., a path to some configuration $\boldsymbol{n}'$ such that $\boldsymbol{n}' \geq \boldsymbol{n}$, where $\geq$ is interpreted component-wise. Finally, given Petri nets $\mathcal{A}$ and $\mathcal{B}$ with the same set of places, inclusion asks whether the set of markings reachable in $\mathcal{A}$ is contained in the set of those reachable in $\mathcal{B}$. All of these problems have been extensively studied in the literature. One of the earliest results was obtained by Lipton, who showed that reachability and coverability are EXPSPACE-hard [2]. Subsequently, Rackoff established a matching upper bound for coverability [3], and Mayr showed that reachability is decidable [4]. This result was later refined [5,6] and shown in a different way in [7], and an actual complexity-theoretic upper bound, namely membership in $\mathbf{F}_{\omega^3}$, a level of the fast-growing hierarchy, was only recently established [8]. For inclusion, it is known that this problem is in general undecidable [9] and Ackermann ($\mathbf{F}_\omega$)-complete when restricting to Petri nets with a finite reachability set [10].

For some application domains, standard Petri nets are not sufficiently expressive. For instance, as discussed, e.g., in [11], the verification of concurrent finite-state shared-memory programs requires additional operations on places such as transfers, where the content of one place can be copied to another one. Another example is the validation of business processes, which requires reset operations on places, i.e., a special kind of transitions which assign the value zero to some place [12]. The computational price for these extensions is high: reachability in the presence of any such extension becomes undecidable [13,14], while the complexity of coverability increases significantly to Ackermann ($\mathbf{F}_\omega$)-completeness in the presence of resets [15].

One of the main sources of the high complexity of decision problems for Petri nets and their extensions is the restriction that the places contain a *non-negative* number of tokens. This restriction enables one to enforce an order in which transitions can be taken, which is at the heart of many hardness proofs. In this paper, we relax this restriction and study the computational complexity of decision problems for a subclass of Petri nets, where the nets have additional counters that range over the integers and can be reset and where transitions are also structurally restricted. One advantage of this class is the decidability and a much lower computational complexity of standard decision problems when compared to usual Petri nets with reset operations.

Our contribution.

The main focus of this paper is the computational complexity of reachability, coverability and inclusion for so-called communication-free Petri nets extended with integer counters and resets, and for subclasses thereof. A communication-free Petri net is a Petri net in which every transition can remove a token from at most one place. An important property of communication-free Petri nets is that their sets of reachable markings are semi-linear [16–18], meaning in particular that they are closed under all Boolean operations (this is not the case for general Petri nets [19]). Communication-free Petri nets are essentially equivalent[3] to context-free commutative grammars, or basic parallel processes, and have extensively been studied in the literature [17,18,20–25]. For technical convenience we adopt the view of communication-free Petri nets as context-free commutative grammars in the technical part of this paper.

As our first main result, we show that context-free commutative grammars can be extended by a finite number of integer counters, i.e., counters that range over the integers and can be reset by transitions, while retaining NP-completeness of reachability and coverability, as well as preserving semi-linearity of the reachability set. This is achieved by showing that the reachability set of our extended class can be defined by a formula in existential Presburger arithmetic of polynomial size. The characterization obtained in this way can then be used in order to show coNEXP-completeness of the inclusion problem by application of complexity bounds for Presburger arithmetic.

Our second main result is a more refined analysis of the complexity of the inclusion problem. We show that even in the structurally simplest case of context-free commutative grammars with integer counters and *without any* control structure, i.e., a singleton non-terminal alphabet, the inclusion problem is hard for the second level of the polynomial hierarchy and, in fact, Π_2^{P}-complete. In essence, this problem is equivalent to asking, given two integer matrices A, B and a vector $\boldsymbol{v} \in \mathbb{N}^d$, whether for all $\boldsymbol{x} \in \mathbb{N}^m$ there exists some $\boldsymbol{y} \in \mathbb{N}^n$ such that $A \cdot \boldsymbol{x} + B \cdot \boldsymbol{y} = \boldsymbol{v}$. We prove hardness of this problem by developing a new Π_2^{P}-complete variant of the classical Subset Sum problem, which we believe is a contribution of independent interest.

This paper is an extended version of our conference paper [26], which appeared in the proceedings of the 8th International Workshop on Reachability Problems (RP 2014) held in Oxford, UK, in September 2014. It extends the results from [26] by considering a more general model: context-free commutative grammars with integer counters and resets instead of integer vector addition systems with states and resets considered in [26]; we also provide full proofs. Moreover, in [26] we left as an open question the precise complexity of the aforementioned Π_2^{P}-complete inclusion problem, which we could only show to be NP-hard and in Π_2^{P}. This question is now resolved in this paper.

Related work.

Apart from the related work mentioned above, closely connected to the problems considered in this paper is the work by Kopczyński and To [23] and Kopczyński [24]. In their work, the complexity of various decision problems for context-free commutative grammars and subclasses thereof has been studied when the number of alphabet symbols (which roughly

[3] This will be made more precise in Section 2.4.

corresponds to the number of places in the Petri net representation) is fixed. In [24], alphabet symbols may, informally speaking, be erased and negative quantities of alphabet symbols are possible. This essentially corresponds to adding counters with integer values to context-free commutative grammars. A further generalization of communication-free Petri nets, recently studied by Mayr and Weihmann, are communication-free Petri nets with arbitrary edge multiplicities [27]. In this class, transitions may also only consume tokens from one place, but they may take an arbitrary number of them.

A powerful technical tool in this context that we employ in this paper is defining Parikh images of communication-free Petri nets in existential Presburger arithmetic. This approach has been directly or indirectly taken, for instance, in [22,28–32]. In particular, in this paper we generalize a technique of Verma et al. [30], which has also been done in [32] in order to show decidability and complexity results for pushdown systems equipped with reversal-bounded counters.

As discussed above, we achieve a lower complexity for standard decision problems in comparison to general Petri nets by relaxing counters to range over the integers. Another approach going into a similar direction is to allow counters to range over the positive reals, for instance in continuous Petri nets introduced in [33]. It has been shown in recent work by Fraca and Haddad [34] that the decision problems we consider in this paper become substantially easier for such continuous Petri nets, with reachability even being decidable in P.

Finally, constraining the sequences of production rules applicable in language generating devices is a classical topic in formal language theory and commonly studied in the setting of controlled grammars. In this context, valence grammars [35] and blind counter automata [36] are closely related to our work and have led to a large body of research, see e.g. [37–39] and the references therein. Valence grammars over the monoid $(\mathbb{Z}, +)$ are context-free grammars in which every production rule is tagged with an integer, and a word is generated by the grammar whenever the sum of all integers that tag the production rules in its derivation equals zero. The results of this paper allow one to obtain complexity-theoretic upper bounds for deciding emptiness in (a generalization of) valence grammars over this monoid.

2. Preliminaries

In this section, we provide basic definitions that we rely on in this paper. First, we introduce some general notation and standard definitions related to Presburger arithmetic and formal language theory. We then introduce the class of context-free commutative grammars that we study in this paper and recall some known results about this class from the literature.

2.1. General notation

In the following, $\mathbb{Z}$ and $\mathbb{N}$ are the sets of integers and natural numbers (non-negative integers), respectively, and $\mathbb{N}^d$ and $\mathbb{Z}^d$ are the set of dimension-d vectors in $\mathbb{N}$ and $\mathbb{Z}$, respectively. If not stated otherwise, all numbers in this paper are assumed to be encoded in binary. For $a, b \in \mathbb{Z}$ such that $a < b$, we denote by $[a, b]$ the set $\{a, a+1, \ldots, b\}$. As an abbreviation, $[d]$ denotes $[1, d]$. For $\boldsymbol{v} \in \mathbb{Z}^d$ we write $\boldsymbol{v}(i)$ for the i-th component of $\boldsymbol{v}$ for $i \in [d]$. Let $z \in \mathbb{Z}$, we denote by $\boldsymbol{z}$ the vector in any dimension which has value z in each of its components. Given two vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{Z}^d$, we write $\boldsymbol{v}_1 \geq \boldsymbol{v}_2$ if and only if for all $i \in [d]$, $\boldsymbol{v}_1(i) \geq \boldsymbol{v}_2(i)$. Given a vector $\boldsymbol{v} \in \mathbb{Z}^d$ and a set $R \subseteq [d]$, by $\boldsymbol{v}_{|R}$ we denote the vector which coincides with $\boldsymbol{v}$ except for components from R which are *reset* to zero, i.e.,

$$\boldsymbol{v}_{|R}(i) \stackrel{\text{def}}{=} \begin{cases} \boldsymbol{v}(i) & \text{if } i \notin R \\ 0 & \text{otherwise.} \end{cases}$$

We call $|R$ the *reset operator*.

2.2. Presburger arithmetic

The first-order theory of the structure $\langle \mathbb{N}, 0, 1, +, \geq \rangle$, i.e., quantified linear arithmetic over natural numbers, is commonly known as *Presburger arithmetic (PA)*. The size $|\Phi|$ of a PA-formula Φ is the number of symbols required to write it down. Two fragments of Presburger arithmetic with a fixed number of quantifier alternations are relevant to us in this paper.

Proposition 1. *The existential Σ_1-fragment of Presburger arithmetic is* NP*-complete [40]. Validity in the Π_2-fragment of PA, i.e. its restriction to a $\forall^*\exists^*$-quantifier prefix, is* coNEXP*-complete [41,42].*

Given a PA-formula $\Phi(x_1, \ldots, x_d)$ in d free variables, we define

$$[\![\Phi(x_1, \ldots, x_d)]\!] \stackrel{\text{def}}{=} \{(n_1, \ldots, n_d) \in \mathbb{N}^d : \Phi(n_1/x_1, \ldots, n_d/x_d) \text{ is valid}\}.$$

Here, $\Phi(n_1/x_1, \ldots, n_d/x_d)$ is obtained from Φ by replacing every x_i with n_i; we also write $\Phi(\boldsymbol{n}/\boldsymbol{x})$ as a shorthand for replacing the components of $\boldsymbol{x}$ with the respective components of $\boldsymbol{n}$ in Φ. For notational convenience, we sometimes use vectors of vectors of first-order variables and denote them by bold capital letters, e.g., $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$, where the $\boldsymbol{x}_i$ are vectors of first-order variables.

A set $M \subseteq \mathbb{N}^d$ is *PA-definable* if there exists a PA formula $\Phi(x_1, \ldots, x_d)$ such that $M = [\![\Phi(x_1, \ldots, x_d)]\!]$. Recall that a result due to Ginsburg & Spanier states that PA-definable sets coincide with *semi-linear sets* [43]. A subset of $M \subseteq \mathbb{N}^d$ is linear if there exist $\boldsymbol{b} \in \mathbb{N}^d$ and $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n\} \subseteq \mathbb{N}^d$ such that

$$M = L(\boldsymbol{b}, Q) \stackrel{\text{def}}{=} \boldsymbol{b} + \{\lambda_1 \cdot \boldsymbol{q}_1 + \cdots + \lambda_n \cdot \boldsymbol{q}_n : \lambda_i \in \mathbb{N}\};$$

semi-linear sets are finite unions of linear sets and are closed under all Boolean operations [43].

In this paper, we sometimes wish to define subsets of $\mathbb{Z}^d$ via formulas of Presburger arithmetic. Clearly, any integer z can be represented as the difference of two natural numbers x and y. Hence, the homomorphism $h : \mathbb{N}^{2d} \to \mathbb{Z}^d$ defined as

$$h : (x_1, y_1, \ldots, x_n, y_n) \mapsto (x_1 - y_1, \ldots, x_n - y_n)$$

can be lifted in order to uniquely assign a subset of $\mathbb{Z}^d$ to every subset of $\mathbb{N}^{2d}$. Thus, whenever it is convenient for us, we may with no loss of generality interpret some open variables of formulas of Presburger arithmetic in the integers (we will explicitly mention such cases).

2.3. Formal languages

Let $\Sigma = \{a_1, \ldots, a_m\}$ be a finite alphabet. The free monoid generated by Σ is denoted by Σ^*, and by $\Sigma^\odot$ we denote the free commutative monoid generated by Σ. Elements of Σ^* are words, i.e., finite sequences of elements from Σ, with the usual concatenation operation $\cdot$. Elements of $\Sigma^\odot$ are commutative words; we treat them as mappings of the form $\Sigma \to \mathbb{N}$, or, equivalently, as vectors from $\mathbb{N}^m$ with component-wise addition. The empty word is denoted by ε. Given $w \in \Sigma^* \cup \Sigma^\odot$ and $a \in \Sigma$, $|w|_a$ denotes the number of times a occurs in the (usual or commutative) word w. We interchangeably use different equivalent ways in order to represent a word $w \in \Sigma^\odot$. For $j \in [m]$ let $i_j = |w|_{a_j}$; 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 $w(a_j) = i_j$, whichever is most convenient. Given $v, w \in \Sigma^\odot$, we write $v + w$ to denote the sum of v and w. Given $w \in \Sigma^*$, we denote by $\pi(w) \in \Sigma^\odot$ its Parikh image, i.e., $\pi(w) \stackrel{\text{def}}{=} (|w|_{a_1}, \ldots, |w|_{a_m})$.

Viewing commutative words as elements of $\mathbb{N}^m$ allows us to employ them inside formulas of Presburger arithmetic. In particular, given a vector $\boldsymbol{x} = (x_1, \ldots, x_m)$ of first-order variables and a commutative word $w = (i_1, \ldots, i_m) \in \Sigma^\odot$, then $\boldsymbol{x} = w$ abbreviates $\bigwedge_{1 \le j \le m} x_j = i_j$.

2.4. Context-free commutative grammars with integer counters and resets

The main objects studied in this paper are derived from a general class of context-free commutative grammars equipped with integer counters[4] which can be reset, incremented or decremented when production rules are applied. Formally, these grammars are defined as follows.

Definition 2. A *context-free commutative grammar with integer counters and resets ($\mathbb{Z}$-CFCG$_R$)* is a quadruple $\mathcal{G} = (N, C, P, S)$ where

- N is a finite alphabet of *non-terminal symbols*;
- C is a finite set of *counters*;
- $P \subseteq N \times 2^C \times \mathbb{Z}^C \times N^\odot$ is a finite set of *production rules*; and
- $S \in N$ is the *axiom*.

We often write $p \in P$ as a tuple of elements indexed by p, i.e., as $p = (a_p, R_p, \boldsymbol{z}_p, w_p)$. Informally, the production p can be applied whenever the non-terminal a_p is available; it then resets the counters specified by R_p and adds $\boldsymbol{z}_p$ to all counters while producing non-terminal symbols w_p. Formally, let $C(\mathcal{G}) \stackrel{\text{def}}{=} N^\odot \times \mathbb{Z}^C$ be the *set of configurations of* $\mathcal{G}$. Given configurations $(s, \boldsymbol{u}), (t, \boldsymbol{v}) \in C(\mathcal{G})$ and $p \in P$, we write $(s, \boldsymbol{u}) \xrightarrow{p}_{\mathcal{G}} (t, \boldsymbol{v})$ if there is some $w \in N^\odot$ such that

- $s = w + a_p$,
- $t = w + w_p$; and
- $\boldsymbol{v} = \boldsymbol{u}_{|R} + \boldsymbol{z}_p$.

We write $(s, \boldsymbol{u}) \to_{\mathcal{G}} (t, \boldsymbol{v})$ whenever $(s, \boldsymbol{u}) \xrightarrow{p}_{\mathcal{G}} (t, \boldsymbol{v})$ for some $p \in P$.

A *run* is a word $\gamma = p_1 \cdots p_n \in P^*$ such that there exists a finite sequence of configurations $\varrho : c_0 c_1 \cdots c_n$ such that $c_i \xrightarrow{p_{i+1}}_{\mathcal{G}} c_{i+1}$ for all $0 \le i < n$, and we write $c_0 \xrightarrow{\gamma}_{\mathcal{G}} c_n$ in this case. Furthermore, we write $c \to^*_{\mathcal{G}} c'$ if there is a run $\gamma \in P^*$

[4] In the literature, such counters are often also called *blind counters*.

such that $c \xrightarrow{\gamma}_{\mathcal{G}} c'$. We drop the subscript $\mathcal{G}$ if it is clear from the context. Given $\boldsymbol{u} \in \mathbb{Z}^C$, the *reachability set starting from* $\boldsymbol{u}$ is defined as

$$\mathit{reach}(\mathcal{G}, \boldsymbol{u}) = \{\boldsymbol{v} \in \mathbb{Z}^C : (S, \boldsymbol{u}) \to^*_{\mathcal{G}} (t, \boldsymbol{v}) \text{ for some } t \in N^{\odot}\}.$$

Remark 3. Context-free (commutative) grammars are commonly used as language acceptors or generators. In our setting, when restricting counter updates to $\mathbb{N}$ (i.e., when $P \subseteq N \times 2^C \times \mathbb{N}^C \times N^{\odot}$), we may view $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ as generators of languages over $C^{\odot}$.

In this paper, we study the computational complexity of deciding reachability, coverability and inclusion in $\mathbb{Z}$-CFCG$_{\mathrm{R}}$.

$\mathbb{Z}$-CFCG$_{\mathrm{R}}$ Reachability/Coverability/Inclusion

INPUT: $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ $\mathcal{G}$, $\mathcal{H}$ over the same set of counters C and configurations $(s, \boldsymbol{u}), (t, \boldsymbol{v}) \in C(\mathcal{G})$, $\boldsymbol{v}, \boldsymbol{v}' \in \mathbb{Z}^C$.
QUESTION: *Reachability:* Is there a run $(s, \boldsymbol{u}) \to^*_{\mathcal{G}} (t, \boldsymbol{v})$?
Coverability: Is there a $\boldsymbol{z} \in \mathbb{Z}^C$ such that $(s, \boldsymbol{u}) \to^*_{\mathcal{G}} (t, \boldsymbol{z})$ and $\boldsymbol{z} \geq \boldsymbol{v}$?
Inclusion: Does $\mathit{reach}(\mathcal{G}, \boldsymbol{u}) \subseteq \mathit{reach}(\mathcal{H}, \boldsymbol{v})$ hold?

We also study and discuss natural subclasses of $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ where we restrict the use of reset operations or the set of productions of the grammar. A $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ $\mathcal{G} = (N, C, P, S)$ is an

- *integer vector addition system with states ($\mathbb{Z}$-VASS)* if $P \subseteq N \times \{\emptyset\} \times \mathbb{Z}^C \times (N \cup \{\varepsilon\})$;
- *integer vector addition system ($\mathbb{Z}$-VAS)* if $\mathcal{G}$ is a $\mathbb{Z}$-VASS and $N = \{S\}$.

$\mathbb{Z}$-VASS are obtained from $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ by restricting the grammar to be left-linear and by disallowing resets. We use them in order to obtain stronger lower bounds. Left-linear context-free grammars are known to recognize regular languages, and equivalently, $\mathbb{Z}$-VASS can be seen as finite-state automata equipped with integer counters. Formalized in this manner, it is easier to see that classical vector addition systems with states (VASS) can be recovered from the definition of $\mathbb{Z}$-VASS by restricting the set of configurations to $(N \cup \{\varepsilon\}) \times \mathbb{N}^C$ and adjusting the definition of $\to_{\mathcal{G}}$ appropriately. This justifies the term "$\mathbb{Z}$-VASS". Note that in $\mathbb{Z}$-VASS, we restrict commutative words in configurations to length at most one, and we restrict the reachability problem accordingly.

Finally, note that a $\mathbb{Z}$-VAS $\mathcal{A} = (\{S\}, C, P, S)$ can simply be represented by a matrix $A \in \mathbb{Z}^{d \times k}$ where $d = |C|$ and $k = |P|$ and A is the matrix whose columns are $\boldsymbol{z}_p$ for $p \in P$. The matrix A has the property that for all $\gamma \in P^*$, $(S, \boldsymbol{u}) \xrightarrow{\gamma} (S, \boldsymbol{u} + A \cdot \pi(\gamma))$. Consequently, reachability in $\mathbb{Z}$-VAS and all classes subsuming $\mathbb{Z}$-VAS is NP-hard, which can be shown by a reduction from the feasibility problem of a system of linear Diophantine equations $A \cdot \boldsymbol{x} = \boldsymbol{u}, \boldsymbol{x} \geq \boldsymbol{0}$. This problem is known to be NP-hard even when numbers are encoded in unary [44]. We have that $A \cdot \boldsymbol{x} = \boldsymbol{u}, \boldsymbol{x} \geq \boldsymbol{0}$ is valid if and only if $(S, -\boldsymbol{u}) \to^*_{\mathcal{A}} (S, \boldsymbol{0})$ in the corresponding $\mathbb{Z}$-VAS.

Relationship to communication-free Petri nets.

As already stated in the introduction, context-free commutative grammars are closely related to communication-free Petri nets. For inter-reducibility results, see, e.g., [18]. For the sake of completeness, here we briefly state the relationship on an informal level.

Viewed in our framework, context-free commutative grammars are $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ whose integer counters can only be incremented and thus correspond to *terminal symbols*. Context-free commutative grammars correspond to communication-free Petri nets by viewing the set of non-terminal and terminal symbols as the set of places of the Petri net. Similarly, $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ correspond to communication-free Petri nets which are additionally equipped with special places that can take a possibly negative number of tokens and with special arcs that can set the number of tokens on those counters to zero. All upper bounds for context-free commutative grammars carry over to communication-free Petri nets, and all lower bounds for communication-free Petri nets carry over to context-free commutative grammars, see for example [18].

3. Reachability and coverability in $\mathbb{Z}$-CFCG$_{\mathrm{R}}$

In this section, we consider the reachability and coverability problem for $\mathbb{Z}$-CFCG$_{\mathrm{R}}$. We begin by showing that reachability and coverability are logarithmic-space interreducible; such a reduction is not known for general Petri nets and cannot exist for Petri nets equipped with reset operations since reachability in such nets is undecidable whereas coverability is decidable [13]. Subsequently, we show that reachability and hence coverability in $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ is NP-complete by showing that the reachability relation is definable by a sentence in existential Presburger arithmetic of polynomial size.

3.1. Reachability and coverability are interreducible

Here, we show that reachability and coverability are logarithmic-space interreducible in $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ and all of the subclasses we introduced in Section 2.4. Thanks to this observation, all lower and upper bounds for reachability carry over to coverability, and *vice versa*.

Theorem 4. *Reachability and coverability are logarithmic-space interreducible in each of the classes $\mathbb{Z}$-CFCG$_R$, $\mathbb{Z}$-VASS and $\mathbb{Z}$-VAS. The reduction doubles the number of counters.*

Proof. We first show how to reduce coverability to reachability. We adapt the folklore construction used for reducing coverability in VASS to reachability in VASS. This reduction adds extra transitions in order to make the VASS *lossy*, i.e. transitions that allow counters to be non-deterministically decremented at any time. To this end, let $\mathcal{G} = (N, C, P, S)$ be a $\mathbb{Z}$-CFCG$_R$ and $(s, \boldsymbol{u}), (t, \boldsymbol{v})$ two configurations of $\mathcal{G}$. Define $\mathcal{H} = (N, C, P', S)$ where $P' = P \cup \{(S, \emptyset, -\boldsymbol{c}, S) : c \in C\}$. Here $\boldsymbol{c}$ is the vector $\boldsymbol{c} : C \to \mathbb{N}$ such that $\boldsymbol{c}(c) = 1$ and $\boldsymbol{c}(c') = 0$ for all $c' \neq c$. It is easily seen that $(t, \boldsymbol{v})$ can be covered in $\mathcal{G}$ starting at $(s, \boldsymbol{u})$ if and only if there is a run $(s + S, \boldsymbol{u}) \to^*_{\mathcal{H}} (t + S, \boldsymbol{v})$ in $\mathcal{H}$ due to the monotonicity of coverability.

We now show how to reduce reachability to coverability. Let $\mathcal{G} = (N, C, P, S)$ be a $\mathbb{Z}$-CFCG$_R$ and let $(s, \boldsymbol{u}), (t, \boldsymbol{v}) \in C(\mathcal{G})$. We construct a $\mathbb{Z}$-CFCG$_R$ $\mathcal{H} = (N, C \uplus \tilde{C}, P', S)$ where $\tilde{C} \stackrel{\text{def}}{=} \{\tilde{c} : c \in C\}$ consists of an additional disjoint copy of C and P' contains a production of the form $(a, R \cup \tilde{R}, (\boldsymbol{z}, -\boldsymbol{z}), w)$ whenever $(a, R, \boldsymbol{z}, w)$ is a production of P. The $\mathbb{Z}$-CFCG$_R$ $\mathcal{H}$ therefore has the two following properties:

- starting from a configuration $(s, (\boldsymbol{u}, -\boldsymbol{u}))$, any configuration reached is of the form $(t, (\boldsymbol{v}, -\boldsymbol{v}))$; and
- $(s, \boldsymbol{u}) \to^*_{\mathcal{G}} (t, \boldsymbol{v})$ iff $(s, (\boldsymbol{u}, -\boldsymbol{u})) \to^*_{\mathcal{H}} (t, (\boldsymbol{v}, -\boldsymbol{v}))$.

These properties are easily shown by induction on the length of the run. Consequently, the configuration $(t, \boldsymbol{v}, -\boldsymbol{v}))$ can be covered starting at $(s, (\boldsymbol{u}, -\boldsymbol{u}))$ in $\mathcal{H}$ if and only if there exists some $\boldsymbol{z}$ such that $(s, (\boldsymbol{u}, -\boldsymbol{u})) \to^*_{\mathcal{H}} (t, (\boldsymbol{z}, -\boldsymbol{z}))$ and $\boldsymbol{z} \geq \boldsymbol{v}$ and $-\boldsymbol{z} \geq -\boldsymbol{v}$; i.e. if and only if the configuration $(t, (\boldsymbol{v}, -\boldsymbol{v}))$ is actually reached in $\mathcal{H}$. Consequently this is equivalent to $(t, \boldsymbol{v})$ being reachable from $(s, \boldsymbol{u})$ in $\mathcal{G}$.

Finally, observe that all of the reductions described above preserve the restrictions imposed on the subclasses of $\mathbb{Z}$-CFCG$_R$ and are thus also valid for $\mathbb{Z}$-VASS and $\mathbb{Z}$-VAS. □

3.2. Reachability and coverability in $\mathbb{Z}$-CFCG$_R$ are NP*-complete*

As discussed in Section 2.4, reachability is already NP-hard for $\mathbb{Z}$-VAS. In this section, we establish a matching upper bound for $\mathbb{Z}$-CFCG$_R$. One main idea for showing the upper bound is that since there are no constraints on the values of the integer counters along a run, a reset on a particular counter allows to forget any information about the value of this counter up to this point, i.e., a reset cuts the run. Hence, in order to determine the value of a particular counter at the end of a run, we only need to sum up the effects of the operations on this counter since the last occurrence of a reset on this counter. Moreover, since addition and subtraction are commutative, the order in which these effects occur is irrelevant. That is, to determine whether a certain configuration on integer counters is reached by a run, it suffices to consider the Parikh image of this run.

Subsequently, we introduce a generalization of the notion of the Parikh image of a run that, in effect, enables us to access the last occurrence of a reset on a counter. This can be achieved by recording the last occurrence of each production in $P = \{p_1, \ldots, p_k\}$, some of which may not reset any counter at all. This idea leads to the following unique decomposition of any run $\gamma \in P^+$ into *partial runs* $\gamma_1, \ldots, \gamma_{i_\ell}$ as

$$\gamma = \gamma_1 p_{i_1} \gamma_2 p_{i_2} \cdots \gamma_l p_{i_\ell}$$

for some $\ell \leq k$ such that all i_j are pairwise distinct and for all $j \in [\ell]$, $\gamma_j \in \{p_{i_j}, \ldots, p_{i_\ell}\}^*$. This decomposition simply keeps track of the last occurrence of each production used in γ. For instance for $P = \{a, b, c, d, e\}$, the word $\gamma = aaebaeabba$ can uniquely be decomposed as $(aaeba)e(ab)b()a$. This decomposition is formalized in the following definition as the *generalized Parikh image* of a word. By $\mathfrak{S}_k$ we denote the permutation group on k symbols, and we sometimes treat its elements as vectors of $\mathbb{N}^k$.

Definition 5. Let $\mathcal{G} = (N, C, P, S)$ be a $\mathbb{Z}$-CFCG$_R$ with $P = \{p_1, \ldots, p_k\}$. A triple $(\boldsymbol{A}, \sigma, m) = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_k, \sigma, m) \in (\mathbb{N}^k)^k \times \mathfrak{S}_k \times [k]$ is *a generalized Parikh image of* $\gamma \in P^+$ if there exists a decomposition

$$\gamma = \gamma_m p_{\sigma(m)} \gamma_{m+1} p_{\sigma(m+1)} \cdots \gamma_k p_{\sigma(k)}$$

such that

(i) for all $m \leq i \leq k$, $\gamma_i \in \{p_{\sigma(i)}, \ldots, p_{\sigma(k)}\}^*$; and
(ii) for all $1 \leq i < m$, $\boldsymbol{\alpha}_i = \boldsymbol{0}$, and for all $m \leq i \leq k$, $\boldsymbol{\alpha}_i = \pi(\gamma_i)$.

We denote by $\Pi(\gamma)$ the set of all generalized Parikh images of a word $\gamma \in P^+$.

This definition formalizes the intuition, combining the decomposition described above with some padding by dummy vectors for productions that do not occur in γ, in order to obtain canonical objects of *uniform size*. Even though generalized Parikh images are not unique, two generalized Parikh images of the same word differ only in the order of productions that

do not appear in γ. For instance, if $P = \{a, b, c, d, e\}$, the word $\gamma = aaebaeabba$ has two generalized Parikh images: they agree on $\boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_2 = (0, 0, 0, 0, 0)$, $\boldsymbol{\alpha}_3 = (3, 1, 0, 0, 1)$, $\boldsymbol{\alpha}_4 = (1, 1, 0, 0, 0)$, $\boldsymbol{\alpha}_5 = (0, 0, 0, 0, 0)$ and $\sigma(3) = 5$, $\sigma(4) = 2$, $\sigma(5) = 1$, and $m = 3$, and only differ on $\sigma(1)$ and $\sigma(2)$ that can be 3 and 4, or 4 and 3, respectively.

Generalized Parikh images can now be applied to reachability in $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ as follows: the counter values at the end of a run $\gamma \in P^+$, starting from an initial configuration $(s, \boldsymbol{u})$, are fully determined by a generalized Parikh image of γ, as shown in the next lemma. Recall that, if $P = \{p_1, \ldots, p_k\}$, then each production p_ℓ resets the counters in the set $R_{p_\ell} \subseteq C$. Subsequently, for $i \in [1, k]$ we write

$$R_i = R_{p_{\sigma(i)}} \cup \cdots \cup R_{p_{\sigma(k)}};$$

note that R_i depends on the set of productions P and on the permutation σ. Also, R_{k+1} will denote the empty set.

Lemma 6. *Let $\mathcal{G} = (N, C, P, S)$ be a $\mathbb{Z}$-CFCG$_R$ with $P = \{p_1, \ldots, p_k\}$, $(s, \boldsymbol{u})$ and $(t, \boldsymbol{v})$ two configurations of $C(\mathcal{G})$ and $\gamma \in P^+$ such that $(s, \boldsymbol{u}) \xrightarrow{\gamma} (t, \boldsymbol{v})$. Moreover, let $(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k, \sigma, m) \in \Pi(\gamma)$ be a generalized Parikh image of γ. Then the following holds:*

$$\boldsymbol{v} = \boldsymbol{u}_{|R_m} + \sum_{i=m}^{k} \left[\left(\sum_{p \in P} \boldsymbol{\alpha}_i(p) \cdot \boldsymbol{z}_p \right)_{|R_i} + \left(\boldsymbol{z}_{p_{\sigma(i)}}\right)_{|R_{i+1}} \right].$$

Proof. The proof of the lemma formalizes the intuition given in the introduction of this section: in order to determine the final counter values at the end of the run, it is sufficient to only consider the effects after the last reset has occurred on a particular counter.

Formally, let $\gamma = \gamma_m p_{\sigma(m)} \gamma_{m+1} \cdots \gamma_k p_{\sigma(k)}$ be the decomposition associated to the generalized Parikh image $(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k, \sigma, m)$ of γ. Moreover, let $(s_m, \boldsymbol{u}_m), \ldots, (s_k, \boldsymbol{v}_k)$ and $(t_m, \boldsymbol{v}_m), \ldots, (t_k, \boldsymbol{v}_k)$ be the configurations such that for any $i \in [m, k]$,

$$(s_i, \boldsymbol{u}_i) \xrightarrow{\gamma_i}_{\mathcal{G}} (t_i, \boldsymbol{v}_i) \xrightarrow{p_{\sigma(i)}}_{\mathcal{G}} (s_{i+1}, \boldsymbol{u}_{i+1}),$$

where $(s_m, \boldsymbol{u}_m) = (s, \boldsymbol{u})$ and $(s_{k+1}, \boldsymbol{u}_{k+1}) = (t, \boldsymbol{v})$.

We prove the following statement by induction on $j \in [m, k]$:

$$\boldsymbol{u}_{j\,|R_j} = \boldsymbol{u}_{|R_m} + \sum_{i=m}^{j-1} \left[\left(\sum_{p \in P} \boldsymbol{\alpha}_i(p) \cdot \boldsymbol{z}_p \right)_{|R_i} + \left(\boldsymbol{z}_{p_{\sigma(i)}}\right)_{|R_{i+1}} \right] \tag{1}$$

$$\boldsymbol{v}_{j\,|R_j} = \boldsymbol{u}_{|R_m} + \sum_{i=m}^{j-1} \left[\left(\sum_{p \in P} \boldsymbol{\alpha}_i(p) \cdot \boldsymbol{z}_p \right)_{|R_i} + \left(\boldsymbol{z}_{p_{\sigma(i)}}\right)_{|R_{i+1}} \right] + \left(\sum_{p \in P} \boldsymbol{\alpha}_j(p) \cdot \boldsymbol{z}_p \right)_{|R_j}. \tag{2}$$

Base case $j = m$: Equation (1) is obvious. Since only resets on components $c \in R_m$ occur in γ_m by definition of the decomposition, and since addition is commutative and associative, only the number of times each production appears is important. Hence

$$\boldsymbol{v}_{m|R_m} - \boldsymbol{u}_{m|R_m} = \left(\sum_{p \in P} |\gamma_m|_p \cdot \boldsymbol{z}_p \right)_{|R_m} = \left(\sum_{p \in P} \boldsymbol{\alpha}_m(p) \cdot \boldsymbol{z}_p \right)_{|R_m}.$$

Induction step $j > m$: The configuration $(s_j, \boldsymbol{u}_j)$ is obtained from the configuration $(t_{j-1}, \boldsymbol{v}_{j-1})$ using the production $p_{\sigma(j-1)}$, therefore

$$\boldsymbol{u}_j = (\boldsymbol{v}_{j-1})_{|R_{p_{\sigma(j-1)}}} + \boldsymbol{z}_{p_{\sigma(j-1)}},$$

which leads to

$$\begin{aligned}
\boldsymbol{u}_{j\,|R_j} &= ((\boldsymbol{v}_{j-1})_{|R_{p_{\sigma(j-1)}}} + \boldsymbol{z}_{p_{\sigma(j-1)}})_{|R_j} \\
&= (\boldsymbol{v}_{j-1})_{|R_{p_{\sigma(j-1)}} \cup R_j} + (\boldsymbol{z}_{p_{\sigma(j-1)}})_{|R_j} \\
&= (\boldsymbol{v}_{j-1})_{|R_{j-1}} + (\boldsymbol{z}_{p_{\sigma(j-1)}})_{|R_j} \\
&= \boldsymbol{u}_{|R_m} + \sum_{i=m}^{j-2} \left[\left(\sum_{p \in P} \boldsymbol{\alpha}_i(p) \cdot \boldsymbol{z}_p \right)_{|R_i} + \left(\boldsymbol{z}_{p_{\sigma(i)}}\right)_{|R_{i+1}} \right] + \left(\sum_{p \in P} \boldsymbol{\alpha}_{j-1}(p) \cdot \boldsymbol{z}_p \right)_{|R_{j-1}} + \boldsymbol{z}_{p_{\sigma(j-1)}|R_j} \\
&= \boldsymbol{u}_{|R_m} + \sum_{i=m}^{j-1} \left[\left(\sum_{p \in P} \boldsymbol{\alpha}_i(p) \cdot \boldsymbol{z}_p \right)_{|R_i} + \left(\boldsymbol{z}_{p_{\sigma(i)}}\right)_{|R_{i+1}} \right].
\end{aligned}$$

In a similar way, the configuration $(t_j, \boldsymbol{v}_j)$ is obtained from the configuration $(s_j, \boldsymbol{u}_j)$ by applying the partial run γ_j, which only resets counters in R_j. Therefore,

$$\boldsymbol{v}_{j|R_j} = \left(\boldsymbol{u}_j + \sum_{p\in P} \boldsymbol{\alpha}_j(p)\cdot \boldsymbol{z}_p\right)_{|R_j}.$$

The statement of the lemma now follows from taking $j = k+1$ in Equation (1). □

Thus, in order to decide reachability in $\mathbb{Z}$-CFCG$_{\mathrm{R}}$, it suffices to find a suitable way to reason about generalized Parikh images. In [30], Verma et al. show how to construct in polynomial time an existential Presburger formula representing the Parikh image of the language of a context-free grammar. We generalize this construction to generalized Parikh images of $\mathbb{Z}$-CFCG$_{\mathrm{R}}$. First, let us state the result from [30] using the terminology of this paper.

Proposition 7. *[30, Thm. 4] Given a $\mathbb{Z}$-CFCG$_R$ $\mathcal{G} = (N, C, P, S)$ with $|N| = n$ and $|P| = k$, one can compute in polynomial time an existential Presburger formula $\varphi_{\mathcal{G}}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\alpha})$ where $\boldsymbol{s}, \boldsymbol{t}$ are n-tuples and $\boldsymbol{\alpha}$ is a k-tuple of first-order variables such that for all s, t in $N^{\circledcirc}$ and $\boldsymbol{\alpha} \in \mathbb{N}^k$, the following are equivalent:*

- $(s, t, \boldsymbol{\alpha}) \in [\![\varphi_{\mathcal{G}}]\!]$
- *there is a run $\gamma \in P^*$ with $\pi(\gamma) = \boldsymbol{\alpha}$ such that for any $\boldsymbol{u} \in \mathbb{Z}^C$, $(s, \boldsymbol{u}) \xrightarrow{\gamma}_{\mathcal{G}} (t, \boldsymbol{v})$ for some $\boldsymbol{v} \in \mathbb{Z}^C$.*

In other words, a model $(s, t, \boldsymbol{\alpha})$ of the formula $\varphi_{\mathcal{G}}$ asserts that $\boldsymbol{\alpha}$ is the Parikh image of a valid run between the commutative words s and t. To implement the definition of generalized Parikh image in Presburger arithmetic, it is now sufficient to guess the intermediate words and "connect" them using formulas $\varphi_{\mathcal{G}}$. Subsequently, whenever we define a permutation σ in Presburger arithmetic, we write $\boldsymbol{\sigma}$ for the corresponding vector of first-order variables defining the respective components of the vector representation of σ.

Lemma 8. *Let $\mathcal{G} = (N, C, P, S)$ be a $\mathbb{Z}$-CFCG$_R$. There exists a polynomial-time computable existential Presburger formula $\Psi_{\mathcal{G}}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{A}, \boldsymbol{\sigma}, m)$ defining the generalized Parikh images of runs of $\mathcal{G}$ from s to t.*

Proof. Let $P = \{p_1, \ldots, p_k\}$. Subsequently, we identify productions $p_i \in P$ with their index i. This enables us to write atomic formulas such as $x = p_i$, where x is a first-order variable. Given $p = (a_p, R_p, \boldsymbol{z}_p, w_p) \in P$, we denote by a_p and w_p the corresponding vectors from $\mathbb{N}^N$ as constant terms in the logic, and, similarly, by $\boldsymbol{z}_p$ the corresponding constant vector from $\mathbb{Z}^k$. Remember that equalities between vectors or commutative words in Presburger arithmetic abbreviates the conjunction of formulas expressing equality of their components.

The formula we construct has vectors of free variables $\boldsymbol{s}$ and $\boldsymbol{t}$ for the starting and ending non-terminal commutative words; $\boldsymbol{\alpha_1}, \ldots, \boldsymbol{\alpha_k}$ gathered in the matrix of first order variables $\boldsymbol{A}$, $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_k)$, and a variable m that encode a generalized Parikh image. First, we construct a formula φ_{perm} asserting that $\boldsymbol{\sigma}$ is a permutation on the set $[k]$:

$$\varphi_{\mathrm{perm}}(\boldsymbol{\sigma}) \stackrel{\mathrm{def}}{=} \bigwedge_{i\in[k]} \left(1 \le \sigma_i \le k \ \wedge \bigwedge\nolimits_{j\in[k]} i \ne j \to \sigma_i \ne \sigma_j\right).$$

This formula has size $O(k^2)$ and is thus polynomial in $|\mathcal{G}|$. Now we must "compute" the k partial runs, but first we have to "guess" the starting and ending words of $N^{\circledcirc}$ of each of these partial runs, in order to use the formula from Lemma 7. Let $\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k)$ and $\boldsymbol{T} = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_k)$ and define

$$\begin{aligned}\varphi_{\mathrm{words}}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\sigma}, m, \boldsymbol{S}, \boldsymbol{T}) \stackrel{\mathrm{def}}{=} \boldsymbol{s}_1 = \boldsymbol{s} \ \wedge \bigwedge_{p\in P} (\sigma_k = p \to \boldsymbol{t}_k(a_p) > 0 \wedge \boldsymbol{t} = \boldsymbol{t}_k - a_p + w_p) \ \wedge \\ \wedge \bigwedge_{1\le i<k} \Big[(i < m \to \boldsymbol{s}_i = \boldsymbol{t}_i \wedge \boldsymbol{t}_i = \boldsymbol{s}_{i+1}) \ \wedge \\ \wedge \Big(m \le i \to \Big(\bigwedge_{p\in P} \sigma_i = p \to \boldsymbol{t}_i(a_p) > 0 \wedge \boldsymbol{s}_{i+1} = \boldsymbol{t}_i - a_p + w_p\Big)\Big)\Big].\end{aligned}$$

Here, m is used as in Definition 5, and $\boldsymbol{t}_k(a_p)$ and $\boldsymbol{t}_i(a_p)$ denote components of $\boldsymbol{t}_k$ and $\boldsymbol{t}_i$, respectively, whose index coincides with the index of the only non-zero entry in the constant vector $a_p \in \mathbb{N}^N$. The first two lines enforce that the run is going from $\boldsymbol{s}$ to $\boldsymbol{t}$. The third line imposes $\boldsymbol{s}_m = \boldsymbol{s}$, and the last one ensures that the production p_{σ_i} can be applied from $\boldsymbol{t}_i$ and reaches $\boldsymbol{s}_{i+1}$. We can now express that the k partial runs have Parikh images $\boldsymbol{\alpha}_i$ and are connecting $\boldsymbol{s}_i$ with $\boldsymbol{t}_i$, and that the production p_{σ_i} is not occurring afterwards in the decomposition.

$$\varphi_{\text{runs}}(\boldsymbol{\sigma}, m, \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{T}) \stackrel{\text{def}}{=} \bigwedge_{i\in[k]} \Bigg[(i < m \to \boldsymbol{\alpha}_i = \boldsymbol{0}) \wedge \\ \wedge \left(m \le i \to \left(\varphi_{\mathcal{G}}(\boldsymbol{s}_i, \boldsymbol{t}_i, \boldsymbol{\alpha}_i) \wedge \bigwedge_{1\le j<i} \bigwedge_{p\in P} p = \sigma_j \to \boldsymbol{\alpha}_i(p) = 0 \right) \right) \Bigg].$$

In summary, φ_{perm}, φ_{words} and φ_{runs} enforce the constraints from Definition 5. Putting everything together yields:

$$\Psi_{\mathcal{G}}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{A}, \boldsymbol{\sigma}, m) \stackrel{\text{def}}{=} \exists \boldsymbol{S}, \boldsymbol{T}.$$
$$1 \le m \le k \wedge \varphi_{\text{perm}}(\boldsymbol{\sigma}) \wedge \varphi_{\text{words}}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\sigma}, m, \boldsymbol{S}, \boldsymbol{T}) \wedge \varphi_{\text{runs}}(\boldsymbol{\sigma}, m, \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{T}).$$

Note that the size of $\Psi_{\mathcal{G}}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{A}, \boldsymbol{\sigma}, m)$ is polynomial in $|\mathcal{G}|$. □

By combining $\Psi_{\mathcal{G}}$ with Lemma 6, we obtain the main theorem of this section. Subsequently, $\boldsymbol{u}$ and $\boldsymbol{v}$ are interpreted as vectors over the integers (and not over the naturals); the details are as discussed previously in Section 2.2.

Theorem 9. *Let $\mathcal{G}$ be a $\mathbb{Z}$-CFCG$_R$. There exists a polynomial-time computable existential Presburger formula $\Phi_{\mathcal{G}}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{A}, \boldsymbol{\sigma}, m)$ such that for all s, t in $N^{\odot}$, $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}^C$ and $(\boldsymbol{A}, \sigma, m) \in (\mathbb{N}^k)^k \times \mathbb{N}^k \times \mathbb{N}$ the following are equivalent:*

- $(s, t, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{A}, \sigma, m) \in [\![\Phi_{\mathcal{G}}]\!]$,
- *there is $\gamma \in P^+$ such that $(s, \boldsymbol{u}) \xrightarrow{\gamma}_{\mathcal{G}} (t, \boldsymbol{v})$ and $(\boldsymbol{A}, \sigma, m)$ is a generalized Parikh image of γ.*

In particular, reachability and coverability in $\mathbb{Z}$-CFCG$_R$ are NP*-complete.*

Proof. Thanks to the characterization of generalized Parikh images via $\Psi_{\mathcal{G}}$ obtained from Lemma 8, it suffices to show that the equation obtained in Lemma 6 can be encoded in Presburger arithmetic. For any $c \in C$, this equation can be rewritten as follows:

$$\boldsymbol{v}(c) = \boldsymbol{u}_{|R_m}(c) + \sum_{i=m}^{k} \left[\left(\sum_{p\in P} \boldsymbol{\alpha}_i(p) \cdot \boldsymbol{z}_p \right)_{|R_i} (c) + (\boldsymbol{z}_{p_{\sigma(i)}})_{|R_{i+1}}(c) \right]$$
$$= \lambda_{m,c} \cdot \boldsymbol{u}(c) + \sum_{i=1}^{k} \left[\lambda_{i,c} \cdot \left(\sum_{p\in P} \boldsymbol{\alpha}_i(p) \cdot \boldsymbol{z}_p(c) \right) + \lambda_{i+1,c} \cdot \boldsymbol{z}_{p_{\sigma(i)}}(c) \right]$$

where

$$\lambda_{i,c} = \begin{cases} 0 & \text{if } c \in R_i \text{ or } i < m \\ 1 & \text{otherwise.} \end{cases}$$

Although it is easy to define $\lambda_{i,c}$ in Presburger arithmetic, the above equality is not a syntactically correct Presburger formula since the terms $\lambda_{m,c} \cdot \boldsymbol{u}(c)$ and $\lambda_{i,c} \cdot \boldsymbol{\alpha}_i(p)$ are not linear. To work around this problem, we therefore introduce intermediate variables β_j^c and δ_j^c that enable us to handle the effect of resets in a step-wise fashion. Informally, we want these variables to satisfy the following conditions:

$$\beta_j^c = \lambda_{m,c} \cdot \boldsymbol{u}(c) + \sum_{i=1}^{j} \left[\lambda_{i,c} \cdot \left(\sum_{p\in P} \boldsymbol{\alpha}_i(p) \cdot \boldsymbol{z}_p(c) \right) + \lambda_{i+1,c} \cdot \boldsymbol{z}_{p_{\sigma(i)}}(c) \right]$$

for $j \in [0, k]$ and $c \in C$, and

$$\delta_j^c = \lambda_{m,c} \cdot \boldsymbol{u}(c) + \sum_{i=1}^{j-1} \left[\lambda_{i,c} \cdot \left(\sum_{p\in P} \boldsymbol{\alpha}_i(p) \cdot \boldsymbol{z}_p(c) \right) + \lambda_{i+1,c} \cdot \boldsymbol{z}_{p_{\sigma(i)}}(c) \right] + \lambda_{j,c} \cdot \sum_{p\in P} \boldsymbol{\alpha}_j(p) \cdot \boldsymbol{z}_p(c)$$

for $j \in [1, k]$ and $c \in C$. This approach is formalized in the formula $\varphi_{\text{counters}}$ below. First, remember that $R_i \stackrel{\text{def}}{=} R_{p_{\sigma(i)}} \cup \cdots \cup R_{p_{\sigma(k)}}$, and, therefore, for any $c \in C$:

$$c \in R_i \iff c \in R_{p_{\sigma(i)}} \cup \cdots \cup R_{p_{\sigma(k)}}$$
$$\iff \bigvee_{i\le j\le k} c \in R_{p_{\sigma(j)}}$$

$$\iff \bigvee_{i \le j \le k} \bigvee_{d \in R_{p_{\sigma(j)}}} d = c.$$

We therefore introduce the notation $c \in R_x$, where x can be a first-order variable, as an abbreviation for the following formula:

$$\bigvee_{j=1}^{k} (j \ge x) \wedge \Big(\bigwedge_{\ell=1}^{k} (\ell = \sigma_j \to \bigvee_{d \in R_{p_\ell}} d = c)\Big).$$

Note that formulas of the form $\bigwedge_{\ell=1}^{k} \ell = \sigma_j \to \dots$ are used when we need to use σ_j as an index (which would not be correct since σ_j is a first-order variable). To improve readability, we also write $\lambda_{i,c} = 0$ to denote the formula $c \in R_i \vee i < m$, and $\lambda_{i,c} = 1$ to denote $c \notin R_i \wedge i \ge m$. Now, $\varphi_{\text{counters}}$ can be defined as follows:

$$\begin{aligned}
\varphi_{\text{counters}}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{A}, \boldsymbol{\sigma}, m) \stackrel{\text{def}}{=} \exists \boldsymbol{B}. \exists \boldsymbol{D}. \\
\bigwedge_{c \in C} \Big\{ & (\lambda_{m,c} = 0 \to \beta_0^c = 0) \wedge (\lambda_{m,c} = 1 \to \beta_0^c = \boldsymbol{u}(c)) \wedge \\
& \wedge\ \boldsymbol{v}(c) = \beta_k^c \wedge \\
& \wedge \bigwedge_{j=1}^{k} \Big[(\lambda_{j,c} = 0 \to \delta_j^c = \beta_{j-1}^c) \wedge \\
& \wedge (\lambda_{j,c} = 1 \to \delta_j^c = \beta_{j-1}^c + \sum_{p \in P} \boldsymbol{\alpha}_j(p) \cdot \boldsymbol{z}_p(c)) \wedge \\
& \wedge (\lambda_{j+1,c} = 0 \to \beta_j^c = \delta_j^c) \\
& \wedge \Big(\lambda_{j+1,c} = 1 \to \bigwedge_{\ell=1}^{k} (\ell = \sigma_j \to \beta_j^c = \delta_j^c + z_{p_\ell}(c)) \Big) \Big] \Big\}.
\end{aligned}$$

In this formula, the first line deals with β_0^c: it is either 0 or $\boldsymbol{u}(c)$ depending on whether $c \in R_m$, i.e., whether c is reset at some point in the run. The second line gives the desired value to $\boldsymbol{v}$. The four last lines compute β_j^c (respectively δ_j^c) from δ_j^c (respectively β_{j-1}^c): if $\lambda_{j,c} = 0$ then nothing is added to the sum.

Since satisfiability in existential Presburger arithmetic is NP-complete, this allows us to conclude that reachability in $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ is in NP and hence NP-complete. By Theorem 4, the same result carries over to coverability in $\mathbb{Z}$-CFCG$_{\mathrm{R}}$. □

Finally, we obtain as a corollary an existential Presburger formula that defines the reachability set of a $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ that we will use in the next section.

Corollary 10. *Let $\mathcal{G}$ be a $\mathbb{Z}$-CFCG$_{\mathrm{R}}$. There exists a polynomial-time computable existential Presburger formula $\Phi_{\mathrm{rs}}^{\mathcal{G}}(\boldsymbol{u}, \boldsymbol{v})$ such that*

$$(\boldsymbol{u}, \boldsymbol{v}) \in [\![\Phi_{\mathrm{rs}}^{\mathcal{G}}]\!] \iff \boldsymbol{v} \in \mathit{reach}(\mathcal{G}, \boldsymbol{u}).$$

4. Inclusion for $\mathbb{Z}$-CFCG$_{\mathrm{R}}$

In this section, we study inclusion problems for $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ and subclasses thereof. We first remark that the general problem is coNEXP-complete. Subsequently, we show that the inclusion problem is Π_2^{P}-complete, even for the smallest subclass $\mathbb{Z}$-VAS. The proof of the lower bound requires us to develop a new Π_2^{P}-complete variant of the classic Subset Sum problem, which we believe is a contribution of independent interest.

4.1. The general case

In this section, we show the following theorem.

Theorem 11. *The inclusion problem for $\mathbb{Z}$-CFCG$_{\mathrm{R}}$ is* coNEXP*-complete.*

In the conference version of this paper [26], we showed that inclusion is coNEXP-hard for $\mathbb{Z}$-VASS, even when numbers are encoded in unary. Our construction was subsequently strengthened in [25] were it was shown that inclusion is already coNEXP-hard for $\mathbb{Z}$-VASS when counter updates are restricted to be non-negative and given in unary. The coNEXP-lower bound of Theorem 11 consequently follows from [25].

Thanks to our characterization of reachability sets of $\mathbb{Z}$-CFCG$_R$ via existential Presburger formulas of polynomial size obtained from Corollary 10, a matching upper bound is also not difficult to obtain. Let $\mathcal{G}$ and $\mathcal{H}$ be $\mathbb{Z}$-CFCG$_R$, $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}^C$, and let $\Phi_{rs}^{\mathcal{G}}(\boldsymbol{x}, \boldsymbol{z})$ and $\Phi_{rs}^{\mathcal{H}}(\boldsymbol{y}, \boldsymbol{z})$ be the formulas from Corollary 10. We then have that

$$\begin{aligned} & reach(\mathcal{G}, \boldsymbol{u}) \subseteq reach(\mathcal{H}, \boldsymbol{v}) \\ \iff & \psi \stackrel{\text{def}}{=} \neg(\exists \boldsymbol{z}. \Phi_{rs}^{\mathcal{G}}(\boldsymbol{u}/\boldsymbol{x}, \boldsymbol{z}) \wedge \neg(\Phi_{rs}^{\mathcal{H}}(\boldsymbol{v}/\boldsymbol{y}, \boldsymbol{z}))) \text{ is valid.} \end{aligned}$$

Bringing ψ into prenex normal form yields a Π_2-PA sentence for which validity can be decided in coNEXP, cf. Proposition 1. This concludes the proof of Theorem 11.

4.2. Inclusion for $\mathbb{Z}$-VAS

In this section, we show that already for $\mathbb{Z}$-VAS, the inclusion problem is computationally difficult.

Theorem 12. *The inclusion problem for $\mathbb{Z}$-VAS is Π_2^{P}-complete.*

In fact, the lower bound already holds when numbers are encoded in unary (see Theorem 15 in the following subsection). The starting point for our lower bound here (for the binary encoding) is the following generalization of Subset Sum, which is known to be complete for the second level of the polynomial hierarchy. Recall that, unless explicitly stated otherwise, all numbers in the considered problem settings are written in binary.

Π_2-Subset Sum
INPUT: Finite sets $U, V \subseteq \mathbb{N}$ and $t \in \mathbb{N}$.
QUESTION: For every $U' \subseteq U$, does there exist a $V' \subseteq V$ such that $\sum U' + \sum V' = t$?

Here and below, for $A \subseteq \mathbb{N}$ we use $\sum A$ as a shorthand for $\sum_{a \in A} a$.

Proposition 13 *(Berman et al. [45]).* Π_2-Subset Sum *is Π_2^{P}-complete.*

There is no obvious reduction from Π_2-Subset Sum to inclusion for $\mathbb{Z}$-VAS. Informally, the lack of control structure in $\mathbb{Z}$-VAS makes it difficult to encode the alternation of quantifiers (for all U' there exists a V') and the subset constraints (each element of U, respectively V, participates at most once in U', respectively V'). Accordingly, we define another variant of Subset Sum, implicit in [46].

Simultaneous Subset Sum
INPUT: A finite set $W \subseteq \mathbb{N}$, and $h, 2^m, t \in \mathbb{N}$ such that $t < h$.
QUESTION: For every $i \in [0, 2^m - 1]$, does there exist a $W' \subseteq W$ such that $\sum W' = t + i \cdot h$?

Lemma 14. Simultaneous Subset Sum *is Π_2^{P}-complete.*

Proof. The problem is easily seen to be in Π_2^{P}. To show hardness, let $U = \{u_1, \ldots, u_r\}$, $V = \{v_1, \ldots, v_s\} \subseteq \mathbb{N}$ and $t \in \mathbb{N}$ form an instance of Π_2-Subset Sum. We define the corresponding instance of Simultaneous Subset Sum as follows:

- $h \stackrel{\text{def}}{=} \sum U + \sum V + 1$;
- $W \stackrel{\text{def}}{=} \{u_1 + h, u_2 + 2 \cdot h, \ldots, u_r + 2^{r-1} \cdot h, v_1, \ldots, v_s\}$;
- $m \stackrel{\text{def}}{=} r$, and t is unchanged.

We now show that this reduction is faithful. With no loss of generality, we may assume $t < h$, otherwise the original instance is clearly a no-instance. We actually show a slightly stronger statement: define a bijection between $T \stackrel{\text{def}}{=} \{t + i \cdot h \colon i \in [0, 2^r - 1]\}$ and 2^U as follows: associate with $t + i \cdot h \in T$ the set $U_i \subseteq U$ such that

$$u_j \in U_i \iff 2^{j-1} \text{ has non-zero coefficient in the binary expansion of } i,$$

i.e., U_i is such that $i = \sum_{u_j \in U_i} 2^{j-1}$. We claim that every $t + i \cdot h \in T$ can be represented as a sum of some $W' \subseteq W$ if and only if for the subset $U_i \subseteq U$ there is some $V' \subseteq V$ such that $\sum U_i + \sum V' = t$. Indeed, observe that

$$\begin{aligned} & \sum U_i + \sum V' = t \\ \iff & \sum_{u_j \in U_i} u_j + \sum V' = t \end{aligned}$$

$$\iff \sum_{u_j \in U_i} u_j + \sum V' + \sum_{u_j \in U_i} 2^{j-1} \cdot h = t + i \cdot h$$

$$\iff \sum_{u_j \in U_i} (u_j + 2^{j-1} \cdot h) + \sum V' = t + i \cdot h \tag{3}$$

$$\iff \sum W' = t + i \cdot h \quad \text{for some } W' \subseteq W, \tag{4}$$

where the implication (4) $\Rightarrow$ (3) holds by our choice of h. □

We now apply Lemma 14 in order to obtain the lower bound for $\mathbb{Z}$-VAS inclusion. Let $W = \{w_1, \ldots, w_n\} \subseteq \mathbb{N}$ and $h, 2^m, t \in \mathbb{N}$ define an instance of Simultaneous Subset Sum. Set $\boldsymbol{w} \stackrel{\text{def}}{=} (w_1, \ldots, w_n)$, then this instance is a yes-instance if and only if

$$\text{for all } i \in [0, 2^m - 1] \text{ there exists a } \boldsymbol{y} \in \{0,1\}^n \text{ such that } \boldsymbol{w} \cdot \boldsymbol{y} = t + i \cdot h. \tag{5}$$

It follows from the discussion in Section 2, p. 151, that the $\mathbb{Z}$-VAS inclusion problem can equivalently be expressed as follows: For matrices $A \in \mathbb{Z}^{d \times r}$ and $B \in \mathbb{Z}^{d \times s}$, and some $\boldsymbol{v} \in \mathbb{Z}^d$, decide whether

$$\text{for all } \boldsymbol{x} \in \mathbb{N}^r, \text{ there exists a } \boldsymbol{y} \in \mathbb{N}^s \text{ such that } A \cdot \boldsymbol{x} + B \cdot \boldsymbol{y} = \boldsymbol{v}. \tag{6}$$

We now transform (5) into the form (6), thus proving Π_2^{P}-hardness of $\mathbb{Z}$-VAS inclusion. Observe that (5) is almost of the same form as (6). However, the domains of the quantified variables in (5) and (6) disagree. In order to overcome this issue, first we observe that the existence of some $\boldsymbol{y} \in \{0,1\}^n$ is equivalent to the existence of $\boldsymbol{y}, \boldsymbol{z} \in \mathbb{N}^n$ such that $\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{1}$. Second, the restriction of i to numbers less than 2^m can be avoided by introducing another existentially quantified variable $c \in \mathbb{N}$ and replacing i with $i - 2^m \cdot c$ in (5). Informally speaking, this ensures that i is evaluated only modulo 2^m and, effectively, does not "overflow". Putting everything together, we claim that (5) is equivalent to the following condition:

$$\text{for all } i \in \mathbb{N} \text{ there exist } \boldsymbol{y}, \boldsymbol{z} \in \mathbb{N}^n \text{ and } c \in \mathbb{N} \text{ such that } \boldsymbol{w} \cdot \boldsymbol{y} = t + (i - 2^m \cdot c) \cdot h \text{ and } \boldsymbol{y} + \boldsymbol{z} = \boldsymbol{1}. \tag{7}$$

Indeed, (5) implies (7); conversely, (7) implies (5), because for $i < 2^m$ no $c > 0$ can satisfy the first equation in (7): the right-hand side is $t + (i - 2^m \cdot c) \cdot h \le t - h < 0$, while the left-hand side is $\boldsymbol{w} \cdot \boldsymbol{y} \ge 0$ for all $\boldsymbol{y} \in \mathbb{N}^n$. It is readily verified that (7) is of the form (6) with $r = 1$, $s = 2 \cdot n + 1$, $d = n + 1$ and

$$A \stackrel{\text{def}}{=} \begin{pmatrix} -h \\ \boldsymbol{0} \end{pmatrix}, \quad B \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{w}^{\mathsf{T}} & \boldsymbol{0}^{\mathsf{T}} & 2^m \cdot h \\ I_n & I_n & \boldsymbol{0} \end{pmatrix}, \quad \boldsymbol{v} \stackrel{\text{def}}{=} \begin{pmatrix} t \\ \boldsymbol{1} \end{pmatrix}.$$

This concludes the proof of the Π_2^{P}-hardness of $\mathbb{Z}$-VAS inclusion when numbers are encoded in binary.

We now turn towards a matching upper bound for $\mathbb{Z}$-VAS inclusion. Given a $\mathbb{Z}$-VAS $\mathcal{G} = (\{S\}, C, P, S)$ such that $P = \{(S, \boldsymbol{v}_1, S), \ldots, (S, \boldsymbol{v}_n, S)\}$ and a configuration $(S, \boldsymbol{v})$, we obviously have

$$\mathit{reach}(\mathcal{G}, \boldsymbol{v}) = \left\{ \boldsymbol{v} + \sum_{1 \le i \le n} \lambda_i \cdot \boldsymbol{v}_i : \lambda_i \in \mathbb{N}, 1 \le i \le n \right\},$$

which is a linear (and thus semi-linear) set in $\mathbb{Z}^d$. It follows from the results in [47] (and implicitly also from [21]) that the inclusion problem for semi-linear sets in $\mathbb{Z}^d$ given by their generators is in Π_2^{P} (and is, in fact, Π_2^{P}-complete). As a consequence, we conclude that the inclusion problem for $\mathbb{Z}$-VAS is also contained in Π_2^{P}, and hence is Π_2^{P}-complete.

4.3. Inclusion for $\mathbb{Z}$-VAS under unary encoding of integers

It is interesting to note that, modulo standard computational complexity assumptions, both Π_2-Subset Sum and Simultaneous Subset Sum are only Π_2^{P}-hard if numbers are represented in binary: it is folklore that Subset Sum has a pseudo-polynomial time dynamic programming algorithm, and thus the unary versions of Π_2-Subset Sum and Simultaneous Subset Sum are not Π_2^{P}-hard unless the polynomial hierarchy collapses. This phenomenon, however, does not extend to the inclusion problem for $\mathbb{Z}$-VAS. Specifically, Theorem 15 gives a stronger form of the lower bound in Theorem 12:

Theorem 15. *The inclusion problem for $\mathbb{Z}$-VAS remains Π_2^{P}-hard even when numbers are encoded in unary.*

Proof. We reduce the inclusion problem with numbers encoded in binary to the inclusion problem with numbers encoded in unary. More precisely, we transform the problem described in Equation (6) into an instance of the same problem with numbers encoded in unary. Recall that this problem is to decide, given matrices $A \in \mathbb{Z}^{d \times r}$, $B \in \mathbb{Z}^{d \times s}$ and a vector $\boldsymbol{v} \in \mathbb{Z}^d$ whether

$$\text{for all } \boldsymbol{x} \in \mathbb{N}^r, \text{ there exists } \boldsymbol{y} \in \mathbb{N}^s \text{ such that } A \cdot \boldsymbol{x} + B \cdot \boldsymbol{y} = \boldsymbol{v}.$$

We construct an instance $(A', B', \boldsymbol{v}')$ of the same problem but where A', B' and $\boldsymbol{v}$ use only numbers among $\{-2, -1, 0, 1, 2\}$. We first introduce some auxiliary definitions.

Let m be the minimal number of bits sufficient to write in binary every number occurring in A, B and $\boldsymbol{v}$ disregarding the signs—in other words, m is the smallest natural number such that the absolute value of every entry of A (respectively B and $\boldsymbol{v}$) is smaller than 2^m. Given a vector $\boldsymbol{x} = (x_1, \ldots, x_d) \in [-2^m + 1, 2^m - 1]^d$, define $b(\boldsymbol{x}) \in \{-1, 0, 1\}^{d \cdot m}$, the *binary expansion* of $\boldsymbol{x}$, as

$$b(\boldsymbol{x}) = (x_1^{(m-1)}, x_1^{(m-2)}, \ldots, x_1^{(0)}, x_2^{(m-1)}, \ldots, x_d^{(0)})$$

where, for $i \in [d]$, $x_i = \sum_{j=0}^{m-1} x_i^{(j)} \cdot 2^j$ denotes the unique binary representation of x_i (for negative numbers, coefficients $x_i^{(j)}$ are in $\{-1, 0\}$, and for positive numbers they are in $\{0, 1\}$). Conversely, given

$$\boldsymbol{y} = (y_1^{(m-1)}, y_1^{(m-2)}, \ldots, y_1^{(0)}, y_2^{(m-1)}, \ldots, y_d^{(0)}) \in \mathbb{Z}^{d \cdot m},$$

we define the reverse function $r(\boldsymbol{y}) = (r_1, \ldots, r_d) \in \mathbb{Z}^d$ with $r_i = \sum_{j=0}^{m-1} y_i^{(j)} \cdot 2^j$ for $i \in [d]$. Note that for any $\boldsymbol{x} \in [-2^m + 1, 2^m - 1]^d$, $r(b(\boldsymbol{x})) = \boldsymbol{x}$, but $b(r(\boldsymbol{y}))$ is not necessarily equal to $\boldsymbol{y}$ since components of $\boldsymbol{y} \in \mathbb{Z}^{d \cdot m}$ do not have to belong to $\{0, 1\}$ or $\{-1, 0\}$.

Let us also introduce the following definition. Given an integer $n \in \mathbb{Z}$, its *weak binary representation* is an expansion of n as a sum of powers of 2 with arbitrary coefficients from $\mathbb{Z}$ (the usual binary representation only allows coefficients 0 and 1). Then, with $d = 1$, the set $\{\boldsymbol{y} \in \mathbb{Z}^m : r(\boldsymbol{y}) = n\}$ is the set of all weak binary representations of n of *height* m. For instance, $(0, 0, 1, 1, 1)$ and $(1, 0, -1, -3, 1)$ are two weak binary representations of 7, both of height 5.

Now define the matrix $D_m = (d_{i,j}) \in [-2, 2]^{m \times (m-1)}$ by the following rule:

$$d_{i,j} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } i = j, \\ -2 & \text{if } i = j + 1, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

For instance,

$$D_5 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & -2 \end{pmatrix}.$$

The image $\{D_m \cdot \boldsymbol{z} \ : \ \boldsymbol{z} \in \mathbb{Z}^{m-1}\}$ of the matrix D_m is the set of all weak binary representations of height m of the integer 0:

Claim 16. *For every vector $\boldsymbol{y} \in \mathbb{Z}^m$, $r(\boldsymbol{y}) = 0$ iff there exists a $\boldsymbol{z} \in \mathbb{Z}^{m-1}$ such that $D_m \cdot \boldsymbol{z} = \boldsymbol{y}$.*

We prove Claim 16 at the end of the section. By linearity, it follows that the set $\{\boldsymbol{y} + D_m \cdot \boldsymbol{z} : \boldsymbol{z} \in \mathbb{Z}^m\}$ is the set of all weak binary representations of $r(\boldsymbol{y})$. For instance,

$$\begin{pmatrix} 1 \\ 0 \\ -1 \\ -3 \\ 1 \end{pmatrix} + D_5 \cdot \begin{pmatrix} -1 \\ -2 \\ -2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix},$$

and all height-5 weak binary representations of 7 can be obtained this way. The same property can be obtained for vectors of dimension d by defining

$$E_m^d = \begin{pmatrix} D_m & 0 & \ldots & 0 \\ 0 & D_m & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & D_m \end{pmatrix} \in [-2, 2]^{(d \cdot m) \times (d \cdot (m-1))}.$$

We now define the following instance of $\mathbb{Z}$-VAS inclusion:

- $A' = b(A) \in \{-1, 0, 1\}^{(d \cdot m) \times r}$, i.e., A' is the matrix whose columns are the vectors $b(\boldsymbol{a})$ for every column $\boldsymbol{a}$ of A;
- $B' = \begin{pmatrix} b(B) & E_m^d & -E_m^d \end{pmatrix} \in [-2, 2]^{(d \cdot m) \times (s + 2d \cdot (m-1))}$; and
- $\boldsymbol{v}' = b(\boldsymbol{v}) \in \{-1, 0, 1\}^{d \cdot m}$.

This instance is a yes-instance if and only if for all $\boldsymbol{x} \in \mathbb{N}^r$, there exists a $(\boldsymbol{y}, \boldsymbol{z}_1, \boldsymbol{z}_2) \in \mathbb{N}^{s+2d\cdot(m-1)}$ such that

$$b(A)\cdot\boldsymbol{x} + \begin{pmatrix} b(B) & E_m^d & -E_m^d \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix} = b(\boldsymbol{v}),$$

i.e., if there exists a $\boldsymbol{t} \in \mathbb{Z}^{d\cdot(m-1)}$ such that

$$b(B)\cdot\boldsymbol{y} + E_m^d \cdot \boldsymbol{t} = b(\boldsymbol{v}) - b(A)\cdot\boldsymbol{x}.$$

According to Claim 16, there exists such a $\boldsymbol{t}$ if and only if $b(B)\cdot\boldsymbol{y}$ and $b(\boldsymbol{v}) - b(A)\cdot\boldsymbol{x}$ are two weak binary representations of the same vector. By application of r on both sides and due to the linearity of r, we obtain that for any $\boldsymbol{x} \in \mathbb{N}^r$ there exists some $\boldsymbol{y} \in \mathbb{N}^s$ such that $A\cdot\boldsymbol{x} + B\cdot\boldsymbol{y} = \boldsymbol{v}$. This completes the proof of Theorem 15. □

It remains to prove Claim 16. Recall that we wish to show that for any $\boldsymbol{y} = (y^{(m-1)}, \ldots, y^{(0)})$,

$$r(\boldsymbol{y}) \stackrel{\text{def}}{=} \sum_{j=0}^{m-1} y^{(j)} \cdot 2^j = 0 \iff \exists \boldsymbol{z}.\ D_m \cdot \boldsymbol{z} = \boldsymbol{y}.$$

Let C_m be the square matrix that consists of the $m-1$ first rows of D_m, i.e., C_m is such that

$$D_m = \begin{pmatrix} C_m \\ 0 \cdots 0 \ -2 \end{pmatrix}.$$

We then have:

$$\exists \boldsymbol{z}.\ D_m \cdot \boldsymbol{z} = \boldsymbol{y} \iff \exists \boldsymbol{z}.\ C_m \cdot \boldsymbol{z} = (y^{(m-1)}, \ldots, y^{(1)}) \text{ and } -2 \cdot z^{(1)} = y^{(0)}. \tag{8}$$

Since C_m is invertible, for every $\boldsymbol{y}$ there exists exactly one $\boldsymbol{z} = (z^{(m-1)}, \ldots, z^{(1)})$ such that $C_m \cdot \boldsymbol{z} = (y^{(m-1)}, \ldots, y^{(1)})$; in particular $z^{(i)} = \sum_{j=i}^{m-1} y^{(j)} \cdot 2^{j-i}$ for every $i \in [m-1]$. Note that $\boldsymbol{z} \in \mathbb{Z}^{m-1}$ whenever $\boldsymbol{y} \in \mathbb{Z}^{m-1}$. Therefore, the equivalence in (8) can be reformulated and continued as follows:

$$\begin{aligned} \exists \boldsymbol{z}.\ D_m \cdot \boldsymbol{z} = \boldsymbol{y} &\iff -2 \sum_{j=1}^{m-1} y^{(j)} \cdot 2^{j-1} = y^{(0)} \\ &\iff 0 = \sum_{j=1}^{m-1} y^{(j)} \cdot 2^j + y^{(0)} \cdot 2^0 \\ &\iff r(\boldsymbol{y}) = 0. \end{aligned}$$

This concludes the proof of Claim 16.

5. Conclusion

In this paper, we studied standard decision problems for $\mathbb{Z}$-CFCG$_\mathrm{R}$, an extension of context-free commutative grammars with integer counters and reset operations on them. We showed that reachability and coverability are logarithmic-space inter-reducible in this class and NP-complete. For our NP-upper bound, we showed that the reachability relation for $\mathbb{Z}$-CFCG$_\mathrm{R}$ can be defined by an existential formula of Presburger arithmetic of polynomial size. In particular, this implies that $\mathbb{Z}$-CFCG$_\mathrm{R}$ have semi-linear reachability sets. Moreover, we showed that inclusion for $\mathbb{Z}$-CFCG$_\mathrm{R}$ is, in general, coNEXP-complete, and Π_2^P-complete for $\mathbb{Z}$-VAS, a subclass of $\mathbb{Z}$-CFCG$_\mathrm{R}$. In order to show the latter lower bound, we introduced a new Π_2^P-complete decision problem Simultaneous Subset Sum, a variant of the classical Subset Sum problem.

One can view $\mathbb{Z}$-CFCG$_\mathrm{R}$ as an over-approximation of classical reset Petri nets in which places may contain a negative number of tokens. Hence, Theorem 9 enables witnessing non-reachability in reset Petri nets in coNP, i.e., at comparatively low computational costs given that the problem is, in general, undecidable. In particular, our characterization of reachability in terms of existential Presburger arithmetic immediately enables the use of SMT solvers and thus paves the way for an easy implementation of our approach. This approach, over-approximating reachability in Petri nets, has recently been proved surprisingly efficient when applied to real-world instances [48]. As for future work, it would be interesting to investigate whether $\mathbb{Z}$-CFCG$_\mathrm{R}$ can be extended with transfer operations while retaining definability of their reachability sets in Presburger arithmetic.

Acknowledgements

We would like to thank Sylvain Schmitz, Philippe Schnoebelen and the anonymous reviewers of RP'14 for their helpful comments and suggestions on an earlier version of this paper.

References

[1] S.M. German, A.P. Sistla, Reasoning about systems with many processes, J. ACM 39 (3) (1992) 675–735.
[2] R. Lipton, The reachability problem is exponential-space-hard, Tech. rep., Yale University, New Haven, CT, 1976.
[3] C. Rackoff, The covering and boundedness problems for vector addition systems, Theoret. Comput. Sci. 6 (2) (1978) 223–231.
[4] E.W. Mayr, An algorithm for the general Petri net reachability problem, SIAM J. Comput. 13 (3) (1984) 441–460.
[5] S.R. Kosaraju, Decidability of reachability in vector addition systems (preliminary version), in: H.R. Lewis, B.B. Simons, W.A. Burkhard, L.H. Landweber (Eds.), Symposium on Theory of Computing (STOC'82), ACM, 1982, pp. 267–281.
[6] J. Lambert, A structure to decide reachability in Petri nets, Theoret. Comput. Sci. 99 (1) (1992) 79–104.
[7] J. Leroux, Vector addition systems reachability problem (a simpler solution), in: A. Voronkov (Ed.), Turing-100 – the Alan Turing Centenary, in: EPiC Series, vol. 10, EasyChair, 2012, pp. 214–228.
[8] J. Leroux, S. Schmitz, Demystifying reachability in vector addition systems, in: Logic in Computer Science (LICS'15), IEEE Computer Society, 2015, pp. 56–67.
[9] M. Hack, The equality problem for vector addition systems is undecidable, Theoret. Comput. Sci. 2 (1) (1976) 77–95.
[10] P. Jančar, Nonprimitive recursive complexity and undecidability for Petri net equivalences, Theoret. Comput. Sci. 256 (1–2) (2001) 23–30.
[11] A. Kaiser, D. Kroening, T. Wahl, A widening approach to multithreaded program verification, ACM Trans. Program. Lang. Syst. 36 (4) (2014) 14:1–14:29.
[12] M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, D. Edmond, Synchronization and cancellation in workflows based on reset nets, Int. J. Coop. Inf. Syst. 18 (1) (2009) 63–114.
[13] C. Dufourd, A. Finkel, P. Schnoebelen, Reset nets between decidability and undecidability, in: K.G. Larsen, S. Skyum, G. Winskel (Eds.), Automata, Languages and Programming (ICALP'98), in: Lect. Notes Comp. Sci., vol. 1443, Springer, 1998, pp. 103–115.
[14] A. Finkel, S. Göller, C. Haase, Reachability in register machines with polynomial updates, in: [49], 2013, pp. 409–420.
[15] P. Schnoebelen, Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets, in: P. Hlinený, A. Kucera (Eds.), Mathematical Foundations of Computer Science (MFCS'10), in: Lect. Notes Comp. Sci., vol. 6281, Springer, 2010, pp. 616–628.
[16] D. Huynh, The complexity of equivalence problems for commutative grammars, Inform. Control 66 (1–2) (1985) 103–121.
[17] H. Yen, On reachability equivalence for BPP-nets, Theoret. Comput. Sci. 179 (1–2) (1997) 301–317.
[18] E.W. Mayr, J. Weihmann, Complexity results for problems of communication-free Petri nets and related formalisms, Fund. Inform. 137 (1) (2015) 61–86.
[19] J.E. Hopcroft, J. Pansiot, On the reachability problem for 5-dimensional vector addition systems, Theoret. Comput. Sci. 8 (1979) 135–159.
[20] D.T. Huynh, Commutative grammars: the complexity of uniform word problems, Inform. Control 57 (1) (1983) 21–39.
[21] D.T. Huynh, A simple proof for the Σ_2^p upper bound of the inequivalence problem for semilinear sets, Elektron. Inform. Kybernet. 22 (4) (1986) 147–156.
[22] J. Esparza, Petri nets, commutative context-free grammars, and basic parallel processes, Fund. Inform. 31 (1) (1997) 13–25.
[23] E. Kopczynski, A.W. To, Parikh images of grammars: complexity and applications, in: Logic in Computer (LICS'10), IEEE, 2010, pp. 80–89.
[24] E. Kopczyński, Complexity of problems of commutative grammars, Log. Methods Comput. Sci. 11 (1) (2015).
[25] C. Haase, P. Hofman, Tightening the complexity of equivalence problems for commutative grammars, in: N. Ollinger, H. Vollmer (Eds.), Symposium on Theoretical Aspects of Computer Science (STACS'15), in: LIPIcs, vol. 47, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016, pp. 41:1–41:14.
[26] C. Haase, S. Halfon, Integer vector addition systems with states, in: J. Ouaknine, I. Potapov, J. Worrell (Eds.), Reachability Problems (RP'14), in: Lect. Notes Comp. Sci., vol. 8762, Springer, 2014, pp. 112–124.
[27] E.W. Mayr, J. Weihmann, Completeness results for generalized communication-free Petri nets with arbitrary arc multiplicities, Fund. Inform. 143 (3–4) (2016) 355–391.
[28] W. Plandowski, W. Rytter, Complexity of language recognition problems for compressed words, in: J. Karhumäki, H. Maurer, G. Păun, G. Rozenberg (Eds.), Jewels Are Forever, 1999, pp. 262–272.
[29] H. Seidl, T. Schwentick, A. Muscholl, P. Habermehl, Counting in trees for free, in: J. Díaz, J. Karhumäki, A. Lepistö, D. Sannella (Eds.), Automata, Languages and Programming (ICALP'04), in: Lect. Notes Comp. Sci., vol. 3142, Springer, 2004, pp. 1136–1149.
[30] K.N. Verma, H. Seidl, T. Schwentick, On the complexity of equational horn clauses, in: R. Nieuwenhuis (Ed.), Automated Deduction – CADE-20, in: Lect. Notes Comp. Sci., vol. 3632, Springer, 2005, pp. 337–352.
[31] C. Haase, S. Kreutzer, J. Ouaknine, J. Worrell, Reachability in succinct and parametric one-counter automata, in: M. Bravetti, G. Zavattaro (Eds.), Concurrency Theory (CONCUR'09), in: Lect. Notes Comp. Sci., vol. 5710, Springer, 2009, pp. 369–383.
[32] M. Hague, A.W. Lin, Model checking recursive programs with numeric data types, in: G. Gopalakrishnan, S. Qadeer (Eds.), Computer Aided Verification (CAV'11), in: Lect. Notes Comp. Sci., vol. 6806, Springer, 2011, pp. 743–759.
[33] R. David, H. Alla, Continuous Petri nets, in: Proceedings of the 8th European Workshop on Application and Theory of Petri Nets, 1987, pp. 275–294.
[34] E. Fraca, S. Haddad, Complexity analysis of continuous Petri nets, Fund. Inform. 137 (1) (2015) 1–28.
[35] G. Păun, A new generative device: valence grammars, Rev. Roumaine Math. Pures Appl. 25 (6) (1980) 911–924.
[36] S.A. Greibach, Remarks on blind and partially blind one-way multicounter machines, Theoret. Comput. Sci. 7 (1978) 311–324.
[37] H.J. Hoogeboom, Context-free valence grammars – revisited, in: W. Kuich, G. Rozenberg, A. Salomaa (Eds.), Developments in Language Theory (DLT'01), in: Lect. Notes Comp. Sci., vol. 2295, Springer, 2001, pp. 293–303.
[38] H. Fernau, R. Stiebe, Sequential grammars and automata with valences, Theoret. Comput. Sci. 276 (1–2) (2002) 377–405.
[39] P. Buckheister, G. Zetzsche, Semilinearity and context-freeness of languages accepted by valence automata, in: [49], 2013, pp. 231–242.
[40] I. Borosh, L. Treybing, Bounds on positive integral solutions of linear Diophantine equations, Proc. Amer. Math. Soc. 55 (1976) 299–304.
[41] E. Grädel, Dominoes and the complexity of subclasses of logical theories, Ann. Pure Appl. Logic 43 (1) (1989) 1–30.
[42] C. Haase, Subclasses of Presburger arithmetic and the weak EXP hierarchy, in: T.A. Henzinger, D. Miller (Eds.), Joint Meeting of Computer Science Logic (CSL) and Logic in Computer Science (LICS), CSL-LICS'14, ACM, 2014, pp. 47:1–47:10.
[43] S. Ginsburg, E. Spanier, Semigroups, Presburger formulas and languages, Pacific J. Math. 16 (2) (1966) 285–296.
[44] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, NY, USA, 1979.
[45] P. Berman, M. Karpinski, L.L. Larmore, W. Plandowski, W. Rytter, On the complexity of pattern matching for highly compressed two-dimensional texts, J. Comput. System Sci. 65 (2) (2002) 332–350.
[46] D. Chistikov, R. Majumdar, Unary pushdown automata and straight-line programs, in: J. Esparza, P. Fraigniaud, T. Husfeldt, E. Koutsoupias (Eds.), Automata, Languages, and Programming (ICALP'14), Part II, in: Lect. Notes Comp. Sci., vol. 8573, Springer, 2014, pp. 146–157.
[47] D. Chistikov, C. Haase, The taming of the semi-linear set, in: Automata, Languages and Programming (ICALP'16), 2016, in press.
[48] M. Blondin, A. Finkel, C. Haase, S. Haddad, Approaching the coverability problem continuously, in: M. Chechik, J. Raskin (Eds.), Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16), in: Lect. Notes Comp. Sci., vol. 9636, Springer, 2016, pp. 480–496.
[49] K. Chatterjee, J. Sgall (Eds.), Mathematical Foundations of Computer Science 2013 (MFCS'13), Lect. Notes Comp. Sci., vol. 8087, Springer, 2013.