The Complexity of Bidirected Reachability in Valence Systems

Moses Ganardi
Max Planck Institute for Software Systems (MPI-SWS)
Kaiserslautern, Germany

Rupak Majumdar
Max Planck Institute for Software Systems (MPI-SWS)
Kaiserslautern, Germany

Georg Zetzsche
Max Planck Institute for Software Systems (MPI-SWS)
Kaiserslautern, Germany

ABSTRACT

Reachability problems in infinite-state systems are often subject to extremely high complexity. This motivates the investigation of efficient overapproximations, where we add transitions to obtain a system in which reachability can be decided more efficiently. We consider *bidirected* infinite-state systems, where for every transition there is a transition with opposite effect.

We study bidirected reachability in the framework of valence systems, an abstract model featuring finitely many control states and an infinite-state storage that is specified by a finite graph. By picking suitable graphs, valence systems can uniformly model counters as in vector addition systems, pushdowns, integer counters, and combinations thereof.

We provide a comprehensive complexity landscape for bidirected reachability and show that the complexity drops (often to polynomial time) from that of general reachability, for almost every storage mechanism where reachability is known to be decidable.

CCS CONCEPTS

• **Theory of computation → Models of computation**; **Formal languages and automata theory**; **Computational complexity and cryptography**.

KEYWORDS

Reachability, complexity, infinite-state systems, bidirected, reversible, vector addition systems, pushdown, counters

ACM Reference Format:
Moses Ganardi, Rupak Majumdar, and Georg Zetzsche. 2022. The Complexity of Bidirected Reachability in Valence Systems. In *37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS '22), August 2–5, 2022, Haifa, Israel.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3531130.3533345

1 INTRODUCTION

The reachability problem is one of the most fundamental tasks for infinite-state systems: It asks for a given infinite-state system and two configurations c_1 and c_2, whether it is possible to reach c_2 from c_1. It is a basic component in many types of verification (both for safety and liveness), and has been studied intensively for decades. Unfortunately, the reachability problem often exhibits prohibitively high complexity or is subject to long-standing open problems: In *vector addition systems with states* (VASS), which are systems with counters over the natural numbers that can be incremented and decremented, reachability is Ackermann-complete [10, 22]. Furthermore, if we allow a pushdown in addition to the counters, we obtain *pushdown vector addition systems* (PVASS), for which decidability of the reachability problem is a long-standing open problem [23].

One way to circumvent this is to consider overapproximations of the reachability problem: Instead of deciding reachability in the original system, we decide reachability in a system that has more transitions. Then, a negative answer still certifies safety. A promising overapproximation is *bidirected reachability*: We assume that in our system, for each transition, there is another with opposite effect. For example, in pushdown reachability, a push of a symbol on the forward edge is reverted by the pop of the symbol on the backward edge. In a vector addition system, incrementing a counter is reverted by decrementing it by the same amount. In addition to its theoretical interest, bidirected reachability is also of practical interest: for example, several program analysis problems can be formulated or practically approximated as bidirected pushdown reachability [7, 48] or bidirected interleaved pushdown reachability [19, 24, 25, 40, 41, 49]. A remarkable recent result is that bidirected reachability in PVASS with one counter is decidable [19].

An important measure for the utility of bidirectedness is: *Does bidirectedness reduce complexity?* It is known that bidirected reachability in VASS is EXPSPACE-complete and in logarithmic space for a fixed number of counters [32]. Bidirected pushdown reachability can be solved in almost linear time [7] whereas a truly subcubic algorithm for pushdown reachability is a long-standing open problem [8]. However, the general complexity landscape is far from understood. For example, it is hitherto not even known if bidirected pushdown reachability is P-hard. Further, it is not known whether bidirected reachability in $\mathbb{Z}$-VASS (which allow counter values to become negative) is NP-complete as for reachability [15].

Valence systems

In this paper, we systematically compare the complexity of bidirected reachability with general reachability. To this end, we use the framework of *valence systems over graph monoids* [44–47], which are an abstract model that features finitely many control states and an infinite storage specified by a finite graph (with self-loops allowed). By picking a suitable graph, one can obtain classical infinite-state models: A clique of d unlooped nodes corresponds to VASS with d counters. If the vertices are looped, one obtains $\mathbb{Z}$-VASS. Two isolated unlooped vertices yield pushdown systems.

In order to compare bidirected with general reachability, we focus on storage mechanisms where general reachability is known to be decidable. These mechanisms correspond to a well-understood class of graphs studied in [46]. The latter work characterizes a class of graphs that precisely capture PVASS with one counter. Then,

LICS '22, August 2–5, 2022, Haifa, Israel
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9351-5/22/08.
https://doi.org/10.1145/3531130.3533345

it is shown in [46] that for every graph that avoids these PVASS graphs as induced subgraphs, reachability is decidable iff the graph is a transitive forest. This class of graphs is called $\mathsf{SC}^{\pm}$.

Contribution

We provide a comprehensive complexity landscape for bidirected reachability for valence systems over graphs in $\mathsf{SC}^{\pm}$. For every graph in $\mathsf{SC}^{\pm}$, we prove that bidirected reachability is either L-complete or P-complete. More generally, we consider the setting where the graph is part of the input and drawn from a class $\mathcal{G} \subseteq \mathsf{SC}^{\pm}$: This is the case, e.g. when the input can consist of $\mathbb{Z}$-VASS with an arbitrary number of counters. Even in this general setting, we obtain an almost complete complexity picture: Then, bidirected reachability falls within five possible categories: L-complete, ILD-complete, P-complete, in EXP, or EXPSPACE-complete. Here, the complexity class ILD captures those problems that are logspace-reducible to the feasibility problem for integer linear Diophantine equations (it lies between L and P).

To obtain the P-hardness, we exploit a connection to algorithmic group theory, which allows us to reduce from the subgroup membership problem for free groups [3]. The same connection yields undecidability of bidirected reachability if the graph has an induced 4-cycle via a well-known group theoretic undecidability due to Mikhailova [34]. In particular, bidirected reachability is undecidable in systems with two pushdowns, which answers an open problem from [25][1]. This connection also yields an example of a graph where reachability is *undecidable*, but bidirected reachability is *decidable*. Finally, we can translate back to group theory: Our results imply that the subgroup membership problem in graph groups over any transitive forest is in P.

Our results show that the complexity drops for almost every storage (pure pushdowns without any additional counters are the only exceptions). For example, reachability for $\mathbb{Z}$-VASS and for pushdown $\mathbb{Z}$-VASS (which have a pushdown and several $\mathbb{Z}$-valued counters) are both known to be NP-complete [15, 17], but we place bidirected reachability in P. The same holds for $\mathbb{Z}$-VASS that, in addition, have a fixed number of $\mathbb{N}$-counters. Our results also apply to stacks where each entry contains several counter values. Moreover, in addition to such stacks, one can have $\mathbb{Z}$-counters, then build stacks of such configurations, etc. Our characterization implies that when the number of alternations between building stacks and adding $\mathbb{Z}$-counters is not fixed, then bidirected reachability can still be solved in EXP; in contrast, general reachability is NEXP-complete [16].

Key techniques

These lower complexities are achieved using novel techniques. The aforementioned connection to algorithmic group theory is the logspace inter-reducibility with the subgroup membership problem if the storage graph has self-loops everywhere. This connection between bidirected automata and subgroups has been observed in [29]; we provide logspace reductions.

While the connection to group theory is used for lower bounds, our upper bounds also require new methods. The decidability for reachability in $\mathsf{SC}^{\pm}$ in [46] employs VASS with nested zero tests [5, 36], which we manage to avoid completely. Instead, we rely on results about bidirected VASS reachability sets [20] to essentially eliminate $\mathbb{N}$-counters in our systems first. However, the main innovation is a modified approach to reachability in systems with stacks and $\mathbb{Z}$-counters [16, 17]. Those algorithms use a technique of Verma, Seidl, and Schwentick [38], which constructs, given a context-free grammar, an existential Presburger formula for its Parikh image. While existential Presburger arithmetic is equivalent to systems of integer linear *inequalities* (where feasibility is NP-complete), we show that for the new notion of *bidirected grammars*, the Parikh image can, in some appropriate sense, be described using only *equations*. This leads to a P upper bound, since feasibility of systems of integer linear equations is in P [9].

[1]This problem was answered independently in [19].

Full version

Due to space constraints, some proof details are only available in the full version of this work, which is available as [14].

ACKNOWLEDGMENTS

The authors are grateful to Markus Lohrey for discussions about the algorithm in [29]. This work was sponsered in part by the DFG under project 389792660 TRR 248–CPEC.

2 BIDIRECTED VALENCE SYSTEMS

Algebraic Preliminaries

We assume familiarity with basic notions of monoids, groups, etc. [21]. A *subgroup* of a group G is a subset of the elements of G that themselves form a group; i.e., it is a subset of elements closed under the binary operation as well as inverses. A subgroup H of a group G can be used to decompose the underlying set of G into disjoint equal-size subsets called *cosets*. The *left cosets* (resp. *right cosets*) of H in G are the sets obtained by multiplying each element of H by a fixed element g of G: $gH = \{g \cdot h \mid h \in H\}$ (resp. $Hg = \{h \cdot g \mid h \in H\}$). For a subset S, we write $\langle S \rangle$ for the smallest subgroup containing S; this is the set of all elements of G that can be written as finite products of elements from S and their inverses. If $\langle S \rangle = G$, we say S *generates* G and call the elements of S the *generators* of G.

A *presentation* $(\Sigma \mid R)$ of a monoid is a description of a monoid in terms of a set Σ of generators and a set of binary relations $R \subseteq \Sigma^* \times \Sigma^*$ on the free monoid Σ^* generated by Σ. For a set $R \subseteq \Sigma^* \times \Sigma^*$ define the step relation $\to_R$ by $sut \to_R svt$ for all $(u, v) \in R$ and $s, t \in \Sigma^*$. Define $\equiv_R$ to be the smallest equivalence relation containing $\to_R$. Then $\equiv_R$ is a *congruence*, meaning that if $u \equiv_R v$, then $sut \equiv_R svt$ for every $s, t \in \Sigma^*$. The monoid is then presented as the quotient of Σ^* by the congruence $\equiv_R$. For a word $w \in \Sigma^*$, we write $[w]_{\equiv_R}$ for the equivalence class of w under $\equiv_R$. A *commutative semigroup presentation* is a presentation $(\Sigma \mid R)$ where $(xy, yx) \in R$ for all $x \neq y \in \Sigma$. The *word problem for commutative semigroups* asks, given a commutative semigroup presentation $(\Sigma \mid R)$ and two words $u, v \in \Sigma^*$, does $u \equiv_R v$ hold? This problem is known to be EXPSPACE-complete [32].

Graph Monoids

A *graph* is a tuple $\Gamma = (V, I)$, where V is a finite set of vertices and $I \subseteq \{e \subseteq V \mid 1 \leq |e| \leq 2\}$ is a finite set of undirected edges, which can be self-loops. Thus, if $\{v\} \in I$, we say that v is *looped*; otherwise, v is *unlooped.* If all nodes of Γ are (un)looped, we call Γ *(un)looped.* By Γ° and Γ^-, we denote the graph obtained from Γ by adding (removing) self-loops on all vertices. The edge relation is also called an *independence relation.* We also write uIv for $\{u, v\} \in I$. A subset $U \subseteq V$ is a *clique* if uIv for any two distinct $u, v \in U$. We say that $U \subseteq V$ is an *anti-clique* if we do not have uIv for any distinct $u, v \in U$. Given a graph Γ, we define a monoid as follows. We define the alphabet $X_\Gamma = V \cup \bar{V}$ where $\bar{V} = \{\bar{v} \mid v \in V\}$. We define $R_\Gamma = \{(v\bar{v}, \varepsilon) \mid v \in V\} \cup \{(xy, yx) \mid x \in \{u, \bar{u}\},\ y \in \{v, \bar{v}\},\ uIv\}$. We write $\to_\Gamma$ instead of $\to_{R_\Gamma}$ and $\equiv_\Gamma$ for the smallest equivalence relation containing $\to_\Gamma$. As observed above, $\equiv_\Gamma$ is a congruence. In particular, if v has a self-loop, then $\bar{v}v \equiv_\Gamma \varepsilon$. We define the monoid $\mathbb{M}\Gamma := X_\Gamma^*/{\equiv_\Gamma}$. We write $[w]$ for the equivalence class of w under $\equiv_\Gamma$ and 1 for $[\varepsilon]$. For each word $w \in X_\Gamma^*$, we define its *inverse* $\bar{w}$ as follows. If $w = v$ for some $v \in V$, then $\bar{w}$ is the letter $\bar{v}$. If $w = \bar{v}$ for $v \in V$, then $\bar{w} = v$. Finally, if $w = w_1 \cdots w_n$ with $w_1, \ldots, w_n \in X_\Gamma$, then $\bar{w} = \bar{w}_n \cdots \bar{w}_1$. It is known that $w \equiv_\Gamma \varepsilon$ is witnessed by a derivation $w = w_0 \to_\Gamma w_1 \to_\Gamma \cdots \to_\Gamma w_n = \varepsilon$ [45, Equation (8.2)].

Valence systems and reachability

Valence systems are an abstract model for studying finite-state transition systems with "storage". They consist of a state transition system on a finite set of states, as well as a monoid that represents an auxiliary storage and determines which paths in the automata form valid computations in the presence of the auxiliary storage. For example, if the underlying storage is a stack, the monoid can encode push and pops and determine computations that produce an empty stack.

In this work, we only consider graph monoids as the underlying monoids. Many classes of infinite-state systems involving combinations of stacks and counters can be modeled as valence systems over graph monoids; see [45] for detailed examples. They have been studied in terms of expressiveness [6, 42], computing downward closures [43], and various forms of reachability problems [12, 33, 37, 46], see [47] for a survey on the latter.

A *valence system* $\mathcal{A}$ over a graph Γ consists of a finite set of states Q, and a finite transition relation $\to \subseteq Q \times X_\Gamma^* \times Q$. We also write a transition (p, u, q) as $p \xrightarrow{u} q$. A *run* is a sequence $(q_0, u_1, q_1)(q_1, u_2, q_2) \ldots (q_{n-1}, u_n, q_n)$ of transitions, also abbreviated $q_0 \xrightarrow{u} q_n$ if $u = u_1 \cdots u_n$.

The *reachability problem* (REACH) for valence systems is the following:

Given A valence system $\mathcal{A}$ and states s, t in $\mathcal{A}$.
Question Is there a run $s \xrightarrow{w} t$ for some $w \in X_\Gamma^*$ with $[w] = 1$?

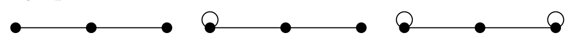
If the reachability problem is restricted to valence systems over a particular graph Γ, then we denote the problem by REACH(Γ). If we restrict the input systems to a class $\mathcal{G}$ of graphs, then we write REACH($\mathcal{G}$). For example, if $\mathcal{V}$ is the class of all unlooped cliques, then REACH($\mathcal{V}$) is the reachability problem for vector addition systems with states (VASS).

A valence system $\mathcal{A}$ is *bidirected* if for any transition $p \xrightarrow{w} q$, we also have $q \xrightarrow{\bar{w}} p$. The *bidirected reachability problem* (BIREACH) is the reachability problem where $\mathcal{A}$ is restricted to bidirected valence systems. As above, we consider the case where the system is over a particular graph Γ, denoted BIREACH(Γ), or where the graph is drawn from a class $\mathcal{G}$, denoted BIREACH($\mathcal{G}$).

2.1 Decidability Landscape for Reachability

PVASS-graphs

A graph Γ is a *PVASS-graph* if it is isomorphic to one of the following three graphs:



We say that a graph is *PVASS-free* if it has no PVASS-graph as an induced subgraph. Here, a graph $\Gamma' = (V', I')$ is an *induced subgraph* of $\Gamma = (V, I)$ if there is an injective map $\iota\colon V' \to V$ with $\{\iota(u), \iota(v)\} \in I$ iff $\{u, v\} \in I'$ for $u, v \in V'$. Observe that a graph Γ is PVASS-free if and only if in the neighborhood of each unlooped vertex, any two vertices are adjacent. The terminology stems from the fact that if Γ is a PVASS-graph, then REACH(Γ) is inter-reducible with reachability for PVASS with one counter [46]. Whether reachability is decidable for these is a long-standing open problem [23] (however, bidirected reachability is decidable [19]).

Transitive forests

A graph Γ is a *transitive forest* if it can be built as follows. First, the empty graph is a transitive forest. Moreover, if Γ_1 and Γ_2 are transitive forests, then (i) the disjoint union of Γ_1 and Γ_2 is a transitive forest and (ii) if Γ is the graph obtained by adding one looped or unlooped vertex v to Γ_1 so that v is adjacent to every vertex in Γ_1, then Γ is also a transitive forest.

The complexity of our algorithms will depend on the height of the trees in transitive forests. Formally, every non-empty transitive forest is either (i) a disjoint union of connected transitive forests, or (ii) has a *universal vertex*, i.e. a vertex that is adjacent to all other vertices (take the root of the underlying tree). This induces a successive decomposition of the transitive forest into smaller ones: For a disjoint union, take the disjoint connected transitive forests. If there is a universal vertex, remove that vertex to obtain a smaller transitive forest.

The decomposition is unique up to isomorphism: This is obvious in the case of a disjoint union. If there are several universal vertices, then all removals result in isomorphic graphs. We define the *height* $h(\Gamma)$ of a transitive forest $\Gamma = (V, I)$: If $V = \emptyset$, then $h(\Gamma) = 0$. If Γ is a disjoint union of connected transitive forests $\Gamma_1, \ldots, \Gamma_n$, then $h(\Gamma) = \max\{h(\Gamma_i) \mid i \in [1, n]\} + 1$. If Γ has a universal vertex whose removal leaves Γ', then $h(\Gamma) = h(\Gamma')$.

Among the PVASS-free graphs, it is well-understood when reachability is decidable:

Theorem 2.1 ([46]). *Let* Γ *be PVASS-free. Then* REACH(Γ) *is decidable iff* Γ *is a transitive forest.*

By SC$^\pm$, we denote the class of PVASS-free graphs Γ that are transitive forests. The abbreviation SC$^\pm$ reflects that valence systems over SC$^\pm$ are equivalent to *stacked counter machines*, as explained in Section 3. Hence, Theorem 2.1 says that for every graph in SC$^\pm$, the reachability problem is decidable. Moreover, for every graph Γ outside of $\mathsf{SC}^{\pm}$, either Γ contains a PVASS-graph (thus showing decidability of $\mathsf{REACH}(\Gamma)$ would in particular solve a long-standing open problem) or $\mathsf{REACH}(\Gamma)$ is known to be undecidable. Therefore, $\mathsf{SC}^{\pm}$ is the largest class of graphs Γ for which $\mathsf{REACH}(\Gamma)$ is currently known to be decidable.

3 MAIN RESULTS

We assume familiarity with the basic complexity classes L (deterministic logspace), P (deterministic polynomial time), NP (non-deterministic polynomial time), EXP (deterministic exponential time), NEXP (non-deterministic exponential time), and $\mathsf{EXPSPACE}$ (exponential space). By ILD, we denote the class of problems that are logspace-reducible to the problem of solvability of integer linear Diophantine equations (ILD):

Given A matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$ and a vector $\mathbf{b} \in \mathbb{Z}^m$.
Question Is there a vector $\mathbf{x} \in \mathbb{Z}^n$ with $\mathbf{Ax} = \mathbf{b}$?

It is well known that ILD is solvable in polynomial time.

Theorem 3.1 ([9]). *ILD is solvable in polynomial time.*

See [9, Theorems 1 and 13]. In particular, the class ILD lies in between L and P. The exact complexity of ILD seems to be open [1]. It is conceivable that ILD coincides with L or P or that it lies strictly in between. Hence, we have the inclusions

$$\mathsf{L} \subseteq \mathsf{ILD} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{EXP} \subseteq \mathsf{NEXP} \subseteq \mathsf{EXPSPACE}.$$

In order to formulate our main result about the complexity of $\mathsf{BIREACH}$, we need some terminology. We say that $\mathcal{G}$ is *UC-bounded* if there is a k such that for every Γ in $\mathcal{G}$, every unlooped clique in Γ has size at most k. Otherwise, it is called *UC-unbounded.* Similarly, *LC-bounded* (*LC-unbounded*, respectively) if the same condition holds for looped cliques. We say that $\mathcal{G}$ is *height-bounded* if there is a k with $h(\Gamma) \leq k$ for every Γ in $\mathcal{G}$. Otherwise, $\mathcal{G}$ is *height-unbounded.* We now present an almost complete complexity classification of $\mathsf{BIREACH}(\mathcal{G})$, where $\mathcal{G}$ is a subclass of $\mathsf{SC}^{\pm}$. Here, we assume that $\mathcal{G}$ is closed under taking induced subgraphs. This is a mild assumption that only rules out some pathological exceptions.

Theorem 3.2 (Classification Theorem). *Let $\mathcal{G} \subseteq \mathsf{SC}^{\pm}$ be closed under induced subgraphs. Then $\mathsf{BIREACH}(\mathcal{G})$ is*

(1) L-complete if $\mathcal{G}$ consists of cliques of bounded size,
(2) ILD-complete if $\mathcal{G}$ consists of cliques, is UC-bounded, and LC-unbounded,
(3) P-complete if $\mathcal{G}$ contains a graph that is not a clique, and $\mathcal{G}$ is UC-bounded and height-bounded,
(4) in EXP if $\mathcal{G}$ is UC-bounded and height-unbounded,
(5) $\mathsf{EXPSPACE}$-complete otherwise.

From Theorem 3.2, we can deduce our dichotomy for individual graphs Γ: Take as $\mathcal{G}$ the set of graphs containing Γ and its induced subgraphs. Then $\mathcal{G}$ is UC-bounded, LC-bounded, and height-bounded and thus falls into case (1) or (3) above.

Corollary 3.3 (Dichotomy for $\mathsf{BIREACH}$). *Let $\Gamma \in \mathsf{SC}^{\pm}$ be a graph. If Γ is a clique, then $\mathsf{BIREACH}(\Gamma)$ is L-complete. Otherwise, the problem $\mathsf{BIREACH}(\Gamma)$ is P-complete.*

We complement Theorem 3.2 with the following undecidability result. The graph C4 is shown in Fig. 1d.

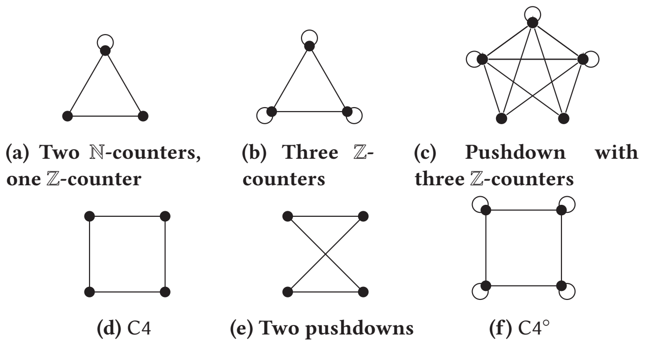

(a) **Two $\mathbb{N}$-counters, one $\mathbb{Z}$-counter** **(b)** **Three $\mathbb{Z}$-counters** **(c)** **Pushdown with three $\mathbb{Z}$-counters** **(d)** C4 **(e)** **Two pushdowns** **(f)** C4°

Figure 1: Example graphs for storage mechanisms

Theorem 3.4. *If Γ^{-} contains* C4 *as an induced subgraph, then* $\mathsf{BIREACH}(\Gamma)$ *is undecidable.*

In the case $\Gamma = \mathsf{C4}$, undecidability also follows from an independent result of Kjelstrøm and Pavlogiannis [19] that uses different techniques.

Intuition on graph classes

Let us phrase our results in terms of infinite-state systems. In (1) and (2), we have cliques. A clique with d unlooped vertices and e looped vertices corresponds to counter machines with d-many $\mathbb{N}$-counters and e-many $\mathbb{Z}$-counters (see Fig. 1a with $d = 2$, $e = 1$). Thus, (1) and (2) show L-completeness for for fixed d and e, and ILD-completeness if only d is fixed. This is in contrast to reachability: Even for fixed d, reachability in machines with d-many $\mathbb{N}$-counters can be non-elementary [10, 22]. Moreover, for $d = 0$, these graphs correspond to $\mathbb{Z}$-VASS (see Fig. 1b for $e = 3$). Thus, we show: For a fixed number of counters in $\mathbb{Z}$-VASS, the complexity drops from NL (for general reachability) to L. If the number of counters in $\mathbb{Z}$-VASS is not fixed, then we have a drop from NP [15] to $\mathsf{ILD} \subseteq \mathsf{P}$.

In (3), we go beyond just counters. Recall the recursive structure of every graph in $\mathsf{SC}^{\pm}$. In terms of storage, taking a disjoint union $\Gamma_1 \cup \Gamma_2$ is the same as *building stacks*: One obtains a stack, where each entry of the stack is either a storage content of Γ_1 or Γ_2. Moreover, adding a universal looped vertex corresponds to *adding a $\mathbb{Z}$-counter* [44, 45]: This means, in addition to the configuration of the previous model, we also have a new $\mathbb{Z}$-counter. After decomposing Γ further and further, we are left with a clique of unlooped vertices. Therefore, valence systems over $\mathsf{SC}^{\pm}$ are called *stacked counter machines*: We start with a number of $\mathbb{N}$-counters and then alternate between building stacks and adding $\mathbb{Z}$-counters. The height $h(\Gamma)$ is the number of alternations between the two steps (building stacks and adding $\mathbb{Z}$-counters).

For example, if Γ_k has two non-adjacent nodes v_1, v_2 and looped nodes $u_1, \ldots, u_k$ such that each u_i is adjacent to all other vertices (see Fig. 1c for Γ_3), then valence systems over Γ_k are pushdown $\mathbb{Z}$-VASS with k counters. Thus, (3) says that in stacked counter automata, with a bounded number of $\mathbb{N}$-counters and a bounded number of alternations (but arbitrarily many $\mathbb{Z}$-counters!), bidirected reachability is still in P. This is again a striking complexity drop: In pushdown $\mathbb{Z}$-VASS, reachability is NP-complete [17], and the same is true for any fixed number of alternations (if there are no

$\mathbb{N}$-counters) [16]. In fact, NP-hardness holds for pushdown $\mathbb{Z}$-VASS already for a single counter [16]. Moreover, our P upper bound still holds with a bounded number of $\mathbb{N}$-counters.

In (4), not even the number of alternations is fixed. We obtain an EXP upper bound, which is again a drop from reachability: In stacked counter automata (without $\mathbb{N}$-counters), reachability is NEXP-complete [16].

In (5), we show that if neither the alternations nor the number of $\mathbb{N}$-counters is fixed, BIREACH is EXPSPACE-complete. This strengthens EXPSPACE-completeness of BIREACH in VASS. Again, general reachability has much higher complexity: Since our model includes VASS, the problem is Ackermann-hard [10, 22] and could be even higher: The algorithm for reachability for general $\mathsf{SC}^{\pm}$ in [46] uses VASS with nested zero tests, for which no complexity upper bound is known.

Finally, our undecidability result (Theorem 3.4) says in particular that reachability in bidirected two-pushdown machines is undecidable: By drawing C4 as in Fig. 1e, one can see that it realizes two stacks (the left two nodes act as a stack, and the right two nodes act as a stack).

4 BIREACH AND SUBGROUP MEMBERSHIP

If Γ is looped, then $\mathbb{M}\Gamma$ is a group. The groups of this form are called *graph groups* or *right-angled Artin groups* and have been studied intensively over the last decades [11, 27, 28, 30], in part because of their rich subgroup structure (see, e.g. [39]). The *subgroup membership problem* (SUBMEM) is the following.

Given A looped graph Γ, words $w_1, \ldots, w_k, w \in X_\Gamma^*$.
Question Does $[w] \in \langle [w_1], \ldots, [w_k] \rangle$?

If the input graph Γ is fixed, we write SUBMEM(Γ). If Γ is drawn from a class $\mathcal{G}$, then we write SUBMEM($\mathcal{G}$). Surprisingly, describing the class of graphs Γ for which SUBMEM(Γ) is decidable is a longstanding open problem [26].

Our first observation is that if Γ is looped, then the complexity of BIREACH(Γ) matches that of subgroup membership over $\mathbb{M}\Gamma$. The connection between subgroups and bidirected valence automata (albeit under different names) is a prominent theme in group theory. It is implicit in the well-known concept of Stallings graphs and was used by Lohrey and Steinberg [29] in decidability results. We show that the conversion can be done in logspace, in both directions.

Theorem 4.1. *For any looped graph* Γ, *the problems* BIREACH(Γ) *and* SUBMEM(Γ) *are logspace inter-reducible.*

Reducing SUBMEM(Γ) to BIREACH(Γ) is easy: To test whether $[w]$ is contained in $\langle [w_1], \ldots, [w_k] \rangle$ we construct a bidirected valence system $\mathcal{A}$ with two states s, t and the transitions $s \xrightarrow{\bar{w}} t$ and $t \xrightarrow{w_i} t$ for all $1 \le i \le k$, and the reverse transitions. Then $[w] \in \langle [w_1], \ldots, [w_k] \rangle$ holds if and only if there exists $u \in X_\Gamma^*$ with $s \xrightarrow{u} t$ and $[u] = 1$. For the converse, the following lemma shows that we can compute in logspace a coset representation of $\{[w] \in \mathbb{M}\Gamma \mid s \xrightarrow{w} t\}$.

Lemma 4.2. *Given a looped graph* Γ, *a bidirected valence system* $\mathcal{A}$ *over* Γ *and states* s, t *from* $\mathcal{A}$, *one can compute words* $w_0, w_1, \ldots, w_n \in X_\Gamma^*$ *in logspace such that* $\{[w] \in \mathbb{M}\Gamma \mid s \xrightarrow{w} t\} = [w_0]\langle [w_1], \ldots, [w_n] \rangle$

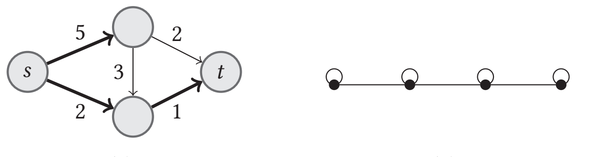


Figure 2: $\mathbb{Z}$-VASS for Example 4.3 (reverse edges are omitted) and the graph P4°

Lemma 4.2 reduces reachability to testing membership of $[\bar{w}_0]$ in the set $\langle [w_1], \ldots, [w_n] \rangle$, which is an instance of SUBMEM. To show Lemma 4.2, we compute in logspace a spanning tree of the automaton (using [35]). Then w_0 is obtained from the unique path from s to t in the tree. The words $w_1, \ldots, w_n$ are obtained as labels of *fundamental cycles* [4]: These are cycles consisting of one edge outside the tree and a path inside the tree.

Example 4.3. Lemma 4.2 can be directly used to show that the problem BIREACH($\mathcal{G}$) is in ILD if $\mathcal{G}$ consists of looped cliques. In other words, bidirectional reachability for $\mathbb{Z}$-VASS is in ILD, whereas standard reachability for $\mathbb{Z}$-VASS is NP-complete [15]. As an example, consider the bidirected $\mathbb{Z}$-VASS in Fig. 2a. We compute the value of an arbitrary s-t-path, e.g. $w_0 = 5 + 2 = 7$, a spanning tree (the bold edges), and the fundamental cycles with values $w_1 = 5 + 3 - 2 = 6$ and $w_2 = 5 + 2 - 1 - 2 = 4$. Since $7 + 6x + 4y = 0$ has no integer solution, there is no s-t-path with value 0.

The connection between BIREACH and SUBMEM can be used to identify a graph for which the problem REACH(Γ) is undecidable but BIREACH(Γ) is decidable: Let P4° be the graph in Fig. 2b. As shown by Lohrey and Steinberg [28], the problem REACH(P4°) is undecidable. However, a result by Kapovich, Weidmann, and Myasnikov [18] implies that if a graph Γ is looped and chordal, then SUBMEM(Γ) is decidable (a simpler proof was then given by Lohrey and Steinberg [29]). Thus, since P4° is looped and chordal, SUBMEM(P4°), and hence BIREACH(P4°), is decidable.

For the proof of Theorem 3.4, we rely on an undecidability result of Mikhailova [34]:

Theorem 4.4 ([34]). SUBMEM(C4°) *is undecidable.*

The graph C4° is shown in Figure 1f. The proof of Theorem 4.4 is a simple (but not obvious) reduction of the word problem of any finitely presented group (since [34] is in Russian, we refer to [31, Chapter IV, Lemma 4.2] or [30, proof of Lemma 4.2] for short expositions of this reduction). Since there are finitely presented groups with an undecidable word problem [31, Chapter IV, Theorem 7.2], this implies undecidability of SUBMEM(C4°).

Via Theorem 4.1, Theorem 4.4 implies that BIREACH(C4°) is undecidable. Using standard arguments one can transfer the undecidability of SUBMEM(C4°) to SUBMEM(C4), and thus obtain Theorem 3.4.

5 LOWER BOUNDS

We now prove the lower bounds in Theorem 3.2.

L-hardness

We can reduce from the reachability on undirected graphs to the problem BIREACH(Γ) for any Γ by replacing each undirected edge by bidirected ε-labeled transitions. Since the former problem is L-complete under AC^0 many-one reductions [2, 35], so is BIREACH(Γ).

ILD-hardness

Next, assume that $\mathcal{G}$ is LC-unbounded. Observe that ILD is the subgroup membership problem over $\mathbb{Z}^m$, where m is part of the input, and hence log-space reducible to SUBMEM($\mathcal{G}$). Since the equivalence of SUBMEM and BIREACH for looped graphs is uniform in the graph Γ (see Section 4), BIREACH($\mathcal{G}$) is ILD-hard.

P-hardness

Suppose Γ has two non-adjacent vertices u, v. In Γ°, u and v generate a free group over two generators, for which subgroup membership is P-hard [3]. By Theorem 4.1, this implies P-hardness of BIREACH(Γ°). Using standard arguments, BIREACH(Γ°) reduces to BIREACH(Γ). Thus, BIREACH(Γ) is P-hard.

EXPSPACE-hardness

We reduce from the word problem over commutative semigroups, which is EXPSPACE-hard [32]. Since $\mathcal{G}$ is UC-unbounded and closed under induced subgraphs, it contains an unlooped clique Γ of size $|\Sigma|$. We can assume that Σ is its node set. Let $\mathcal{A}$ be the bidirected valence system over Γ with three states q_0, q_1, q_2, the transitions $q_0 \xrightarrow{\bar{u}} q_1$, $q_1 \xrightarrow{v} q_2$, and the transitions $q \xrightarrow{\bar{x}y} q$ for all $(x, y) \in R$, and their reverse transitions. Then $u \equiv_R v$ holds if and only if $q_0 \xrightarrow{w} q_2$ for some $w \in X_\Gamma^*$ with $[w] = 1$.

6 UPPER BOUNDS I: L AND ILD

In this section we will study BIREACH($\mathcal{G}$) for classes $\mathcal{G}$ of cliques, and prove the L and ILD upper bounds from Theorem 3.2. If Γ is an unlooped clique then BIREACH(Γ) is the reachability problem over *bidirected vector addition systems with states* or, equivalently, the word problem for commutative semigroups [32].

Fix a bidirected valence system $\mathcal{A} = (Q, \to)$ over a clique $\Gamma = (V, I)$. Let U and L be the sets of unlooped and looped vertices in Γ, respectively, and let $s, t \in Q$. We can view the unlooped vertices as $\mathbb{N}$-counters, which may not fall below zero, and the looped vertices as $\mathbb{Z}$-counters. Formally, the monoid $\mathbb{M}\Gamma$ is isomorphic to $\mathbb{B}^U \times \mathbb{Z}^L$ where $\mathbb{B}$ is the *bicyclic monoid*, i.e. the set $\mathbb{N}^2$ equipped with the associative operation $(a^-, a^+) \oplus (b^-, b^+) = (a^- + b^- - \min(a^+, b^-), a^+ + b^+ - \min(a^+, b^-))$. We identify each element $(0, a^+)$ in $\mathbb{B}$ with $a^+ \in \mathbb{N}$ and each $(a^-, 0)$ with $-a^- \in -\mathbb{N}$. Let $\Phi\colon X_\Gamma^* \to \mathbb{Z}^L$ be the function which computes the value of the $\mathbb{Z}$-counters, i.e. $\Phi(w)(v) = |w|_v - |w|_{\bar{v}}$ for all $w \in X_\Gamma^*$, $v \in L$. Let $\Psi\colon X_\Gamma^* \to \mathbb{B}^U$ be the morphism defined by $\Psi(v)(v) = (0, 1)$ and $\Psi(\bar{v})(v) = (1, 0)$ for all $v \in U$, and $\Psi(x)(v) = (0, 0)$ for all $x \in (V \cup \bar{V}) \setminus \{v, \bar{v}\}$. Then $\mathbb{M}\Gamma \to \mathbb{B}^U \times \mathbb{Z}^L$, $[w] \mapsto (\Psi(w), \Phi(w))$ for $w \in X_\Gamma^*$, is an isomorphism.

$\mathbb{N}$-counters

If we consider only $\mathbb{N}$-counters, i.e. $\mathcal{A}$ is a bidirected VASS, the reachability problem and the structure of reachability sets are well understood:

Lemma 6.1 ([32]). *One can decide in deterministic space $2^{O(|U|)} \cdot \log \|\mathcal{A}\|$ whether there exists a path $s \xrightarrow{w} t$ with $\Psi(w) = \mathbf{0}$ and, if so, return such a path.*

We define $\mathsf{Reach}(p, q) = \{\Psi(w) \mid p \xrightarrow{w}_{\mathcal{A}} q\} \cap \mathbb{N}^U$ for any states $p, q \in Q$. Lemma 6.2 lets us partition the set U of $\mathbb{N}$-counters into (exponentially) bounded components B and *simultaneously* unbounded components $U \setminus B$. In the proof, we employ a representation of $\mathsf{Reach}(p, q)$ as a *hybrid linear set* $\bigcup_{i=1}^m \{\mathbf{b}_i + \sum_{j=1}^\ell \lambda_j \mathbf{p}_j \mid \lambda_1, \dots, \lambda_\ell \in \mathbb{N}\}$ with $\mathbf{b}_i, \mathbf{p}_j \in \mathbb{N}^\Sigma$ as shown in [20]. See our full version [14] for details.

Lemma 6.2. *One can compute in deterministic space $2^{O(|U|)} \cdot \log \|\mathcal{A}\|$ a set $B \subseteq U$ and a number $b \le 2^{O(|U|)} \cdot \|\mathcal{A}\|$ such that for all $q \in Q$ we have:*

- *$\mathbf{v}(u) \le b$ for all $\mathbf{v} \in \mathsf{Reach}(s, q)$ and $u \in B$,*
- *for every $c \in \mathbb{N}$ there exists $\mathbf{v} \in \mathsf{Reach}(s, q)$ with $\mathbf{v}(u) \ge c$ for all $u \in U \setminus B$.*

Adding $\mathbb{Z}$-counters

We will decide reachability in $\mathcal{A}$ by computing a representation for

$$\mathsf{Eff}(s, t) = \{\Phi(w) \mid s \xrightarrow{w}_{\mathcal{A}} t,\ \Psi(w) = \mathbf{0}\}.$$

Since $[w] = 1$ if and only if $\Psi(w) = \mathbf{0}$ and $\Phi(w) = \mathbf{0}$ we only need to test $\mathbf{0} \in \mathsf{Eff}(s, t)$. Observe that $\mathsf{Eff}(s, t)$ is either empty or a coset $\mathsf{Eff}(s, t) = \mathbf{u} + \mathsf{Eff}(s, s)$ for any $\mathbf{u} \in \mathsf{Eff}(s, t)$. Using Lemma 6.1 we can test whether there is a path $s \xrightarrow{w} t$ with $\Psi(w) = \mathbf{0}$, and, if so, we find $\mathbf{u} := \Phi(w) \in \mathsf{Eff}(s, t)$. It remains to compute a representation of the subgroup $\mathsf{Eff}(s, s)$.

Proposition 6.3. *In deterministic space $2^{O(|U|)} \cdot \log \|\mathcal{A}\|$, we can compute $L' \supseteq L$, $|L'| \le |\Gamma|$, and $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{Z}^{L'}$ with $\mathsf{Eff}(s, s) = \{\mathbf{v}|_L \mid \mathbf{v} \in \langle \mathbf{v}_1, \dots, \mathbf{v}_n \rangle,\ \mathbf{v}|_{L' \setminus L} = \mathbf{0}\}$.*

Proof Sketch. We eliminate the $\mathbb{N}$-counters in $\mathcal{A}$ by replacing the unbounded counters by $\mathbb{Z}$-counters and maintaining the bounded ones in the finite state. We obtain a system $\mathcal{A}'$ over a looped clique, for which the statement follows from Lemma 4.2. Clearly, every valid $\mathcal{A}$-run translates into a valid $\mathcal{A}'$-run. However, in a valid $\mathcal{A}'$-run, the formerly unbounded counters can now take negative values. We can prepend a cycle run which takes sufficiently large values in the unbounded components, and append the reverse cycle run to cancel its effect. This ensures that all counters remain nonnegative during the run, which can thus be translated into an $\mathcal{A}$-run. ◻

We can now prove the upper bounds for cases (1) and (2) in Theorem 3.2. As explained above we can test in exponential space (logarithmic space if $\mathcal{G}$ consists of cliques of bounded size) whether $\mathsf{Eff}(s, t)$ is nonempty and, if so compute a vector $\mathbf{u} \in \mathsf{Eff}(s, t)$. It remains to test $\mathbf{0} \in \mathsf{Eff}(s, t)$, which is equivalent to $-\mathbf{u} \in \mathsf{Eff}(s, s)$. Using Proposition 6.3, this can be solved in ILD. If $|\Gamma|$ is bounded then this is in L (even TC^0) [13, Theorem 13].

Proposition 6.4. *If $\mathcal{G}$ is a UC-bounded class of cliques, then the problem* BIREACH($\mathcal{G}$) *belongs to* ILD. *If $\mathcal{G}$ is a class of cliques of bounded size, then* BIREACH($\mathcal{G}$) *belongs to* L.

Similarly we can prove the following result, which will be used in Section 7.

Theorem 6.5. *Given a clique $\Gamma \in \mathcal{G}$, a bidirected valence system $\mathcal{A} = (Q, \to)$ over Γ, and states $s, t \in Q$, one can test in exponential space (polynomial time if $\mathcal{G}$ is UC-bounded) if* $\mathsf{Eff}(s,t)$ *is nonempty and, if so, compute a coset representation* $\mathbf{u} + \langle \mathbf{v}_1, \ldots, \mathbf{v}_n \rangle$ *for* $\mathsf{Eff}(s,t)$.

7 UPPER BOUNDS II: P, EXP, EXPSPACE

We now prove the upper bounds of (3) and (4) in Theorem 3.2. Let $\mathsf{SC}^{\pm}_{d}$ be the class of graphs in $\mathsf{SC}^{\pm}$ where each unlooped clique has size at most d, and let $\mathsf{SC}^{\pm}_{d,\ell}$ be the class of those graphs Γ in $\mathsf{SC}^{\pm}_{d}$ with $h(\Gamma) \leq \ell$. We prove that (i) BIREACH($\mathsf{SC}^{\pm}$) is in EXPSPACE, (ii) for every d, BIREACH($\mathsf{SC}^{\pm}_{d}$) is in EXP and (iii) for every $d, \ell \geq 0$, the problem BIREACH($\mathsf{SC}^{\pm}_{d,\ell}$) is in P.

Key ideas and outline

The graphs in $\mathsf{SC}^{\pm}_{d,\ell}$ correspond to the following storage mechanisms. The simplest case, graphs in $\mathsf{SC}^{\pm}_{d,0}$, consist in collections of d counters with values in $\mathbb{N}$, or d-VASS. The storage mechanism corresponding to $\mathsf{SC}^{\pm}_{d,1}$ is a stack, where each entry contains d such $\mathbb{N}$-counters. In addition, they have $\mathbb{Z}$-counters in parallel to such stacks (note that using more $\mathbb{N}$-counters in addition to a stack would be a PVASS, which we are avoiding). Thus, $\mathsf{SC}^{\pm}_{d,1}$ corresponds to "stacks of d $\mathbb{N}$-counters, plus $\mathbb{Z}$-counters". The *building stacks* and *adding $\mathbb{Z}$-counters* can now be iterated to obtain the storage mechanisms for $\mathsf{SC}^{\pm}_{d,\ell}$ for higher ℓ: A storage mechanism with stacks, where each entry is a configuration of an $\mathsf{SC}^{\pm}_{d,1}$ storage, plus additional $\mathbb{Z}$-counters, is one in $\mathsf{SC}^{\pm}_{d,2}$, etc.

Our algorithms work as follows. We first translate a valence system over $\mathsf{SC}^{\pm}_{d,\ell}$ into a type of grammar that we call k-grammar. This translation is quite similar to existing approaches to show that general reachability over $\mathsf{SC}^{\pm}_{1,\ell}$ is decidable [28] and even NP-complete [16]. The difference to the previous approaches is that for *bidirected* valence systems, we can construct *bidirected k-grammars*, which are k-grammars that satisfy a carefully chosen set of symmetry conditions. In this translation, we also eliminate the d $\mathbb{N}$-counters: Roughly speaking, Theorem 6.5 lets us replace them by a gadget that preserves their interaction with the $\mathbb{Z}$-counters.

After the reduction to grammars, the algorithm in [16] (and also the NP algorithm for pushdown $\mathbb{Z}$-VASS [17], which correspond to $\mathsf{SC}^{\pm}_{0,1}$) employs a result by Verma, Seidl, and Schwentick [38]. It says that given a context-free grammar, one can construct in polynomial time an existential Presburger formula for its Parikh image. Existential Presburger arithmetic corresponds to systems of integer linear *inequalities*, for which feasibility is NP-complete. (To be precise, [16] uses Presburger arithmetic extended with Kleene stars, to deal with $\mathsf{SC}^{\pm}_{0,\ell}$ for $\ell > 1$.) We show that for *bidirected* grammars, we have an analogous result that yields systems of linear integer *equations*, which can be solved in P.

This analogue of [38] lets us express emptiness of bidirected grammars using *coset circuits*, which are (newly introduced) circuits to describe cosets $C \subseteq \mathbb{Z}^m$. They can be seen as compressed representations of integer linear equation systems. In the final step, we observe that those coset circuits translate into exponential-sized linear equation systems. Moreover, for every fixed ℓ, on input from $\mathsf{SC}^{\pm}_{d,\ell}$, the coset circuits have bounded depth, resulting in polynomial-sized equation systems.

Grammars

Let us now define k-grammars. They have a set N of nonterminal symbols (which can be rewritten by a grammar rule) and a set T of terminal symbols (which can not be rewritten). We allow letters in T to occur negatively, but the letters in N can only occur non-negatively. Hence, we derive vectors in $\mathbb{N}^N + \mathbb{Z}^T$, or equivalently, vectors $\mathbf{u} \in \mathbb{Z}^{N \cup T}$, where $\mathbf{u}(a) \geq 0$ for each $a \in N$. We say that a vector $\mathbf{v} \in \mathbb{N}^N$ *occurs* in such a $\mathbf{u}$ if $\mathbf{u}(a) \geq \mathbf{v}(a)$ for every $a \in N$.

Formal definition. A *k-grammar* is a tuple $G = (N, T, P)$, where

- N is a finite alphabet of *nonterminals*, which is a disjoint union $N = \bigcup_{i=0}^{k} N_i$,
- T is a finite alphabet of *terminals*, which is a disjoint union $T = \bigcup_{i=0}^{k} T_i$,
- P is a finite set of *productions* of one of two forms:
  - $a \to b$ with $a \in N_i$, $b \in N_j$, $|i - j| = 1$.
  - $a \to \mathbf{u}$ with $a \in N_0$ and $\mathbf{u} \in \mathbb{N}^{N_0} + \mathbb{Z}^T$.

The letters in T_i (N_i) are the *level-i (non)terminals.* We write

$$N_{[i,j]} = \bigcup_{i \leq r \leq j} N_r,$$

and analogously for $T_{[i,j]}$. Define $R \subseteq N$ as the subset of $a \in N$ that appear on some right-hand side of the grammar. We also write $R_i = R \cap N_i$ and use the notation $R_{[i,j]}$ as for N and T.

Derivations. In these grammars, derivations produce vectors in $\mathbb{N}^N + \mathbb{Z}^T$ instead of words.[2] A *configuration* is a vector $\mathbf{v} \in \mathbb{N}^N + \mathbb{Z}^T$. For $i \in [0, k]$, we define the i-derivation relation $\Rightarrow_i$ as follows. We begin with defining $\Rightarrow_0$ and then define each $\Rightarrow_i$ based on $\Rightarrow_{i-1}$. For configurations $\mathbf{v}, \mathbf{v}' \in \mathbb{N}^N + \mathbb{Z}^T$, we have $\mathbf{v} \Rightarrow_0 \mathbf{v}'$ if there is some $a \in N_0$ and a production $a \to \mathbf{u}$ with $\mathbf{u} \in \mathbb{Z}^{N_0 \cup T}$ or $\mathbf{u} \in N_1$ such that $\mathbf{v}(a) > 0$ and $\mathbf{v}' = \mathbf{v} - a + \mathbf{u}$.

In order to define $\Rightarrow_i$ inductively for $i > 0$, we first need to define generated sets. Let $\overset{*}{\Rightarrow}_i$ denote the reflexive transitive closure of $\Rightarrow_i$. We define the generated set $L(a)$ for each $a \in N_i$:

$$L(a) = \{\mathbf{u} \in \mathbb{N}^{N_{[i+1,k]}} + \mathbb{Z}^{T_{[i+1,k]}} \mid a \overset{*}{\Rightarrow}_i \mathbf{u}\}.$$

We define $\Rightarrow_i$ based on $\Rightarrow_{i-1}$: We have $\mathbf{v} \Rightarrow_i \mathbf{v}'$ if there is an $a \in N_i$ with $\mathbf{v}(a) > 0$ and a production $a \to a'$ for some $a' \in N_{i-1}$ and a $\mathbf{u} \in \mathbb{Z}^{N \cup T}$ with $\mathbf{u} \in L(a')$ and $\mathbf{v}' = \mathbf{v} - a + \mathbf{u}$. If $a \to b$ is a production with $a \in N_i$ and $b \in N_{i+1}$ and $\mathbf{v}(a) > 0$ and $\mathbf{v}' = \mathbf{v} - a + b$, then we also have $\mathbf{v} \Rightarrow_i \mathbf{v}'$. The *emptiness problem* asks, given a k-grammar G and a nonterminal a of G, is $L(a) \neq \emptyset$?

[2] Usually, grammars derive strings rather than vectors. We could also develop our theory using grammars that generate strings, but since we are only interested in the generated strings up to reordering of letters, we simplify the exposition by working directly with vectors.

Intuition. We give some intuition on how REACH($\mathsf{SC}^{\pm}_{0,\ell}$) is translated into a grammar. Each nonterminal is of the form (p, s^x, q), where p, q are states of the valence system and s^x specifies some part of the storage mechanism (see the full version [14] for details). Then (p, s^x, q) represents the set of all runs of the system from p to q such that on part s^x, the run has overall neutral effect.

Specifically, in the case of $\mathsf{SC}^{\pm}_{0,1}$, i.e. pushdown $\mathbb{Z}$-VASS, reachability amounts to checking whether a context-free language contains a word whose letter counts satisfy some linear condition. In a 0-grammar, we have context-free rewriting rules ($a \to \mathbf{u}$), which involve nonterminals (N) and terminals (T). In addition, the definition of $L(a)$ requires that a derived contains no terminal from T_0 anymore: This is used to implement the linear condition on letter counts for pushdown $\mathbb{Z}$-VASS.

For $\mathsf{SC}^{\pm}_{0,\ell}$, $\ell > 1$, we need to simulate stacks with $\mathsf{SC}^{\pm}_{0,\ell-1}$ configurations in each stack entry. Here, we can use a higher k: A derivation step at level k involves an entire derivation at level $k-1$. This corresponds to the fact that between a pair of push and pop of a $\mathsf{SC}^{\pm}_{d,\ell}$ machine, there is an entire run of an $\mathsf{SC}^{\pm}_{d,\ell-1}$ machine.

Bidirected grammars

We are now ready to present the symmetry conditions of bidirected grammars. For a k-grammar G, an *involution* is a map $\cdot^{\dagger} \colon N \to N$ such that for $a \in N_i$, we have $a^{\dagger} \in N_i$ for $i \in [0,k]$ and $(a^{\dagger})^{\dagger} = a$. Then, for $\mathbf{u} \in \mathbb{Z}^{N \cup T}$, we define $\mathbf{u}^{\dagger} \in \mathbb{Z}^{N \cup T}$ as $\mathbf{u}^{\dagger}(a) = \mathbf{u}(a^{\dagger})$ for $a \in N$ and $\mathbf{u}^{\dagger}(a) = -\mathbf{u}(a)$ for $a \in T$. Here, $\mathbf{u}^{\dagger}$ can be thought of as the inverse of $\mathbf{u}$.

We say that a k-grammar G is *bidirected* if there is an involution $\cdot^{\dagger} \colon N \to N$ such that

(1) for every production $a \to \mathbf{u}$ in P, we have a production $a^{\dagger} \to \mathbf{u}^{\dagger}$ in P,
(2) for every $a \in R_0$ and every production $a \to b + \mathbf{u}$ with $b \in R_0$, $\mathbf{u} \in \mathbb{N}^{R_0} + \mathbb{Z}^{T}$, we have $b \overset{*}{\Rightarrow}_0 a + \mathbf{u}^{\dagger}$,
(3) for every production $a \to b$ with $a \in N_i$, $b \in N_j$ with $|i-j| = 1$ and $a \in R$, we also have $b \to a$,
(4) for every $a \in R$, we have $L(a) \neq \emptyset$,
(5) if $a \in R_0$, then $a \overset{*}{\Rightarrow}_0 a + a + a^{\dagger}$.

Intuition on bidirectedness. We give some intuition on these symmetry conditions. Recall that in the translation from automata to grammars, each nonterminal is of the form (p, s^x, q), where p, q are states and s^x represents part of the storage mechanism. The involution is given by $(p, s^x, q)^{\dagger} = (q, s^x, p)$. Since the map $\cdot^{\dagger}$ negates terminal letters, which represent counter values, condition (1) reflects the existence of paths that go in the opposite direction with opposite effect.

Conditions (2),(3) let us reverse productions: Consider a production $(p, s^x, q) \to (p, s^x, t) + (t, s^x, q)$. It says one can get a run from p to q by combining one from p to t with one from t to q. Now (2) yields a derivation $(p, s^x, t) \overset{*}{\Rightarrow}_0 (p, s^x, q) + (t, s^x, q)^{\dagger} = (p, s^x, q) + (q, s^x, t)$. This reflects that paths from p to t can be obtained from ones from p to q and ones from q to t.

Cross-level productions $a \to b$ with $a \in N_i$, $b \in N_j$ with $|i-j| = 1$ describe the relationship between comparable parts s^x of the storage. Here, "comparable" means "a subset of the counters." For example, if r^y represents a subset of the counters in s^x, we have productions $(p, s^x, q) \to (p, r^y, q)$, which tell us: A run ρ from p to q that is neutral on r^y is also neutral on s^x *if* ρ's effect on the additional counters in s^x is zero (recall that a derivation on level $i-1$ can only be used as a step on level i if its effect on the terminal letters is zero). Now (3) says that this is symmetric: The production $(p, r^y, q) \to (p, s^x, q)$ tells us that any run that is even neutral on s^x is in particular neutral on r^y.

The last two conditions (4) and (5) stem from the fact that we sometimes construct derivations that, as a byproduct, create vectors $a + a^{\dagger} = (p, s^x, q) + (q, s^x, p)$. On the one hand, we want to argue that such cycles can always be eliminated by further derivation. This is guaranteed by (4), which lets us derive some vector $\mathbf{u}$ from a and then because of (2), the inverse $\mathbf{u}^{\dagger}$ from $a^{\dagger}$. This results in the vector $\mathbf{u} + \mathbf{u}^{\dagger}$, but the nonterminals in $\mathbf{u}$ are on a higher level than a. Thus, inside of a derivation on level k, we can completely get rid of it. Finally, (5) complements this by letting us create such cycles. This simplifies the set of derived vectors.

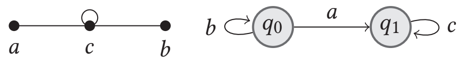
Constructing bidirected grammars. We obtain the following reduction. It is technically involved, but since it follows similar ideas to existing approaches ([16, 28]), we defer details to the full version [14].

Proposition 7.1. *There is a polynomial-time Turing reduction from* BIREACH($\mathsf{SC}^{\pm}_{d}$) *to the emptiness problem for bidirected k-grammars. If the input graphs are from $\mathsf{SC}^{\pm}_{d,\ell}$, then we only use k-grammars with $k \leq 2\ell$.*

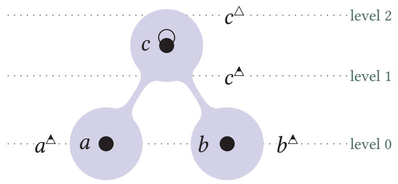
Note that the reduction described in Proposition 7.1 is a Turing reduction. This is because we need to ensure (4). To this end, the reduction involves a saturation that successively enlarges the set of nonterminals that are known to generate a non-empty set: At first, it only allows a small set of triples $(p, s^x, q) \in N$, whose language is non-empty by construction, to appear on right-hand sides. It then invokes the emptiness check, which yields more triples (nonterminals) that can then appear on right-hand sides in the next iteration, etc.

Example 7.2. Let us see Proposition 7.1 in an example. Consider the graph Γ in Fig. 3a and the valence system in Fig. 3b. Observe that Γ corresponds to a pushdown $\mathbb{Z}$-VASS: The nodes a, b realize a stack, and c acts as a $\mathbb{Z}$-counter. For simplicity, we show the translation in the final step of the saturation (described after Proposition 7.1), i.e. where the set of triples $(p, s^x, q) \in N$ with $L((p, s^x, q)) \neq \emptyset$ has stabilized. The entry s^x in the nonterminals represents to a subset of the nodes in the graph: First, $a^{\blacktriangle}$ and $b^{\blacktriangle}$ represent only the node a and b, respectively. Second, $c^{\blacktriangle}$ represents the set $\{a, b\}$. Third, $c^{\triangle}$ represents $\{a, b, c\}$. This decomposition into subsets is derived from the tree structure of Γ as a transitive forest, see Fig. 3c. The decomposition also determines the nonterminal levels: We have $N_0 = \{(p, s^x, q) \mid s^x \in \{a^{\blacktriangle}, b^{\blacktriangle}\}\}$, $N_1 = \{(p, s^x, q) \mid s^x = c^{\blacktriangle}\}$ and $N_2 = \{(p, s^x, q) \mid s^x = c^{\triangle}\}$. The terminal letters correspond to looped nodes, and their levels also stem from the decomposition into a tree, i.e. $T_0 = T_1 = \emptyset$ and $T_2 = \{c\}$.

Intuitively, a nonterminal (p, s^x, q) represents runs of the valence system in which only transitions over nodes (in Γ) in s^x and its ancestors in the tree are used and where the effect on the storage is neutral w.r.t. nodes in s^x. However, the derived multisets of (p, s^x, q)



(a) The graph Γ **(b) Valence system over Γ (reverse edges are not shown)**



(c) Decomposition of Γ into a tree.

Figure 3: Graph and example valence system

can also contain nonterminals (p', r^y, q') as "placeholders", where r^y is above s^x in the tree decomposition. This allows derivations at a higher level to insert neutral runs into each other to simulate runs that are neutral on the union of two subtrees.

Note that $(q_0, u^{\blacktriangle}, q_1)$ and $(q_1, u^{\blacktriangle}, q_0)$ have empty languages for $u \in \{a, b\}$, because there is no run between q_0 and q_1 with a neutral effect w.r.t a and $\bar{a}$. Therefore, we will not use these nonterminals on the right-hand side. However, all other nonterminals (p, s^x, q) have non-empty languages. Thus, the set $R \subseteq N$ contains all nonterminals (p, s^x, q) with $p = q$.

We begin with productions for nonterminals $(p, a^{\blacktriangle}, p)$:

$$(p, u^{\blacktriangle}, p) \to (p, u^{\blacktriangle}, p) + (p, u^{\blacktriangle}, p) \qquad p \in \{q_0, q_1\}, u \in \{a, b\}$$
$$(p, a^{\blacktriangle}, p) \to (q, a^{\blacktriangle}, q) \qquad p, q \in \{q_0, q_1\}$$

Note that the second production stems from the fact that we can start from q_0, then move to q_1 with a, then execute a loop from q to q, and then go back to q_0 with $\bar{a}$. We also have productions for the c and $-c$ loops at q_1:

$$(q_1, u^{\blacktriangle}, q_1) \to c, \qquad (q_1, u^{\blacktriangle}, q_1) \to -c \qquad \text{for } u \in \{a, b\}$$

Finally, we have cross-level productions:

$$(p, c^{\blacktriangle}, p) \to (p, u^{\blacktriangle}, p) \qquad (p, u^{\blacktriangle}, p) \to (p, c^{\blacktriangle}, p) \qquad \text{(level 0/1)}$$
$$(p, c^{\triangle}, p) \to (p, c^{\blacktriangle}, p) \qquad (p, c^{\blacktriangle}, p) \to (p, c^{\triangle}, p) \qquad \text{(level 1/2)}$$

for every $p \in \{q_0, q_1\}$ and $u \in \{a, b\}$.

Reversing derivations on each level

We will show an analogue of (2) at levels $i > 0$, by induction on i. Since a step on level $i > 0$ involves an entire derivation on level $i-1$ (which can result in several nonterminals), the induction works with a stronger property where the reverse derivation can not only start in one produced nonterminal (as in (2)), but arbitrary vectors in $\mathbb{N}^{R_i}$. But then, suppose we want to reverse

$$(p, s^x, q) \Rightarrow_0 (p, s^x, t) + (t, s^x, q).$$

Deriving (p, s^x, q) from $(p, s^x, t) + (t, s^x, q)$ would require either (p, s^x, t) or (t, s^x, q) to derive a vector without nonterminals, which would be too strong a condition. However, we can use a slight relaxation: We will show that derivations can be reversed *up to a byproduct of cycles* $b + b^{\dagger}$. For instance, in our example, the condition (2) yields $(p, s^x, t) \overset{*}{\Rightarrow}_0 (p, s^x, q) + (t, s^x, q)^{\dagger}$ and hence

$$(p, s^x, t) + (t, s^x, q) \overset{*}{\Rightarrow}_0 (p, s^x, q) + (t, s^x, q)^{\dagger} + (t, s^x, q).$$

To make "up to cycles $b + b^{\dagger}$" precise, we now introduce the equivalence relations $\approx_a$.

First, we define the relation $\rightsquigarrow$ on N as follows. For $a \in N_i$ and $b \in N_{[i,k]}$, we have $a \rightsquigarrow b$ iff there is a configuration $\mathbf{u} \in \mathbb{N}^{N_{[i,k]}} + \mathbb{Z}^{T_{[i,k]}}$ with $\mathbf{u}(b) \geq 1$ and $a \Rightarrow_i \mathbf{u}$. In other words, for $a \in N_i$ and $b \in N_{[i,k]}$, we have $a \rightsquigarrow b$ if b can be reached from a in one derivation step on level i.

Let $\overset{*}{\rightsquigarrow}$ denote the reflexive transitive closure of $\rightsquigarrow$. For each $a \in N_i$, we define the monoid Δ_a, which is generated by all $b + b^{\dagger}$ with $b \in N_{[i+1,k]}$ and $a \overset{*}{\rightsquigarrow} b$. Hence, $\Delta_a \subseteq \mathbb{N}^{N_{[i+1,k]}}$. For $\mathbf{u}, \mathbf{u}' \in \mathbb{N}^{N_{[i,k]}} + \mathbb{Z}^{T_{[i,k]}}$, we have $\mathbf{u} \approx_a \mathbf{u}'$ if and only if there are $\mathbf{d}, \mathbf{d}' \in \Delta_a$ with $\mathbf{u} + \mathbf{d} = \mathbf{u}' + \mathbf{d}'$. A k-grammar is called *i-bidirected* if for every $a \in R_i$ and every derivation $a \overset{*}{\Rightarrow}_i \mathbf{u} + \mathbf{v}$ with $\mathbf{u} \in \mathbb{N}^{R_i}$, $\mathbf{u} \neq \mathbf{0}$, $\mathbf{v} \in \mathbb{N}^{R_{[i,k]}} + \mathbb{Z}^{T_{[i,k]}}$, then $\mathbf{u} \overset{*}{\Rightarrow}_i a + \mathbf{v}'$ for some $\mathbf{v}'$ with $\mathbf{v}' \approx_a \mathbf{v}^{\dagger}$. In short, up to differences in Δ_a, we can reverse derivations on level i that start in nonterminals in R. Note that if G is i-bidirected, then on R_i, the relation $\overset{*}{\rightsquigarrow}$ is symmetric and thus an equivalence. We show the following:

Lemma 7.3. *If G is bidirected, then it is i-bidirected for each $i \in [0, k]$.*

Expressing emptiness using cosets

An important ingredient in our proof is to work with integer linear *equations* instead of inequalities. While solution sets of systems of inequalities are semilinear sets in $\mathbb{Z}^m$, solution sets of equation systems are cosets of $\mathbb{Z}^m$. Recall that a coset in $\mathbb{Z}^m$ is a set of the form $\mathbf{u} + U$, where $\mathbf{u} \in \mathbb{Z}^m$ and $U \subseteq \mathbb{Z}^m$ is a subgroup. In this section, we will translate emptiness in bidirected grammars into a problem about cosets of $\mathbb{Z}^{N \cup T}$. In general, we employ three operations for constructing cosets. First, we can take $\mathbf{u} + \langle S \rangle$ for $\mathbf{u} \in \mathbb{Z}^{N \cup T}$ and any set $S \subseteq \mathbb{Z}^{N \cup T}$. Recall that $\langle S \rangle$ is the subgroup generated by S. Second, if $C_1, C_2 \subseteq \mathbb{Z}^{N \cup T}$ are cosets, then their point-wise sum $C_1 + C_2$ is a coset, and the set $C_1 \cap C_2$ is either a coset or empty.

A central role will be played by the group H_a, which we define for each $a \in R_i$:

$$H_a = \langle -b + \mathbf{u} \mid b \in R_i \text{ and } b \Rightarrow_i \mathbf{u} \text{ and } a \overset{*}{\rightsquigarrow} b \rangle \qquad (1)$$

Hence, H_a is the group generated by all differences that are added when applying derivation steps $b \Rightarrow_i \mathbf{u}$ for $a \overset{*}{\rightsquigarrow} b$. We observe that if $a \overset{*}{\Rightarrow}_i \mathbf{u}$, then $-a + \mathbf{u} \in H_a$. We will need a coset to express that in such a derivation, there are no level-i letters (resp. no level-i terminals) left. We will do this by intersecting with the cosets

$$S_i = \mathbb{Z}^{N_{[i+1,k]} \cup T_{[i+1,k]}}, \quad S_i' = \mathbb{Z}^{N_{[i+1,k]}} + \mathbb{Z}^{T_{[i,k]}}.$$

Using H_a and S_i, we can now define the coset that will essentially characterize $L(a)$. For $a \in R_i$, we set

$$L_a = (a + H_a) \cap S_i. \qquad (2)$$

One of the main results of this section will be that for $a \in R_i$, we can describe $L(a)$ in terms of L_a. Following the theme that we can do things only "up to differences in Δ_a" on each level, we need a group version of Δ_a. For every $a \in N_i$, we have the subgroup $D_a \subseteq \mathbb{Z}^{N_{i+1}}$, which we define next. If $a \in N_i$ with $i \in [0, k-1]$, we set $D_a = \langle b + b^\dagger \mid b \in N_{i+1},\ a \overset{*}{\rightsquigarrow} b \rangle$. For $a \in N_k$, we define $D_a = \{\mathbf{0}\}$. In other words, D_a is the group generated by Δ_a, for every $a \in N$. With this notation, one of the crucial ingredients in our proof is to show that $L_a = L(a) + D_a$, i.e. L_a describes $L(a)$ up to differences in D_a (see Theorem 7.4). By our observation about H_a, it is obvious that $L(a) + D_a \subseteq L_a$. The key step is that also $L_a \subseteq L(a) + D_a$.

While the cosets L_a are coset versions of the sets $L(a)$, we will also need coset versions of a slight variant of $L(a)$. For each $a \in R_i$, by $M(a)$, we denote the vectors that are derivable from a, but may still contain level-i terminals:

$$M(a) = \{\mathbf{u} \in \mathbb{N}^{N_{[i+1,k]}} + \mathbb{Z}^{T_{[i,k]}} \mid a \overset{*}{\Rightarrow}_i \mathbf{u}\}.$$

Thus, $M(a)$ differs from $L(a)$ by collecting all derivable $\mathbf{u} \in \mathbb{N}^{N_{[i+1,k]}} + \mathbb{Z}^T$, where all level-$i$ nonterminals have been eliminated, but not necessarily all level-i terminals, meaning $L(a) = M(a) \cap S_i$. Just as L_a is a coset analogue of $L(a)$, we have a coset analogue of $M(a)$:

$$M_a = (a + H_a) \cap S'_i.$$

The cosets M_a will be needed to express, using cosets, whether $L(a)$ is empty for nonterminals $a \in N_i$ that do not necessarily belong to R_i: Those are the nonterminals for which we do not know whether $L(a)$ is empty. Such sets $L(a)$ do not directly correspond to cosets. However, we will be able to use cosets to characterize when $L(a)$ is empty. Here, we use the cosets K_a for $a \in N_i$, $i \in [0, k-1]$:

$$K_a = (L_a + \langle -b + M_b \mid b \in R_{i+1},\ a \overset{*}{\rightsquigarrow} b \rangle) \cap S_{i+1}.$$

We shall see that the coset K_a corresponds to those vectors in $\mathbb{N}^{N_{[i+1,k]}} + \mathbb{Z}^{T_{[i+1,k]}}$ that can be derived using a. Using the cosets K_b for $b \in N_{[0,k-1]}$, we will be able to characterize emptiness of $L(a)$ for $a \in N_{[1,k]}$. In order to do the same for $a \in N_0$, we need a final type of cosets. For each production $a \to \mathbf{u}$ with $\mathbf{u} = b_1 + \cdots + b_n + \mathbf{v}$ in our grammar with $b_1, \ldots, b_n \in N_0$ and $\mathbf{v} \in \mathbb{Z}^T$, we define

$$K_{a \to \mathbf{u}} = (M_{b_1} + \cdots + M_{b_n} + \mathbf{v}) \cap S_0.$$

The key ingredient in our proof is the following analogue of the translation of Verma, Seidl, and Schwentick [38] to Presburger arithmetic. Here, we express the set $L(a)$ of derivable vectors of a nonterminal a as a coset.

Theorem 7.4. *If $G = (N, T, P)$ is bidirected, then $M(a) + D_a = M_a$ for every $a \in R$. In particular, $L(a) + D_a = L_a$.*

Intuition for the proof of Theorem 7.4. Before we sketch the proof, let us compare it with the construction of [38]. They show that if we assign to each production p in a context-free grammar a number $\mathbf{x}(p)$ saying how often p is applied, then there is a derivation consistent with $\mathbf{x}$ if and only if:

(i) each nonterminal is produced as many times as it is consumed (except for the start-symbol, which is consumed once more) and
(ii) the participating nonterminals (i.e. for which $\mathbf{x}(p) > 0$) must be *connected*: The latter means, for each nonterminal, it must be possible to reach its consumed nonterminal by way of productions that occur in $\mathbf{x}$.

Essentially, we will argue that for bidirected grammars, one can drop (ii). This is because, independently of a particular derivation, each set N_i decomposes into connected components with respect to $\overset{*}{\rightsquigarrow}$. Then, instead of stipulating connectedness of the set of used nonterminals, we only need all used nonterminals to be in the same $\overset{*}{\rightsquigarrow}$-component. To construct a derivation inside one $\overset{*}{\rightsquigarrow}$ component, we need to show that there exist "connecting derivations" between each pair in $\overset{*}{\rightsquigarrow}$. This is made precise in the notion of Kirchhoff graphs, which we define next.

Kirchhoff graphs. Let $a \in R_i$. A *Kirchhoff graph for a* is a directed graph whose set of vertices is $\{b \in R_i \mid a \overset{*}{\rightsquigarrow} b\}$, such that there exists an edge (b, c) for each $b, c \in R_i$ with $a \overset{*}{\rightsquigarrow} b$ and $a \overset{*}{\rightsquigarrow} c$, and where an edge (b, c) is weighted by an element $\mathbf{g}_{b,c} \in \mathbb{N}^{N_{[i+1,k]}} + \mathbb{Z}^{T_{[i+1,k]}}$ such that the following holds:

(i) $b \overset{*}{\Rightarrow}_i c + \mathbf{g}_{b,c}$ for every b, c,
(ii) $\mathbf{g}_{b,b} = \mathbf{0}$ for every b, and
(iii) for any vertices b, c, d, we have $\mathbf{g}_{b,c} + \mathbf{g}_{c,d} \approx_a \mathbf{g}_{b,d}$.

The term stems from the fact that these graphs satisfy (up to $\approx_a$) a condition like Kirchhoff's law on voltage drops: The weight sum of every cycle is zero. In our case, adding up the $\mathbf{g}_{b,c}$ along a cycle will yield zero up to $\approx_a$. This implies the following property:

Lemma 7.5. *If $e_1, \ldots, e_\ell, f_1, \ldots, f_\ell \in R_i$ satisfy $a \overset{*}{\rightsquigarrow} e_j$ and $a \overset{*}{\rightsquigarrow} f_j$ for $j \in \{1, \ldots, \ell\}$ and π is a permutation of $\{1, \ldots, \ell\}$, then in a Kirchhoff graph for a, we have*

$$\mathbf{g}_{e_1,f_1} + \cdots + \mathbf{g}_{e_\ell,f_\ell} \approx_a \mathbf{g}_{e_1,f_{\pi(1)}} + \cdots + \mathbf{g}_{e_\ell,f_{\pi(\ell)}}.$$

Proof. Let us first observe that for any $e \in \{e_1, \ldots, e_\ell\}$ and $f \in \{f_1, \ldots, f_\ell\}$, the relation

$$\begin{aligned}\mathbf{g}_{e,f} + \mathbf{g}_{e',f'} &\approx_a (\mathbf{g}_{e,e'} + \mathbf{g}_{e',f}) + (\mathbf{g}_{e',e} + \mathbf{g}_{e,f'}) \\ &\approx_a (\mathbf{g}_{e,e'} + \mathbf{g}_{e',e}) + (\mathbf{g}_{e',f} + \mathbf{g}_{e,f'}) \\ &\approx_a \mathbf{g}_{e',f} + \mathbf{g}_{e,f'}\end{aligned}$$

follows from the definition of Kirchhoff graphs. This is the case of

$$\mathbf{g}_{e_1,f_1} + \cdots + \mathbf{g}_{e_\ell,f_\ell} \approx_a \mathbf{g}_{e_1,f_{\pi(1)}} + \cdots + \mathbf{g}_{e_\ell,f_{\pi(\ell)}} \tag{3}$$

where π is a transposition, i.e. a permutation that swaps two points and lets all others fixed. Since every permutation of $\{1, \ldots, \ell\}$ can be written as a composition of several transpositions, (3) follows in full generality. □

Lemma 7.6. *If G is i-bidirected, then for each $a \in R_i$, there exists a Kirchhoff graph for a.*

Proof. Write $\{b \in R_i \mid a \overset{*}{\rightsquigarrow} b\} = \{b_1, \ldots, b_n\}$. To simplify notation, we write $\mathbf{g}_{r,s}$ instead of $\mathbf{g}_{b_r,b_s}$. We have to pick $\mathbf{g}_{j,j} = \mathbf{0}$. Since G is i-bidirected, we know that $\overset{*}{\rightsquigarrow}$ is symmetric. In particular, for any $j \in [1, n-1]$, there exists a $\mathbf{g}_{j,j+1}$ such that $b_j \overset{*}{\Rightarrow}_i b_{j+1} + \mathbf{g}_{j,j+1}$. Moreover, i-bidirectedness of G guarantees that there exists a $\mathbf{g}_{j+1,j}$ with $\mathbf{g}_{j+1,j} \approx_{b_j} \mathbf{g}_{j,j+1}{}^\dagger$ such that $b_{j+1} \overset{*}{\Rightarrow}_i b_j + \mathbf{g}_{j+1,j}$. Note that since $a \overset{*}{\rightsquigarrow} b_j$ and $\overset{*}{\rightsquigarrow}$ is symmetric, we have $\Delta_{b_j} = \Delta_a$ and thus $\mathbf{g}_{j+1,j} \approx_a \mathbf{g}_{j,j+1}{}^\dagger$. Finally, for $r, s \in [1, n]$ with $r < s$, we pick $\mathbf{g}_{r,s} = \mathbf{g}_{r,r+1} + \cdots + \mathbf{g}_{s-1,s}$ and similarly $\mathbf{g}_{s,r} = \mathbf{g}_{s,s-1} + \cdots + \mathbf{g}_{r+1,r}$.

Let us now show that this is indeed a Kirchhoff graph for a. We clearly have $b_r \overset{*}{\Rightarrow}_i b_s + \mathbf{g}_{r,s}$ for any $r, s \in [1, n]$. It remains to show that

$$\mathbf{g}_{r,s} + \mathbf{g}_{s,t} \approx_a \mathbf{g}_{r,t}$$

for any $r, s, t \in [1, n]$. It suffices to do this in the case that $|s - t| = 1$, because the other cases follow by induction. Consider the case $t = s + 1$ (the case $s = t + 1$ is analogous). We have to show that

$$\mathbf{g}_{r,s} + \mathbf{g}_{s,s+1} \approx_a \mathbf{g}_{r,s+1}.$$

If $r \leq s$, then both sides are identical by definition. If $r > s$, then $\mathbf{g}_{r,s}$ is defined as $\mathbf{g}_{r,s} = \mathbf{g}_{r,s+1} + \mathbf{g}_{s+1,s}$. By the choice of $\mathbf{g}_{s+1,s}$, we know that $\mathbf{g}_{s+1,s} \approx_a \mathbf{g}_{s,s+1}{}^{\dagger}$ and thus $\mathbf{g}_{s+1,s} + \mathbf{g}_{s,s+1} \approx_a \mathbf{0}$. Hence:

$$\mathbf{g}_{r,s} + \mathbf{g}_{s,s+1} = \mathbf{g}_{r,s+1} + \mathbf{g}_{s+1,s} + \mathbf{g}_{s,s+1} \approx_a \mathbf{g}_{r,s+1}. \qquad \square$$

For Theorem 7.4, we need two additional lemmas. The first follows from property (5) and induction on i.

Lemma 7.7. *For bidirected $G = (N, T, P)$ and $a \in R_i$, we have $D_a \subseteq H_a$.*

Next observe that L_a contains vectors obtained by adding, but also *subtracting* effects of derivation steps. We now show that if our grammar is i-bidirected, then each such subtraction can be realized by a sequence of ordinary derivation steps: The lemma says that every element of H_a can be written as a positive sum of derivation effects (up to a difference in D_a).

Lemma 7.8. *If G is i-bidirected, then*

$$H_a = \{-b + \mathbf{u} \mid \exists b \in R_i \colon b \Rightarrow_i \mathbf{u} \text{ and } a \overset{*}{\rightsquigarrow} b\}^* + D_a$$

for every $a \in R_i$

Here, F^* denotes the submonoid of $\mathbb{Z}^{N \cup T}$ generated by the set F. We are now ready to prove Theorem 7.4.

Proof of Theorem 7.4. We begin with the inclusion "$\subseteq$". A simple induction on the length of a derivation shows that every element of $M(a)$ belongs to M_a. Lemma 7.7 tells us that $D_a \subseteq H_a$, and since $D_a \subseteq S_i$, this implies $M_a + D_a \subseteq M_a$, hence $M(a) + D_a \subseteq M_a$.

We now prove "$\supseteq$". An element of M_a is of the form $a + \mathbf{v}$ with $\mathbf{v} \in H_a$ and $a + \mathbf{v} \in S'_i$. We claim that then $a + \mathbf{v}$ belongs to $M(a) + D_a$. Since $\mathbf{v} \in H_a$, Lemma 7.8 tells us that

$$\mathbf{v} = \sum_{j=1}^{n} -b_j + \mathbf{u}_j + \mathbf{x}_j$$

with $b_j \in R_i$, $a \overset{*}{\rightsquigarrow} b_j$, $\mathbf{u}_j \in \mathbb{N}^{R_i}$, $\mathbf{x}_j \in \mathbb{N}^{N_{[i+1,k]}} + \mathbb{Z}^{T_{[i,k]}}$ where $b_j \Rightarrow_i \mathbf{u}_j + \mathbf{x}_j$ for $j \in [1, n]$.

Since G is i-bidirected by Lemma 7.3, Lemma 7.6 yields a Kirchhoff graph for a with weights $\mathbf{g}_{b,c}$ for any $b, c \in R_i$ with $a \overset{*}{\rightsquigarrow} b$ and $a \overset{*}{\rightsquigarrow} c$. Let us now construct a derivation in G. Without loss of generality, we may assume that $\mathbf{u}_1, \ldots, \mathbf{u}_\ell \neq \mathbf{0}$ and $\mathbf{u}_{\ell+1} = \cdots = \mathbf{u}_n = \mathbf{0}$. For each $j \in [1, \ell]$, we pick some nonterminal $c_j \in R_i$ such that $c_1 = a$ and $\mathbf{u}_{j-1}(c_j) > 0$ for $j \in [2, \ell]$. By our choice of the $\mathbf{g}$'s, we can now derive as follows. We use the derivation steps $b_j \Rightarrow_i \mathbf{u}_j + \mathbf{x}_j$. But since it is possible that $b_{j+1} \Rightarrow_i \mathbf{u}_{j+1} + \mathbf{x}_{j+1}$ cannot be applied after $b_j \Rightarrow_i \mathbf{u}_j + \mathbf{x}_j$, we use derivations $c_j \overset{*}{\Rightarrow}_i b_j + \mathbf{g}_{c_j,b_j}$ as connecting derivations. Here, we think of the $\mathbf{g}_{c_j,b_j}$ as "garbage" that we produce in order to use the connecting derivations. Afterwards, we will cancel out these garbage elements. We begin with a connecting derivation in order to apply $b_1 \Rightarrow_i \mathbf{u}_1 + \mathbf{x}_1$:

$$\begin{aligned} a = c_1 &\overset{*}{\Rightarrow}_i b_1 + \mathbf{g}_{c_1,b_1} \\ &\Rightarrow_i \mathbf{u}_1 + \mathbf{x}_1 + \mathbf{g}_{c_1,b_1} = a + (-c_1 + \mathbf{u}_1) + \mathbf{g}_{c_1,b_1}. \end{aligned}$$

Since c_2 must occur in $a + (-c_1 + \mathbf{u}_1 + \mathbf{x}_1) + \mathbf{g}_{c_1,b_1}$, we can apply $c_2 \overset{*}{\Rightarrow}_i b_2 + \mathbf{g}_{c_2,b_2}$, etc. Doing this ℓ times yields

$$a \overset{*}{\Rightarrow}_i a + \sum_{j=1}^{\ell} -c_j + \mathbf{u}_j + \mathbf{x}_j + \mathbf{g}_{c_j,b_j}.$$

Let us denote the sum on the right-hand side by $\mathbf{y}$. Since want to derive $a + \sum_{j=1}^{\ell}(-b_j + \mathbf{u}_j + \mathbf{x}_j)$ instead of $a + \mathbf{y}$, we now need to correct two aspects: (i) Our derivation subtracted $c_1, \ldots, c_\ell$ instead of $b_1, \ldots, b_\ell$, so we need to add c's and subtract b's and (ii) we need to cancel out the garbage elements $\mathbf{g}_{c_j,b_j}$. When replacing b's by c's, it could be that some c's are equal to b's, so for (i), we don't have to change those. So we pick a permutation π of $\{1, \ldots, \ell\}$ and a number $r \in [1, \ell]$ so that (a) $c_j = b_{\pi(j)}$ for $j \in [1, r]$ and (b) $\{b_{\pi(r+1)}, \ldots, b_{\pi(\ell)}\}$ and $\{c_{r+1}, \ldots, c_\ell\}$ are disjoint. Now observe that the nonterminals $\{b_{\pi(r+1)}, \ldots, b_{\pi(\ell)}\}$ are never consumed in the derivation arriving at $a + \mathbf{y}$. However, since $a + \mathbf{v} \in S_i$, we know that $b_1 + \cdots + b_\ell$ must occur in $\sum_{j=1}^{\ell} \mathbf{u}_j$. Therefore, in particular $b_{\pi(r+1)} + \cdots + b_{\pi(\ell)}$ must occur in $\mathbf{y}$. But this means we can, for each $j \in [r + 1, \ell]$, apply the derivation $b_{\pi(j)} \overset{*}{\Rightarrow}_i c_j + \mathbf{g}_{b_{\pi(j)},c_j}$ to $\mathbf{y}$ (in any order). Thus, from a, using $\overset{*}{\Rightarrow}_i$, we can derive

$$\begin{aligned} &a + \sum_{j=1}^{\ell} -c_j + \mathbf{u}_j + \mathbf{x}_j + \mathbf{g}_{c_j,b_j} + \sum_{j=r+1}^{\ell} -b_{\pi(j)} + c_j + \mathbf{g}_{b_{\pi(j)},c_j} \\ &= a + \sum_{j=1}^{\ell} -b_j + \mathbf{u}_j + \mathbf{x}_j + \left(\sum_{j=1}^{\ell} \mathbf{g}_{c_j,b_j} + \sum_{j=r+1}^{\ell} \mathbf{g}_{b_{\pi(j)},c_j} \right). \end{aligned}$$

Moreover, since $a + \mathbf{v} \in S_i$, we know that $b_{\ell+1} + \cdots + b_n$ must occur in $\sum_{j=1}^{\ell} -b_j + \mathbf{u}_j + \mathbf{x}_j$, and so we can just apply the steps $b_j \Rightarrow_i \mathbf{u}_j + \mathbf{x}_j$ for $j \in [\ell + 1, n]$ in any order to obtain:

$$a \overset{*}{\Rightarrow}_i a + \sum_{j=1}^{n} -b_j + \mathbf{u}_j + \mathbf{x}_j + \left(\sum_{j=1}^{\ell} \mathbf{g}_{c_j,b_j} + \sum_{j=r+1}^{\ell} \mathbf{g}_{b_{\pi(j)},c_j} \right).$$

Now since $a + \mathbf{v} \in S'_i$, the right-hand side contains no more level-i nonterminals. Hence, the right-hand side belongs to $M(a)$. Finally, since $\pi(j) = j$ for $j \in [1, r]$ and $\mathbf{g}_{c_j,c_j} \approx \mathbf{0}$ for $j \in [1, \ell]$, the term in parentheses belongs to D_a by Lemma 7.5. Hence, $a + \mathbf{v} \in M(a) + D_a$.

Finally, note that $L(a) + D_a = L_a$ follows from $M(a) + D_a = M_a$, because $D_a \subseteq S_i$ and thus $L(a) + D_a = (M(a) \cap S_i) + D_a = (M(a) + D_a) \cap S_i = M_a \cap S_i = L_a$. $\square$

While Theorem 7.4 describes what nonterminals $a \in R_i$ can derive, we also need an analogue $a \in N_i \setminus R_i$. It is a straightforward consequence of previous steps:

Corollary 7.9. *Suppose G is i-bidirected. (1) For $a \in N_i$ for $i \in [1, k]$, we have $L(a) \neq \emptyset$ iff there is some $a' \in N_{i-1}$ and a production $a \to a'$ such that $K_{a'} \neq \emptyset$. (2) For $a \in N_0$, we have*

$L(a) \neq \emptyset$ iff there is some production $a \to \mathbf{u}$ with $\mathbf{u} \in \mathbb{N}^{N_0} + \mathbb{Z}^T$ such that $K_{a\to\mathbf{u}} \neq \emptyset$.

Constructing coset circuits

We will express our cosets in compact representations called *coset circuits*. Let Y be a finite set. A *matrix representation* of a coset $S \subseteq \mathbb{Z}^Y$ is a matrix $\mathbf{A} \in \mathbb{Z}^{X\times Z}$ and a vector $\mathbf{b} \in \mathbb{Z}^X$ such that $Y \subseteq Z$ and $S = \{\pi_Y(\mathbf{x}) \mid \mathbf{A}\mathbf{x} = \mathbf{b}\}$, where $\pi_Y\colon \mathbb{Z}^Z \to \mathbb{Z}^Y$ is the projection onto $\mathbb{Z}^Y$. Note that for a coset given as $S = \mathbf{v} + \langle \mathbf{u}_1, \dots, \mathbf{u}_n\rangle$, we can directly compute a matrix representation. A *coset circuit over* $\mathbb{Z}^Y$ is a directed acyclic graph C, whose vertices are called *gates* and

(1) Leaves, i.e. gates g with in-degree 0, are labeled by a matrix representation of a coset $C(g)$.
(2) Gates with in-degree > 0 are labeled by $+$ or $\cap$.

In a coset circuit, each gate g evaluates to a coset $C(g)$ of $\mathbb{Z}^Y$ (or the empty set): The leaves evaluate to their labels. A gate g with incoming edges from $g_1, \dots, g_m$, $m \geq 1$, evaluates to either $C(g_1) + \cdots + C(g_m)$ or $C(g_1) \cap \cdots \cap C(g_m)$ depending on whether v's label is $+$ or $\cap$.

We will construct a coset circuit that has for each $a \in N$, a gate for H_a, L_a, M_a, K_a and also for $K_{a\to\mathbf{u}}$ for productions $a \to \mathbf{u} \in P$. By Corollary 7.9, this lets us check emptiness of $L(a)$ for each $a \in N$. For $H_a, L_a, K_{a\to\mathbf{u}}$ with $a \in N_0$, we can directly create leaves with matrix representations. For the others, the definitions do not tell us directly how to compute the cosets using sums and intersections, e.g.: H_a is defined by a (potentially infinite) generating set. Thus, we first show that H_a is generated by finitely many cosets.

Lemma 7.10. *Let $G = (N, T, P)$ be a bidirected k-grammar. For every $a \in R_i$, $i \in [1, k]$, we have*

$$H_a = \langle -b + L_c \mid b \in R_i, c \in R_{i-1}, a \overset{*}{\rightsquigarrow} b, b \to c \in P\rangle.$$

Proof. By definition, we have $H_a = \langle -b + L(c) \mid b \in R_i, c \in R_{i-1}, a \overset{*}{\rightsquigarrow} b, b \to c \in P\rangle$. Since Theorem 7.4 tells us that $L_c = L(c) + D_c$, the inclusion "$\subseteq$" is immediate. For "$\supseteq$", because of $L(c) + D_c$, we shall prove that $D_c \subseteq H_a$. (Note that this is not an immediate consequence of Lemma 7.7, because $a \in R_i$ and $c \in R_{i-1}$ are on different levels).

For $D_c \subseteq H_a$, it suffices to prove that every generator $e + e^\dagger$ with $e \in R_i$, $c \overset{*}{\rightsquigarrow} e$, of D_c belongs to H_a. Since $c \overset{*}{\rightsquigarrow} e$, e appears on a right-hand side of a production and thus $L(e) \neq \emptyset$. Hence, there is some $\mathbf{u} \in \mathbb{Z}^{N_{[i+1,k]}\cup T}$ with $e \overset{*}{\Rightarrow}_i \mathbf{u}$. This implies that $e + e^\dagger \overset{*}{\Rightarrow}_i \mathbf{u} + \mathbf{u}^\dagger$. Since $\mathbf{u} + \mathbf{u}^\dagger \in D_a \subseteq H_a$ (Lemma 7.7) and $e + e^\dagger - (\mathbf{u} + \mathbf{u}^\dagger) \in H_a$, this proves that $e + e^\dagger \in H_a$. ◻

This is a straightforward consequence of Theorem 7.4. We have now described each coset in terms of other cosets using sum, intersection, but also *generated subgroup* (such as in Lemma 7.10). In order to describe cosets only using sums and intersections, we use:

Lemma 7.11. *Let $g_1, \dots, g_n \in \mathbb{Z}^m$ and let $U, S \subseteq \mathbb{Z}^m$ be subgroups with $(g_i + U) \cap S \neq \emptyset$ for all $i \in [1, n]$. Then*

$$\langle (g_1 + U) \cap S, \dots, (g_n + U) \cap S\rangle = (\langle g_1, \dots, g_n\rangle + U) \cap S. \tag{4}$$

In other words, Lemma 7.11 says that instead of imposing the condition of belonging to S *locally* at each summand $g_i + U$, it suffices to impose it *globally* on the sum $\langle g_1, \dots, g_n\rangle + U$.

Proof of Lemma 7.11. The inclusion "$\subseteq$" is obvious because U and S are subgroups. Conversely, suppose $h = x_1 g_1 + \cdots + x_n g_n + u \in S$ for some $x_1, \dots, x_n \in \mathbb{Z}$ and $u \in U$. Since $(g_i + U) \cap S \neq \emptyset$, we can choose $u_i \in U$ for each $i \in [1, n]$ such that $g_i + u_i \in S$. We compute in the quotient A/S. Note that since $h \in S$ and $g_i + u_i \in S$, we have $[u] = -[x_1 g_1 + \cdots + x_n g_n] = [x_1 u_1 + \cdots + x_n u_n]$ and thus $u - x_1 u_1 - \cdots - x_n u_n \in S$. Therefore, we have

$$\begin{aligned} h = \underbrace{x_1(g_1 + u_1)}_{\in \langle (g_1+U)\cap S\rangle} + \cdots + \underbrace{x_n(g_n + u_n)}_{\in \langle (g_n+U)\cap S\rangle} \\ + \underbrace{u - x_1 u_1 - \cdots - x_n u_n}_{\in U\cap S}. \end{aligned} \tag{5}$$

This proves that h belongs to the left-hand side of (4), since that set is closed under adding $U \cap S$. ◻

Hence, if we have built gates for U and S and are given vectors $g_1, \dots, g_n$, then to create a gate for $\langle (g_1 + U), \dots, (g_n + U)\rangle \cap S$, we can create one for $(\langle g_1, \dots, g_n\rangle + U) \cap S$. Since we can directly compute a matrix representation for $\langle g_1, \dots, g_n\rangle$, we can introduce a leaf for this subgroup.

Let us now describe how, using Lemma 7.11, each coset can be expressed using only sums and intersection. To simplify notation, for any $a \in N_i$, we write $P_a = \{(b, c) \mid b \in N_i, c \in N_{i-1}, a \overset{*}{\rightsquigarrow} b, b \to c \in P\}$. Note that

$$\begin{aligned} H_a &= \langle -b + \mathbf{u} \mid a \overset{*}{\rightsquigarrow} b,\ b \to \mathbf{u} \in P\rangle && \text{for } a \in R_0, \\ H_a &= \langle -b + L_c \mid (b, c) \in P_a\rangle && \text{for } a \in R_{[1,k]}. \end{aligned}$$

Thus, for $a \in R_0$, we have a finite generating set for H_a given explicitly in the grammar and can thus create a leaf gate for H_a labeled by an explicit matrix representation for H_a. However, for $a \in R_{[1,k]}$, we need to eliminate the the $\langle\cdot\rangle$ operator. To this end, we write

$$\begin{aligned} H_a &= \langle -b + L_c \mid (b, c) \in P_a\rangle \\ &= \sum_{(b,c)\in P_a} \langle (-b + c + H_c) \cap S_{i-1}\rangle \\ &= \sum_{(b,c)\in P_a} (\langle -b + c\rangle + H_c) \cap S_{i-1} \end{aligned} \tag{6}$$

where in the first step, we plug in the definition of L_c and in the second step, we apply Lemma 7.11. Here, the sum on the right only uses those $(b, c) \in P_a$ for which $(\langle -b + c\rangle + H_c) \cap S_{i-1} \neq \emptyset$. Now, each $\langle -b + c\rangle$ is a group for which we can create a gate with an explicit representation.

We also need to express K_a in terms of sum and intersection. First note that for $b \in R_{i+1}$,

$$\begin{aligned} \langle -b + M_b\rangle &= \langle -b + ((b + H_b) \cap S'_{i+1})\rangle \\ &= \langle (-b + S'_{i+1}) \cap H_b\rangle \\ &= H_b \cap \langle -b + S'_{i+1}\rangle \end{aligned} \tag{7}$$

provided that $(-b + S'_{i+1}) \cap H_b \neq \emptyset$; otherwise, $\langle -b + M_b\rangle$ is the trivial group $\{\mathbf{0}\}$. In the first equality, we plug in the definition of M_b. The second is due to the definition of S'_{i+1}, and the third applies Lemma 7.11, relying on $(-b + S'_{i+1}) \cap H_b$ being non-empty.

Algorithm 1: Construction of coset circuit for a bidirected k-grammar

```
Input: Bidirected $k$-grammar $G = (N, T, P)$
Create gates for $S_i$, $S'_i$, $O_a$, and $\langle -a + b\rangle$ for each $i \in [1, k]$, $a, b \in N$
Compute $\rightsquigarrow$ on $N_0$
Create gates for $H_a$, $L_a$, and $M_a$ for each $a \in R_0$
Create gates for $K_{a\to\mathbf{u}}$ for productions $a \to \mathbf{u}$, $a \in N_0$, $\mathbf{u} \in \mathbb{N}^{N_0} + \mathbb{Z}^T$
for $i = 1, \ldots, k$ do
    Create gate for $L_c \cap O_b$ for each $c \in R_{i-1}$, $b \in N_i$
    Compute the relation $\rightsquigarrow$ on $N_i$ based on non-emptiness of $L_c \cap O_b$ with $c \in N_{i-1}$, $b \in N_i$
    Compute $\overset{*}{\rightsquigarrow}$ on $N_i$ and then $P_a$ for each $a \in N_i$
    Create gate for $H_a$, then for $L_a$, then for $M_a$ for each $a \in R_i$ according to (2), (6) and (8)
    Create gate for $K_a$ for each $a \in R_{i-1}$
```

This implies that for $a \in N_i$,

$$\begin{aligned} K_a &= (L_a + \langle -b + M_b \mid b \in R_{i+1}, a \overset{*}{\rightsquigarrow} b\rangle) \cap S_{i+1} \\ &= \left(L_a + \sum_{b \in R_{i+1}, a \overset{*}{\rightsquigarrow} b} \langle -b + M_b\rangle\right) \cap S_{i+1} \\ &= \left(L_a + \sum_{b \in R_{i+1}, a \overset{*}{\rightsquigarrow} b} H_b \cap \langle -b + S'_{i+1}\rangle\right) \cap S_{i+1}, \end{aligned} \tag{8}$$

where the last sum only uses those $H_b \cap \langle -b + S'_{i+1}\rangle$ for which $(-b + S'_{i+1}) \cap H_b \neq \emptyset$. The equality follows from the definition of K_a and (7). Note that for $-b + S'_{i+1}$, it is again easy to construct a matrix representation.

Finally, observe that all these coset definitions rely on the relation $\rightsquigarrow$. On N_0, we can compute $\rightsquigarrow$ directly. On $N_{[1,k]}$, we have to rely on cosets. To this, note that for $a, b \in N_i$, we have $a \rightsquigarrow b$ if and only if there is a $c \in R_{i-1}$ with a production $a \to c$ and some $\mathbf{u} \in L_c$ with $\mathbf{u}(b) = 1$. In other words, if and only if $L_c \cap O_b \neq \emptyset$, where O_b is the coset $\{\mathbf{u} \in \mathbb{Z}^{N \cup T} \mid \mathbf{u}(b) = 1\}$.

The exact order in which the gates are constructed is given in Algorithm 1. We observe that the constructed coset circuit has depth $\leq ck$ for some constant c: The gates for H_a, L_a, M_a only depend on gates created in the last iteration of the for-loop. Moreover, each of them only adds a constant depth to the gates produced before. The gates K_a depend on gates H_b created in the same iteration, thus also adding only constant depth. Finally, note that the entries in the matrices in the labels of the leaves require at most polynomially many bits: The gates for S_i, S'_i, O_a, $\langle -a + b\rangle$ and for H_a, L_a, and M_a for $a \in R_0$ are obtained directly from the productions of G. The numbers in those, in turn, have at most polynomially many bits as they are computed using Theorem 6.5.

From coset circuits to equations

In each of our three algorithms for BIREACH, we check emptiness of coset circuits by translating them into integer linear equation systems. Let $n = |Y|$. We compute, for each gate g, a matrix $\mathbf{A}_g \in \mathbb{Z}^{s_g \times t_g}$ and a vector $\mathbf{b}_g \in \mathbb{Z}^{s_g}$ such that $C(g) = \{\pi_n(\mathbf{x}) \mid \mathbf{x} \in \mathbb{Z}^{t_g},\ \mathbf{A}_g\mathbf{x} = \mathbf{b}_g\}$, where for any $r \in \mathbb{N}$, by π_n, we denote the projection $\pi_n \colon \mathbb{Z}^r \to \mathbb{Z}^n$ onto the last n coordinates for any $r \geq n$. If g has in-degree 0, then g is already labeled with such a matrix $\mathbf{A}$ and vector $\mathbf{b}$. Now suppose g has incoming gates $g_1, \ldots, g_r$. Let $\mathbf{A}_1, \ldots, \mathbf{A}_r$ and $\mathbf{b}_1, \ldots, \mathbf{b}_r$ with $\mathbf{A}_i \in \mathbb{Z}^{s_i \times t_i}$, $\mathbf{b} \in \mathbb{Z}^{s_i}$, denote the matrices and vectors constructed for the gates $g_1, \ldots, g_r$. Then the matrix $\mathbf{A} \in \mathbb{Z}^{s \times t}$ and $\mathbf{b} \in \mathbb{Z}^s$ for g have the shape

$$\mathbf{A} = \left(\begin{array}{ccc|c} \mathbf{A}_1 & & & \\ & \ddots & & \mathbf{0} \\ & & \mathbf{A}_r & \\ \hline & \mathbf{B} & & \mathbf{C} \end{array}\right), \qquad \mathbf{b} = \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_r \\ \mathbf{0} \end{pmatrix}$$

where $\mathbf{B}$ and $\mathbf{C}$ are chosen depending on whether g is labeled with $+$ or $\cap$. If the label is $+$, then $\mathbf{B} \in \mathbb{Z}^{n \times (t_1 + \cdots + t_r)}, \mathbf{C} \in \mathbb{Z}^{n \times n}$, are chosen so that $\mathbf{Bx} = \mathbf{c}$ expresses that in $\mathbf{x}$, the last n coordinates are the sum of $\mathbf{y}_1 + \cdots + \mathbf{y}_r$, where for each $i \in [1, r]$, the vector $\mathbf{y}_i \in \mathbb{Z}^n$ contains the coordinates $t_1 + \cdots + t_i - n, \ldots, t_1 + \cdots + t_i$, i.e. the last n coordinates corresponding to $\mathbf{A}_i$. Hence, we have $s = s_1 + \cdots + s_r + n$ and $t = t_1 + \cdots + t_r + n$. If the label is $\cap$, then $\mathbf{B} \in \mathbb{Z}^{rn \times (t_1 + \cdots + t_r)}, \mathbf{C} \in \mathbb{Z}^{rn \times n}$ are chosen so that $\mathbf{Bx} = \mathbf{c}$ expresses that for each $i \in [1, r]$, the last n coordinates of $\mathbf{x}$ agree with coordinates $t_1 + \cdots + t_i - n, \ldots, t_1 + \cdots + t_i$ of $\mathbf{x}$, i.e. the last n coordinates corresponding to $\mathbf{A}_i$. Thus, we obtain $s = s_1 + \cdots + s_r + rn$ and $t = t_1 + \cdots + t_r + n$.

Thus, in any case, we have $\max\{s, t\} \leq (r + 1) \cdot \max_i \max\{s_i, t_i\}$. If $\mathbf{A}$ is a matrix, then we define its *norm*, denoted $\|\mathbf{A}\|$, as the maximal absolute value of any entry. Observe that the norm in the above constructions does not grow at all: We have $\|\mathbf{A}\| \leq \max\{\|\mathbf{A}_i\| \mid i \in [1, r]\}$ and $\|\mathbf{b}\| \leq \max\{\|\mathbf{b}_i\| \mid i \in [1, r]\}$. Suppose m is an upper bound on the number of rows and columns of the matrices in the leafs of C, and M is an upper bound on their norm. Then for any gate g of depth i, the resulting matrix $\mathbf{A}_g \in \mathbb{Z}^{s_g \times t_g}$ and vector $\mathbf{b}_g \in \mathbb{Z}^{s_g}$ satisfy $\max\{s_g, t_g\} \leq (r + 1)^i \cdot m$ and $\|\mathbf{A}_g\|, \|\mathbf{b}_g\| \leq M$. Hence, the dimensions of the matrices grow exponentially in the depth of the circuit, but polynomially for fixed depth. Moreover, their entries require no more bits than the matrices in the leaves.

Proposition 7.12. *(1) For every fixed number $d \in \mathbb{N}$, the problem* BIREACH($\mathsf{SC}^{\pm}_d$) *is in* EXP. *(2) If $\ell \in \mathbb{N}$ is fixed as well, then* BIREACH($\mathsf{SC}^{\pm}_{d,\ell}$) *is in* P. *(3)* BIREACH($\mathsf{SC}^{\pm}$) *is in* EXPSPACE.

Proof sketch. We begin with (1) and (2). According to Proposition 7.1, we need to show that emptiness for bidirected k-grammars is in EXP, and in P for fixed k.

Given a bidirected k-grammar, we construct a coset circuit C as described in Algorithm 1 by alternating emptiness checks for gates and building new gates. To check gates for emptiness, recall that the circuit has depth $\leq ck$ for some constant $c \in \mathbb{N}$. Thus, it remains to decide emptiness of a gate g in exponential time, resp. in polynomial time for bounded depth circuits. For each gate g in this coset circuit, we compute a matrix representation for the coset $C(g)$. As argued above, the resulting matrix $\mathbf{A} \in \mathbb{Z}^{s \times t}$ and vector $\mathbf{b} \in \mathbb{Z}^s$ will satisfy $s, t \leq (r + 1)^{ck}$, where r is the largest in-degree of a gate in C. The norm of $\mathbf{A}$ and $\mathbf{b}$ is at most the norm of the matrices in the leaves of C, which means the entries of $\mathbf{A}$ and $\mathbf{b}$ only require polynomially many bits. Therefore, the size of

A and **b** is polynomial in $(r+1)^{ck}$. Hence, by Theorem 3.1, we can decide emptiness of $C(g)$ in time polynomial in $(r+1)^{ck}$. Hence, Theorem 3.1 yields the desired algorithms.

For (3), we proceed slightly differently. Instead of Proposition 7.1, we use a variant with an exponential space reduction. The resulting grammars still use polynomially many productions, but the occurring numbers can be doubly exponential. Our matrix encoding thus yields matrices with exponential dimension and their entries have exponentially many bits. This still allows us to check for emptiness of coset circuits in exponential time, resulting in an EXPSPACE algorithm overall. □

REFERENCES

[1] Eric Allender, Robert Beals, and Mitsunori Ogihara. 1999. The Complexity of Matrix Rank and Feasible Systems of Linear Equations. *Comput. Complex.* 8, 2 (1999), 99–126. https://doi.org/10.1007/s000370050023

[2] Carme Àlvarez and Raymond Greenlaw. 2000. A compendium of problems complete for symmetric logarithmic space. *Comput. Complex.* 9, 2 (2000), 123–145. https://doi.org/10.1007/PL00001603

[3] Jürgen Avenhaus and Klaus Madlener. 1984. The Nielsen reduction and P-complete problems in free groups. *Theoretical Computer Science* 32, 1-2 (1984), 61–76.

[4] Norman Biggs. 1997. Algebraic potential theory on graphs. *Bulletin of the London Mathematical Society* 29, 6 (1997), 641–682.

[5] Rémi Bonnet. 2011. The Reachability Problem for Vector Addition System with One Zero-Test. In *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6907)*, Filip Murlak and Piotr Sankowski (Eds.). Springer, 145–157. https://doi.org/10.1007/978-3-642-22993-0_16

[6] P. Buckheister and Georg Zetzsche. 2013. Semilinearity and Context-Freeness of Languages Accepted by Valence Automata. In *Proc. of the 38th International Symposium on Mathematical Foundations of Computer Science (MFCS 2013) (LNCS, Vol. 8087)*, Krishnendu Chatterjee and Jirí Sgall (Eds.). Springer, Berlin Heidelberg, 231–242. https://doi.org/10.1007/978-3-642-40313-2_22

[7] Krishnendu Chatterjee, Bhavya Choudhary, and Andreas Pavlogiannis. 2018. Optimal Dyck Reachability for Data-Dependence and Alias Analysis. *Proc. ACM Program. Lang.* 2, POPL, Article 30 (Dec. 2018), 30 pages. https://doi.org/10.1145/3158118

[8] Swarat Chaudhuri. 2008. Subcubic algorithms for recursive state machines. In *POPL '08*. ACM, 159–169.

[9] Tsu-Wu J. Chou and George E. Collins. 1982. Algorithms for the Solution of Systems of Linear Diophantine Equations. *SIAM J. Comput.* 11, 4 (1982), 687–708. https://doi.org/10.1137/0211057

[10] Wojciech Czerwiński and Łukasz Orlikowski. 2021. Reachability in Vector Addition Systems is Ackermann-complete. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022.* IEEE, 1229–1240. https://doi.org/10.1109/FOCS52979.2021.00120

[11] Volker Diekert and Anca Muscholl. 2006. Solvability of Equations in Graph Groups Is Decidable. *Int. J. Algebra Comput.* 16, 6 (2006), 1047–1070. https://doi.org/10.1142/S0218196706003372

[12] Emanuele D'Osualdo, Roland Meyer, and Georg Zetzsche. 2016. First-order logic with reachability for infinite-state systems. In *Proc. of the Thirty-First Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2016)*. ACM, New York, NY, USA, 457–466.

[13] Michael Elberfeld, Andreas Jakoby, and Till Tantau. 2012. Algorithmic Meta Theorems for Circuit Classes of Constant and Logarithmic Depth. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France (LIPIcs, Vol. 14)*, Christoph Dürr and Thomas Wilke (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 66–77. https://doi.org/10.4230/LIPIcs.STACS.2012.66

[14] Moses Ganardi, Rupak Majumdar, and Georg Zetzsche. 2021. The complexity of bidirected reachability in valence systems. (2021). https://doi.org/10.48550/arXiv.2110.03654 arXiv:2110.03654

[15] Christoph Haase and Simon Halfon. 2014. Integer Vector Addition Systems with States. In *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8762)*, Joël Ouaknine, Igor Potapov, and James Worrell (Eds.). Springer, 112–124. https://doi.org/10.1007/978-3-319-11439-2_9

[16] Christoph Haase and Georg Zetzsche. 2019. Presburger arithmetic with stars, rational subsets of graph groups, and nested zero tests. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019.* IEEE, 1–14. https://doi.org/10.1109/LICS.2019.8785850

[17] Matthew Hague and Anthony Widjaja Lin. 2011. Model Checking Recursive Programs with Numeric Data Types. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6806)*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.). Springer, 743–759. https://doi.org/10.1007/978-3-642-22110-1_60

[18] Ilya Kapovich, Richard Weidmann, and Alexei Myasnikov. 2005. Foldings, graphs of groups and the membership problem. *International Journal of Algebra and Computation* 15, 01 (2005), 95–128.

[19] Adam Husted Kjelstrøm and Andreas Pavlogiannis. 2022. The decidability and complexity of interleaved bidirected Dyck reachability. *Proc. ACM Program. Lang.* 6, POPL (2022), 1–26. https://doi.org/10.1145/3498673

[20] Ulla Koppenhagen and Ernst W. Mayr. 1997. The Complexity of the Coverability, the Containment, and the Equivalence Problems for Commutative Semigroups. In *Fundamentals of Computation Theory, 11th International Symposium, FCT '97, Kraków, Poland, September 1-3, 1997, Proceedings (Lecture Notes in Computer Science, Vol. 1279)*, Bogdan S. Chlebus and Ludwik Czaja (Eds.). Springer, 257–268. https://doi.org/10.1007/BFb0036189

[21] Serge Lang. 2005. *Algebra* (revised third edition ed.). Springer-Verlag.

[22] Jérôme Leroux. 2021. The Reachability Problem for Petri Nets is Not Primitive Recursive. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022.* IEEE, 1241–1252. https://doi.org/10.1109/FOCS52979.2021.00121

[23] Jérôme Leroux, Grégoire Sutre, and Patrick Totzke. 2015. On the Coverability Problem for Pushdown Vector Addition Systems in One Dimension. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 9135)*, Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann (Eds.). Springer, 324–336. https://doi.org/10.1007/978-3-662-47666-6_26

[24] Yuanbo Li, Qirun Zhang, and Thomas W. Reps. 2020. Fast graph simplification for interleaved Dyck-reachability. In *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*, Alastair F. Donaldson and Emina Torlak (Eds.). ACM, 780–793. https://doi.org/10.1145/3385412.3386021

[25] Yuanbo Li, Qirun Zhang, and Thomas W. Reps. 2021. On the complexity of bidirected interleaved Dyck-reachability. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–28. https://doi.org/10.1145/3434340

[26] Markus Lohrey. 2013. The rational subset membership problem for groups: a survey. In *Groups St Andrews*, Vol. 422. 368–389.

[27] Markus Lohrey and Géraud Sénizergues. 2007. When is a graph product of groups virtually-free? *Communications in Algebra®* 35, 2 (2007), 617–621.

[28] Markus Lohrey and Benjamin Steinberg. 2008. The submonoid and rational subset membership problems for graph groups. *Journal of Algebra* 320, 2 (2008), 728–755.

[29] Markus Lohrey and Benjamin Steinberg. 2010. An automata theoretic approach to the generalized word problem in graphs of groups. *Proc. Amer. Math. Soc.* 138, 2 (2010), 445–453.

[30] Markus Lohrey and Georg Zetzsche. 2018. Knapsack in Graph Groups. *Theory Comput. Syst.* 62, 1 (2018), 192–246. https://doi.org/10.1007/s00224-017-9808-3

[31] Roger C Lyndon and Paul E Schupp. 1977. *Combinatorial Group Theory*. Springer-Verlag.

[32] Ernst W Mayr and Albert R Meyer. 1982. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics* 46, 3 (1982), 305–329. https://doi.org/10.1016/0001-8708(82)90048-2

[33] Roland Meyer, Sebastian Muskalla, and Georg Zetzsche. 2018. Bounded Context Switching for Valence Systems. In *Proc. of the 29th International Conference on Concurrency Theory (CONCUR 2018) (LIPIcs, Vol. 118)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 12:1–12:18. https://doi.org/10.4230/LIPIcs.CONCUR.2018.12

[34] K. A. Mikhailova. 1966. The occurrence problem for direct products of groups. *Matematicheskii Sbornik (Novaya Seriya)* 70(112) (1966), 241–251. Issue 2.

[35] Omer Reingold. 2008. Undirected connectivity in log-space. *J. ACM* 55, 4 (2008), 17:1–17:24. https://doi.org/10.1145/1391289.1391291

[36] Klaus Reinhardt. 2008. Reachability in Petri nets with inhibitor arcs. *Electronic Notes in Theoretical Computer Science* 223 (2008), 239–264.

[37] Aneesh Shetty, Krishna S., and Georg Zetzsche. 2021. Scope-Bounded Reachability in Valence Systems. In *Proc. of the 32nd International Conference on Concurrency Theory (CONCUR 2021) (LIPIcs, Vol. 118)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. https://doi.org/10.4230/LIPIcs.CONCUR.2021.29

[38] Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. 2005. On the Complexity of Equational Horn Clauses. In *Automated Deduction - CADE-20, 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3632)*, Robert Nieuwenhuis (Ed.). Springer, 337–352. https://doi.org/10.1007/11532231_25

[39] D. Wise. 2012. *From Riches to RAAGs: 3-manifold, right-angled Artin groups, and cubical geometry.* American Mathematical Society.

[40] Guoqing Xu, Atanas Rountev, and Manu Sridharan. 2009. Scaling CFL-Reachability-Based Points-To Analysis Using Context-Sensitive Must-Not-Alias

Analysis. In *ECOOP 2009 - Object-Oriented Programming, 23rd European Conference, Genoa, Italy, July 6-10, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5653)*, Sophia Drossopoulou (Ed.). Springer, 98–122. https://doi.org/10.1007/978-3-642-03013-0_6

[41] Dacong Yan, Guoqing Xu, and Atanas Rountev. 2011. Demand-driven context-sensitive alias analysis for Java. In *Proceedings of the 20th International Symposium on Software Testing and Analysis, ISSTA 2011, Toronto, ON, Canada, July 17-21, 2011*, Matthew B. Dwyer and Frank Tip (Eds.). ACM, 155–165. https://doi.org/10.1145/2001420.2001440

[42] Georg Zetzsche. 2013. Silent Transitions in Automata with Storage. In *Proc. of the 40th International Colloquium on Automata, Languages and Programming (ICALP 2013) (LNCS, Vol. 7966)*, Fedor V. Fomin, Rūsiņš Freivalds, Marta Kwiatkowska, and David Peleg (Eds.). Springer, Berlin Heidelberg, 434–445. https://doi.org/10.1007/978-3-642-39212-2_39

[43] Georg Zetzsche. 2015. Computing downward closures for stacked counter automata. In *Proc. of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 30)*, Ernst W. Mayr and Nicolas Ollinger (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 743–756. https://doi.org/10.4230/LIPIcs.STACS.2015.743

[44] Georg Zetzsche. 2016. Monoids as Storage Mechanisms. *Bull. EATCS* 120 (2016). http://eatcs.org/beatcs/index.php/beatcs/article/view/459

[45] Georg Zetzsche. 2016. *Monoids as Storage Mechanisms*. PhD thesis.

[46] Georg Zetzsche. 2021. The emptiness problem for valence automata over graph monoids. *Inf. Comput.* 277 (2021), 104583. https://doi.org/10.1016/j.ic.2020.104583

[47] Georg Zetzsche. 2021. Recent Advances on Reachability Problems for Valence Systems (Invited Talk). In *Reachability Problems - 15th International Conference, RP 2021, Liverpool, UK, October 25-27, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 13035)*, Paul C. Bell, Patrick Totzke, and Igor Potapov (Eds.). Springer, 52–65. https://doi.org/10.1007/978-3-030-89716-1_4

[48] Qirun Zhang, Michael R. Lyu, Hao Yuan, and Zhendong Su. 2013. Fast algorithms for Dyck-CFL-reachability with applications to alias analysis. In *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, Seattle, WA, USA, June 16-19, 2013*, Hans-Juergen Boehm and Cormac Flanagan (Eds.). ACM, 435–446. https://doi.org/10.1145/2491956.2462159

[49] Qirun Zhang and Zhendong Su. 2017. Context-sensitive data-dependence analysis via linear conjunctive language reachability. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, Giuseppe Castagna and Andrew D. Gordon (Eds.). ACM, 344–358. https://doi.org/10.1145/3009837.3009848