Recent Advances on Reachability Problems for Valence Systems (Invited Talk)

Georg Zetzsche(✉)

Max Planck Institute for Software Systems (MPI-SWS), Saarbrücken, Germany
georg@mpi-sws.org

Abstract. Valence systems are an abstract model of computation that consists of a finite-state control and some storage mechanism. In contrast to traditional models, the storage mechanism is not fixed, but given as a parameter. This allows us to precisely state questions like: For which storage mechanisms is the reachability problem decidable?

This survey reports on recent results that aim to understand the impact of the storage mechanism on decidability and complexity of several variants of the reachability problem. The considered problems are configuration reachability, model-checking first-order logic with reachability, and reachability under bounded context switching and scope-boundedness.

1 Introduction

Reachability problems play a central role in automata theory, particularly in applications to verification. The most prominent example is safety verification, where we have some system model and we want to establish algorithmically that in this model it is not possible to reach certain undesirable configurations.

Therefore, during the last few decades, an extensive research effort has aimed to understand, for various kinds of abstract system models, whether reachability is decidable and with which computational complexity. Many of the results in this space consider abstract system models that consist of some *finite-state control* and some *storage mechanism*. This is because when we want to verify a particular program, the finite-state control allows us to describe the control flow of the program, whereas the storage mechanism can be used to store memory contents.

Well-known examples of such models are *vector addition systems with states (VASS)* and *pushdown systems*. In a VASS, the storage mechanism consists of several $\mathbb{N}$-counters: These assume values in the natural numbers that can be *incremented* and *decremented* (but not tested for zero). In a pushdown system, the storage consists of a stack that can be manipulated with *push* and *pop* instructions. In addition to these basic types of storage mechanisms, there exists a rich variety of extensions both of $\mathbb{N}$-counters (resets [7,11], transfers [11], lossiness [24], just to name a few) and of pushdowns (e.g. higher-order stacks [28], additional counters [15,21], etc.).

P. C. Bell et al. (Eds.): RP 2021, LNCS 13035, pp. 52–65, 2021.
https://doi.org/10.1007/978-3-030-89716-1_4

A large part of the questions studied in this space are of the following form: For a concrete storage mechanism, is a certain reachability problem decidable, and if so, what is the complexity?

This survey presents a line of work that takes a slightly different perspective. Instead of studying concrete storage mechanisms, can we obtain general insights into how the structure of the storage mechanism impacts decidability and complexity of reachability problems? To this end, one considers an abstract model in which the storage mechanism appears as a parameter. Then, one can ask: For which storage mechanism is a particular problem decidable, tractable, etc.?

A Too General Question. To set the stage, we start with a question that is likely too general to answer. Suppose we have a storage mechanism whose (finite) set of instructions is an alphabet X. The set of all sequences of instructions that bring the storage into an accepting configurations is a language $L \subseteq X^*$. We consider the following decision problem $\mathsf{REACH}(L)$:

Given A regular language $R \subseteq X^*$.
Question Is $R \cap L$ non-empty?

For example, suppose $X_d = \{a_1, \ldots, a_d, \bar{a}_1, \ldots, \bar{a}_d\}$ and let $V_d \subseteq X_d^*$ be the set of all words where for every i: (1) there are just as many a_i as $\bar{a}_i$ and (2) in every prefix, there are at least as many a_i as there are $\bar{a}_i$. Then, $\mathsf{REACH}(V_d)$ is just the reachability problem in d-dimensional VASS.

Another example is the language $d_2 \subseteq X_2$, where $w \in P_d$ if and only if w can be obtained from the empty word by repeatedly inserting the words $a_i\bar{a}_i$ with $i \in \{1, \ldots, d\}$. Then, $\mathsf{REACH}(P_d)$ is just the reachability problem (of the empty-stack configuration) for pushdown systems with d stack symbols.

Furthermore, suppose $Z_d \subseteq X_d^*$ is the set of words that contain, for each i, the same number of a_i as $\bar{a}_i$. Then $\mathsf{REACH}(Z_d)$ is the reachability problem for automata with d separate $\mathbb{Z}$-counters, which can assume values in $\mathbb{Z}$ and have to be zero in the end. This corresponds to the model of $\mathbb{Z}$-VASS [13] (which are also known as blind counter automata [12] and are in most contexts equivalent to reversal bounded counter machines [16]).

This raises the following question:

Question 1.1. *For which languages L is* $\mathsf{REACH}(L)$ *decidable?*

Of course, understanding this would be extremely useful for designing abstract system models for reasoning about programs. Unfortunately, Question 1.1 currently appears out of reach. In fact, it even seems unlikely that there exists an illuminating characterization.

However, there is a more restricted setting that still covers many storage mechanisms from the literature and, as it turned out during the last few years, several variants of the reachability problem admit simple characterizations in terms of decidability and complexity.

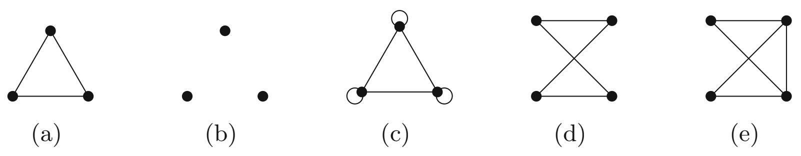


Fig. 1. Example graphs

Graphs. In this setting, the storage mechanism is defined by an undirected graph $\Gamma = (V, E)$ where self-loops are allowed. Hence, V is a finite set of *vertices* and $E \subseteq \{e \subseteq V \mid 1 \leq |e| \leq 2\}$ is its set of *edges*. A vertex v is *looped* if $\{v\} \in E$, otherwise it is *unlooped*. We say that Γ is a *clique* if there is an edge between any two distinct vertices. A graph is *looped* (*unlooped*) if all its vertices are looped (unlooped). A graph $\Gamma_0 = (V_0, E_0)$ is an *induced subgraph* of $\Gamma_1 = (V_1, E_1)$ if Γ_0 is isomorphic to a restriction of Γ_1 onto some vertex set. Formally, this means there there is an injective map $f\colon V_0 \to V_1$ such that for $u, v \in V_0$, we have $\{f(u), f(v)\} \in E_1$ if and only if $\{u, v\} \in E_0$.

To a graph Γ, we associate the alphabet $X_\Gamma = \{v, \bar{v} \mid v \in V\}$, which we think of as a set of instructions as above. Intuitively, $\bar{v}$ will be the inverse instruction of v, as in the examples above. Moreover, the edges of Γ tell us whether the respective instructions should commute. Thus, we obtain a rewriting relation $\twoheadrightarrow$ on the set of words X_Γ^*:

$$r v \bar{v} s \twoheadrightarrow rs \qquad \text{for } r, s \in X_\Gamma^* \text{ and } v \in V\text{, and} \tag{R1}$$

$$rxys \twoheadrightarrow ryxs \qquad \text{for } r, s \in X_\Gamma^* \text{ and } x \in \{u, \bar{u}\},\ y \in \{v, \bar{v}\},\ \{u, v\} \in E \tag{R2}$$

In particular, if v has a self-loop ($\{v\} \in E$), then we have $r\bar{v}vs \twoheadrightarrow rv\bar{v}s \twoheadrightarrow rs$.

This allows us to define a language that can play a similar role as the languages above:

$$L(\Gamma) = \{w \in X_\Gamma^* \mid w \overset{*}{\twoheadrightarrow} \varepsilon\},$$

where $\varepsilon \in X_\Gamma^*$ is the empty word and $\overset{*}{\twoheadrightarrow}$ is the reflexive transitive closure of $\twoheadrightarrow$.

Examples. Let us see how to realize storage mechanisms with graphs. If Γ is one of the graphs from Figs. 1a to 1c, then, up to renaming letters, $L(\Gamma)$ is V_3, P_3, or Z_3 from above.

If Γ is the graph from Fig. 1d, then $L(\Gamma)$ is the language corresponding to two pushdowns. This is because the vertices on the left together behave like one pushdown; the same is true for the two vertices on the right. Moreover, Γ has an edge from any left vertex to any right vertex. Hence, these two pushdowns can be used independently. It is well-known that such systems can simulate Turing machines, so that the reachability problem is undecidable.

Let Γ be the graph from Fig. 1e. The two vertices on the left together realize a pushdown (with two stack symbols). The two vertices on the right are adjacent and thus behave like two $\mathbb{N}$-counters. Furthermore, as in Fig. 1d, there are edges everywhere from left to right. Thus, Γ realizes a storage consisting of one pushdown and two $\mathbb{N}$-counters. Thus, we have a two-dimensional pushdown VASS [21].

Suppose Γ is the *direct product* of graphs Γ_1 and Γ_2. This means, Γ is obtained from the disjoint union of Γ_1 and Γ_2 by adding an edge from every vertex of Γ_1 to every vertex of Γ_2. Then Γ realizes a storage mechanism that consists of both the storage mechanisms of Γ_1 and Γ_2, and they can be used independently. An instance of this which will be used frequently is that of *adding an* $\mathbb{N}$-*counter* or *adding a* $\mathbb{Z}$-*counter*. This means, we take the direct product with an unlooped (resp. looped) vertex.

If Γ is obtained from Γ_0 by adding an isolated vertex v with no self-loop, then Γ behaves like a stack whose entries are configurations of the storage mechanism of Γ_0. Hence, with Γ, we have the instructions of Γ_0, which act on the top-most entry in the stack. Moreover, using v, we can start a new stack entry. With $\bar{v}$, we pop the top-most entry. The latter can only succeed if the top-most entry (which is a configuration of Γ_0) is final according to Γ_0. Thus, we think of the storage mechanism of Γ as obtained from that of Γ_0 by *building stacks*.

Valence Systems. Given a graph $\Gamma = (V, E)$, we can now define a formal machine model that uses Γ as its storage mechanism. A *valence system over* Γ is a pair $\mathcal{A} = (Q, T)$, where Q is a finite set of *states* and $T \subseteq Q \times X_\Gamma \times Q$ is its set of *transitions*. A *pre-configuration* of $\mathcal{A}$ is a pair (q, w) with $q \in Q$ and $w \in X_\Gamma^*$. Then (q, w) can *reach* (q', w') *in one step* if there is a transition (q, u, q') with $w' = wu$. In this case, we write $(q, w) \to (q, w')$.

The *reachability problem for valence systems over* Γ, short $\mathsf{REACH}(\Gamma)$ is the following:

Given A valence system $\mathcal{A} = (Q, T)$ over Γ and states $s, t \in Q$
Question Is there some $w \in X_\Gamma^*$ such that $(s, \varepsilon) \xrightarrow{*} (t, w)$ with $w \overset{*}{\twoheadrightarrow} \varepsilon$?

Now clearly, $\mathsf{REACH}(\Gamma)$ is essentially the same as $\mathsf{REACH}(L(\Gamma))$. This allows us to formulate a more manageable version of Question 1.1:

Question 1.2. *For which graphs Γ is* $\mathsf{REACH}(\Gamma)$ *decidable?*

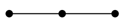
This question is clearly much more restricted in scope than Question 1.1. Therefore, there is hope that we can understand this class of graphs. And in fact, while Question 1.2 is still not settled, at least a partial answer is available (see Theorem 2.1).

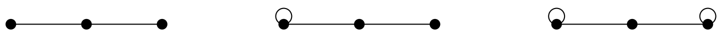
Outline. This survey reports on results about variations of Question 1.2. In other words, we focus on decidability and complexity results concerning reachability-type problems for valence systems. In Sect. 2, we consider Question 1.2 itself. In Sect. 3, we turn to the complexity of $\mathsf{REACH}(\Gamma)$ depending on Γ. In Sect. 4, we look at a harder problem, namely model-checking first-order logic with a reachability predicate for the configuration space of a valence system. Finally, in Sects. 5 and 6, we consider underapproximations of the set of all runs in which reachability is decidable for every Γ.

Historical Notes. Some remarks on the history of the notion of valence systems are in order. The origin of this concept is the idea to study finite automata in which each edge is labeled (in addition to some input) by an element of a (typically infinite) monoid. This type of model has been studied under various names by several authors, either defining acceptance by producing the identity element of the monoid [10,17–19,26] or a prescribed target set [33]. The earliest (although implicit) use of this is probably the Chomsky-Schützenberger theorem, stating that every context-free language can be obtained using a rational transduction from the word problem of the free group [2]. The term *valence automaton* (and thus valence system) originates from the theory of regulated rewriting (see [5] for a general overview), where it was first used for *valence grammars* [29], in which each grammar rule has an associated monoid element. Afterwards, the term was also applied to automata (e.g. in [10]).

The graphs Γ above, together with their interpretation as storage mechanisms, were introduced in [36] and used there to define monoids, which were then used in valence automata. For this survey, it turned out that avoiding the terminology of monoids simplified the exposition.

2 Reachability

We begin with the partial answer that exists for Question 1.2. As we have seen in the examples above (Fig. 1e), there are graphs that realize the storage mechanism of a pushdown with additional $\mathbb{N}$-counters. Systems with such a storage are called *pushdown VASS (PVASS)* in the literature [8,21]. Whether reachability is decidable for these is a long-standing open problem in the area of infinite-state systems [8,21]. In fact, even for PVASS with a single counter (i.e. one-dimensional PVASS), the decidability status is open. Thus, determining the decidability status of REACH for the graph  would amount to a solution to this problem. Aside from this, there are two other graphs that realize a one-dimensional PVASS: We say that Γ is a *PVASS-graph* if it is isomorphic to one of the following three graphs:



The fact that the left and the middle graph represent a (one-dimensional) PVASS can be seen as for Fig. 1e. For the graph on the right, this follows from the classical Chomsky-Schützenberger theorem [2,18].

We say that the graph Γ is *PVASS-free* if it has no PVASS-graph as an induced subgraph. Observe that a graph Γ is PVASS-free if and only if in the neighborhood of each unlooped vertex, any two vertices are adjacent.

To state the result, we need a further notion. We define the class of *transitive forests* inductively. First, every isolated vertex 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 vertex v to Γ_1 so that v is adjacent to every vertex in Γ_1, then Γ is also a transitive forest.

We are now ready to state the partial answer to Question 1.2.

Theorem 2.1 **([38]).** *Let Γ be PVASS-free. Then* $\mathsf{REACH}(\Gamma)$ *is decidable if and only if Γ is a transitive forest.*

By $\mathsf{SC}^{\pm}$, we denote the class of PVASS-free transitive forests. Intuitively, the storage mechanisms in $\mathsf{SC}^{\pm}$ are obtained as follows. In the simplest case, they are unlooped cliques (hence a set of $\mathbb{N}$-counters). In addition to this, we can *build stacks* and *add $\mathbb{Z}$-counters.* In this notation, "$\pm$" stands for the two types of counters: We start with $\mathbb{N}$-counters ($+$), but after building stacks once, we can then only add $\mathbb{Z}$-counters ($-$).

In contrast, let SC^{+} consist of all graphs that are transitive forests and contain a PVASS-graph. One can show that if Γ is not a transitive forest, then $\mathsf{REACH}(\Gamma)$ is undecidable [38]. Thus, SC^{+} is the class of graphs for which decidability remains open. Intuitively, the corresponding storage mechanisms are obtained by starting with a set of $\mathbb{N}$-counters and then alternating (1) building stacks and (2) adding $\mathbb{N}$-counters.

Open Problem 2.2. *Is* $\mathsf{REACH}(\Gamma)$ *decidable for every Γ in* SC^{+}*?*

Of course, the simplest case of Open Problem 2.2 is the reachability problem in one-dimensional PVASS.

3 Complexity

Let us now turn to the complexity of $\mathsf{REACH}(\Gamma)$. What we know so far is confined to the class SC^{-}, which consists of all graphs in $\mathsf{SC}^{\pm}$ that do not have •——• as an induced subgraph. Hence, intuitively, the corresponding storage mechanisms are obtained by starting from a pushdown and then alternating (1) adding $\mathbb{Z}$-counters and (2) building stacks. (Thus, they are the same as $\mathsf{SC}^{\pm}$, but all the counters are $\mathbb{Z}$-counters, which explains the "$-$"). This is an important subclass, because according to a characterization in [1], these are exactly those graphs for which valence automata (i.e. valence systems that can read input and accept languages) have semilinear Parikh images.

Among the graphs in SC^{-}, the complexity landscape is understood.

Theorem 3.1 **([14]).** *Let Γ be a graph in* SC^{-}*. Then* $\mathsf{REACH}(\Gamma)$ *is*

1. NL*-complete if Γ is a looped clique.*
2. P*-complete if Γ is a disjoint union of at least two cliques, and*
3. NP*-complete otherwise.*

Strictly speaking, the proof in [14] is only about the case where Γ has a self-loop on every vertex (in this case, $\mathsf{REACH}(\Gamma)$ is the rational subset membership problem for graph groups, see also [22]). However, the proof works essentially the same for all of SC^-. Moreover, it follows from [14] that if the graph Γ is part of the input, then reachability is NEXP-complete. In order to show these results, the paper [14] introduces an extension of existential Presburger arithmetic inspired by [30] and determines its complexity.

However, the complexity of reachability for the graphs in $\mathsf{SC}^\pm$ is far from being understood.

Open Problem 3.2. *Describe the complexity landscape of* REACH *for the graphs in* $\mathsf{SC}^\pm$*.*

Until a few years ago, Open Problem 3.2 seemed out of reach. However, given the recent stunning resolution of the complexity of reachability in VASS [3,4,20] and the fact that decidability for $\mathsf{SC}^\pm$ is shown in [38] using a reduction to reachability in VASS with nested zero tests, there is hope to obtain new insights into Open Problem 3.2.

4 First-Order Logic with Reachability

We now consider a decision problem that is computationally significantly harder than traditional reachability. Instead of asking whether a particular configuration can reach another, we want to decide a given first-order sentence that can mention configurations and express reachability (either in a single step or in a finite run).

Configuration Graphs of Valence Systems. Let us make this precise. With a valence system $\mathcal{A} = (Q, T)$ over a graph $\Gamma = (V, E)$ we define its configuration space as follows. Its universe consists of the configurations of $\mathcal{A}$, which we define next. It is not appropriate to define pre-configurations (recall that those are pairs $(q, w) \in Q \times X_\Gamma^*$) as elements of the configuration space. This is because, in all the examples mentioned above, the configurations of the realized storage mechanism correspond rather to certain equivalence classes of words in X_Γ^*. For example, if Γ is the graph in Fig. 1a, the set of configurations should be $Q \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ and if Γ is the graph in Fig. 1b with vertices a, b, c, then the configuration graph should be $Q \times \{a, b, c\}^*$. In general, the equivalence relation is given by the rewriting relation $\twoheadrightarrow$. We define the equivalence relation $\equiv$ to be the reflexive, symmetric, and transitive closure of $\twoheadrightarrow$. It is not difficult to show that then we have $w \equiv \varepsilon$ if and only if $w \overset{*}{\twoheadrightarrow} \varepsilon$ (this is because $\twoheadrightarrow$ is terminating and confluent, see [37, Equation (8.2)]). The equivalence class of $w \in X_\Gamma^*$ is denoted $[w]$. We shall define configurations of $\mathcal{A}$ as certain equivalence classes with respect to $\equiv$.

Recall that in a valence system, we consider a run arriving in (q, w) to be valid if $w \overset{*}{\twoheadrightarrow} \varepsilon$. Therefore, we define an equivalence class $[u]_\equiv$ is *admissible (for configurations)* if there is some $v \in X_\Gamma^*$ with $uv \overset{*}{\twoheadrightarrow} \varepsilon$ (equivalently, $[uv] = [\varepsilon]$). This leads to the following definition.

A *configuration* of $\mathcal{A}$ is a pair $(q, [w])$ with $q \in Q$ and $w \in X_\Gamma^*$ such that w is admissible. Observe that in all the examples given above, this definition yields a notion of configuration that fits with the realized storage mechanism.

On configurations, it is now natural to define a step relation: Whenever we have $(q, w) \to (q', w')$, then we also have $(q, [w]) \to (q', [w'])$.

The Logical Structure. With the valence system $\mathcal{A}$, we associate the following logical structure $\mathcal{S}(\mathcal{A})$. Its universe is the set of configurations of $\mathcal{A}$. Moreover, it has the following predicates:

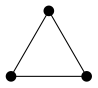
1. For each configuration, it has a constant.
2. For each $q \in Q$, there is a unary predicate $\mathsf{state}_q(\cdot)$, which is true if the finite-state component of a configuration is q.
3. A binary one-step relation $\mathsf{step}(\cdot,\cdot)$, which states that one configuration can reach the other in exactly one step.
4. A binary reachability relation $\mathsf{reach}(\cdot,\cdot)$, with expresses reachability with an arbitrary run.

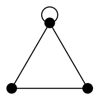
We are now interested in the model checking problem of first-order sentences over the structure $\mathcal{S}(\mathcal{A})$:

Given A first-order formula φ over the above signature.
Question Does φ hold in $\mathcal{S}(\mathcal{A})$?

Since this problem consists in deciding first-order sentences that involve reachability, we call this problem briefly $\mathsf{FO}[\mathsf{R}]$ for Γ.

In order to state the result on decidability of $\mathsf{FO}[\mathsf{R}]$, we need some terminology on graphs. We call a graph an $\mathbb{N}^2$-*triangle* if it is isomorphic to one of the following two graphs:





In other words, the graph realizes either (1) three $\mathbb{N}$-counters or (2) two $\mathbb{N}$-counters and one $\mathbb{Z}$-counter. We say that Γ is $\mathbb{N}^2$-*triangle-free* if it does not contain an $\mathbb{N}^2$-triangle as an induced subgraph. We are now ready to state the result about $\mathsf{FO}[\mathsf{R}]$.

Theorem 4.1 ([6]). *Let Γ be a graph. Then* $\mathsf{FO}[\mathsf{R}]$ *is decidable for Γ if and only if Γ is a disjoint union of $\mathbb{N}^2$-triangle-free cliques.*

5 Underapproximation I: Bounded Context Switching

A well-known example of a storage mechanism for which reachability is undecidable is a *multipushdown*: Two or more stacks that can be used independently. Here, undecidability is unfortunate, because this problem is equivalent to deciding safety properties of multithreaded recursive programs with shared memory [32]. However, it turned out that many bugs in such programs already manifest in runs where the program switches between its threads a small number of times [23,27]. Checking whether such runs exist is called *context-bounded model-checking* [31]. On the side of multipushdown systems, this corresponds to a small number of switches between its stacks. Moreover, given a small bound $k \geq 0$, it is NP-complete to decide if a multipushdown system has a run reaching a given configuration with at most k switches between its stacks [9,31].

Given the sharp drop in complexity from undecidable to NP, it would be useful to have a more abstract notion of bounded context switching that also applies to other storage mechanisms. Such a concept was developed in [25].

Contexts. To define this concept, we begin with the notion of contexts. In the case of a multipushdown system, a context is a segment of the run in which only one stack is used. Observe that a multipushdown consisting of r stacks, with s stack letters each, corresponds to a graph $\mathsf{MP}_{r,s}$, which is a direct product of r separate unlooped anticliques, each having s vertices. Here, an *anticlique* is a graph in which no two distinct vertices have an edge. Thus, if $\Gamma = \mathsf{MP}_{r,s}$, then a context corresponds to a word over X_Γ where the occurring vertices form an anti-clique. This motivates the following definition.

Let Γ be any graph and $w \in X_\Gamma^*$. Then the factorization of w into its *contexts* is obtained as follows. Take the maximal prefix of w whose set of vertices forms an anticlique. This prefix is the first (left-most) context in the factorization. Then, recursively factorize the remaining suffix of w. By $\|w\|$, we denote the number of contexts in its context factorization.

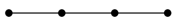
Context-Bounded Reachability. Given the notion of contexts, we can now define the problem of *context-bounded reachability for valence systems over* Γ, which we denote by $\mathsf{BCREACH}(\Gamma)$:

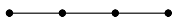
Given A valence system $\mathcal{A} = (Q, T)$, $s, t \in Q$, and $k \geq 0$ (encoded in unary)
Question Is there a $w \in X_\Gamma^*$ such that $(s, \varepsilon) \xrightarrow{*} (t, w)$ with $w \overset{*}{\twoheadrightarrow} \varepsilon$ and $\|w\| \leq k$?

It turns out that this notion of context bounding yields decidability, and even membership in NP, for every graph Γ.

Theorem 5.1 ([25]). *For every graph Γ, $\mathsf{BCREACH}(\Gamma)$ is in NP. If Γ is a transitive forest, then $\mathsf{BCREACH}(\Gamma)$ is in P.*

However, the exact set of graphs for which $\mathsf{BCREACH}$ is in P remains unclear:

Open Problem 5.2. *Describe the complexity landscape of* $\mathsf{BCREACH}$. *In particular, what is the complexity of* $\mathsf{BCREACH}$ *for the graph*  *?*

As mentioned in [25], if one could show NP-hardness for the graph  and every version of it obtained by placing self-loops, then this would yield the complete landscape: This would imply that $\mathsf{BCREACH}(\Gamma)$ is in P for transitive forests and NP-complete otherwise.

6 Underapproximation II: Scope Boundedness

Aside from bounding the number of contexts, there exist a number of other underapproximations of the set of runs of multipushdown systems that lead to decidable reachability. One such underapproximation that covers a relatively large portion of the set of all runs is obtained by *scope-bounding* [35]. The idea is, instead of bounding the number of *all* contexts, we just bound the number of contexts between each push instruction and its matching pop. A run of a multipushdown system is said to be k-scoped if every letter pushed onto some stack i will be popped within at most k visits to the same stack i. Thus a run with at most k contexts is also k-scoped. However, the price of this higher coverage is that the complexity of k-scoped reachability goes up to PSPACE [35].

As in the case of bounded context switching, this motivates the study of analogues of scope boundedness for more general storage mechanisms. In [34], such an analogue was found.

Weak Dependence Classes. In the notion of scope-boundedness, we first need an analogue of two contexts "belonging to the same stack". This is achieved by the notion of weak dependency. Observe that in the case of multipushdowns, i.e. graphs $\mathsf{MP}_{r,s}$, two instructions belong to the same pushdown if they belong to the same anticlique. However, "the same anticlique" may not be well-defined in a general graph: It is possible that for vertices u, v, w, there is an edge from u to v and from v to w, but no edge from u to w. In that case, do u and w to the same anticlique?

Instead, we generalize this in a different way. Note that two vertices in $\mathsf{MP}_{r,s}$ belong to the same stack if and only if there is a path between them in the complement graph of $\mathsf{MP}_{r,s}$. This also makes sense in the general case: We say that vertices u, v of Γ are *weakly dependent* if there is a path between u and v in the complement of Γ. Here, the complement of Γ is the graph with the same set of vertices, but the opposite set of edges. Clearly, weak dependency is an equivalence relation. Moreover, each context in a word $w \in X_\Gamma^*$ belongs to a well-defined weak dependence class.

Greedy Reductions. The next step is to find a generalization of "matching push and pop instructions". It is natural to define this based on which letters cancel in a reduction $w \overset{*}{\twoheadrightarrow} \varepsilon$. However, to avoid some corner cases in the algorithms, we define this with respect to a particular type of reduction. Instead of the rules in (R1) and (R2), consider the slightly different relation defined by

$$r v \bar{v} s \hookrightarrow rs \qquad \text{for } r, s \in X_\Gamma^* \text{ and } v \in V, \text{ and} \qquad (\text{R1}')$$

$$r \bar{v} v s \hookrightarrow rs \qquad \text{for } r, s \in X_\Gamma^* \text{ and } v \in V, \{v\} \in E, \text{ and} \qquad (\text{R2}')$$

$$r x y s \hookrightarrow r y x s \qquad \text{for } r, s \in X_\Gamma^* \text{ and } x \in \{u, \bar{u}\},\ y \in \{v, \bar{v}\},\ \{u, v\} \in E \qquad (\text{R3}')$$

Then of course, we have $w \overset{*}{\hookrightarrow} \varepsilon$ if and only if $w \overset{*}{\twoheadrightarrow} \varepsilon$. A word $w \in X_\Gamma^*$ is *irreducible* if none of the rules (R1) and (R2) (equivalently, none of the rules

(R1′) to (R3′)) are applicable to w. A *reduction* of w is a sequence of applications of (R1′) to (R3′) to obtain ε. We call this reduction *greedy* if it begins with a sequence of applications of (R1′) and (R2′) that turn each each context of w into an irreducible word. Since the relation $\twoheadrightarrow$ and thus $\hookrightarrow$ is confluent, we have $w \overset{*}{\twoheadrightarrow} \varepsilon$ if and only if w admits a greedy reduction.

Matching Relation. A reduction $\pi\colon w \overset{*}{\hookrightarrow} \varepsilon$ naturally defines a matching relation between positions of w: Two positions are related if they are cancelled using (R3′) (after possibly being transported using (R1′) and (R2′)). This binary relation on the set of positions of w, induced by π, is called the *matching relation.*

Scope Boundedness. We are now ready to formulate the notion of scope boundedness. We say that $w \in X_\Gamma^*$ is *k-scoped* if there exists a greedy reduction $\pi\colon w \overset{*}{\hookrightarrow} \varepsilon$ such that: for any two matched positions i and j, there are at most $k-1$ contexts strictly between the contexts of i and j that belong to the same weak dependence class as i and j. It is an easy exercise to observe that in the special case $\Gamma = \mathsf{MP}_{r,s}$, this notion coincides exactly with the original notion of scope boundedness.

This leads to the problem of *bounded scope reachability for valence systems over Γ*, briefly $\mathsf{BSREACH}(\Gamma)$:

Given A valence system $\mathcal{A} = (Q, T)$, $s, t \in Q$, and $k \geq 0$ (encoded in unary)
Question Is there a k-scoped $w \in X_\Gamma^*$ with $(s, \varepsilon) \xrightarrow{*} (t, w)$ and $w \overset{*}{\twoheadrightarrow} \varepsilon$?

We will also consider the problem $\mathsf{BSREACH}_k(\Gamma)$, where k is not part of the input, but fixed.

This notion of scope-boundedness does indeed yield decidable reachability for every graph. Moreover, in contrast to the case of bounded contexts, the complexity landscape is well understood.

Theorem 6.1 **([34]).** *Let Γ be a graph. Then $\mathsf{BSREACH}(\Gamma)$ is*

1. *NL-complete if Γ has at most one vertex,*
2. *P-complete if Γ is an anti-clique with ≥ 2 vertices,*
3. *PSPACE-complete otherwise.*

Note that the complexity of $\mathsf{BSREACH}$ is always PSPACE, except for those cases where scope-bounded reachability is merely classical reachability in one-counter machines (namely, the first case above) or in pushdown automata (the second case). Therefore, the paper [34] also studies the case of fixed k.

Theorem 6.2 **([34]).** *Let Γ be a graph and $k \geq 1$. Then $\mathsf{BSREACH}_k(\Gamma)$ is*

1. *NL-complete if Γ is a clique, and*
2. *P-complete otherwise.*

Since the complexity is well-understood for individual graphs, the paper [34] also studies the case where Γ is part of the input, and drawn from a class of graphs. It then gives partial results on the complexity landscape in terms of the possible graph classes. In this setting, the are many cases where the complexity is not understood yet. We refer to [34] for details.

References

1. Buckheister, P., Zetzsche, G.: Semilinearity and context-freeness of languages accepted by valence automata. In: Chatterjee, K., Sgall, J. (eds.) MFCS 2013. LNCS, vol. 8087, pp. 231–242. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40313-2_22
2. Chomsky, N., Schützenberger, M.P.: The algebraic theory of context-free languages. In: Computer Programming and Formal Systems, pp. 118–161. North-Holland, Amsterdam (1963). https://doi.org/10.1016/S0049-237X(08)72023-8
3. Czerwinski, W., Lasota, S., Lazic, R., Leroux, J., Mazowiecki, F.: The reachability problem for Petri nets is not elementary. In: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019), pp. 24–33. ACM (2019). https://doi.org/10.1145/3313276.3316369
4. Czerwinski, W., Orlikowski, L.: Reachability in vector addition systems is Ackermann-complete. CoRR abs/2104.13866 (2021). https://arxiv.org/abs/2104.13866
5. Dassow, J., Păun, G.: Regulated Rewriting in Formal Language Theory. Springer, Heidelberg (1989)
6. D'Osualdo, E., Meyer, R., Zetzsche, G.: First-order logic with reachability for infinite-state systems. In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2016), pp. 457–466. ACM (2016). https://doi.org/10.1145/2933575.2934552
7. Dufourd, C., Finkel, A., Schnoebelen, P.: Reset nets between decidability and undecidability. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 103–115. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055044
8. Englert, M., et al.: A lower bound for the coverability problem in acyclic pushdown VAS. Inf. Process. Lett. **167**, 106079 (2021). https://doi.org/10.1016/j.ipl.2020.106079
9. Esparza, J., Ganty, P., Poch, T.: Pattern-based verification for multithreaded programs. ACM ToPLaS **36**(3), 9:1–9:29 (2014)
10. Fernau, H., Stiebe, R.: Sequential grammars and automata with valences. Theoret. Comput. Sci. **276**, 377–405 (2002). https://doi.org/10.1016/S0304-3975(01)00282-1
11. Finkel, A., Sutre, G.: Decidability of reachability problems for classes of two counters automata. In: Reichel, H., Tison, S. (eds.) STACS 2000. LNCS, vol. 1770, pp. 346–357. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-46541-3_29
12. Greibach, S.A.: Remarks on blind and partially blind one-way multicounter machines. Theor. Comput. Sci. **7**, 311–324 (1978). https://doi.org/10.1016/0304-3975(78)90020-8
13. Haase, C., Halfon, S.: Integer vector addition systems with states. In: Ouaknine, J., Potapov, I., Worrell, J. (eds.) RP 2014. LNCS, vol. 8762, pp. 112–124. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11439-2_9

14. Haase, C., Zetzsche, G.: Presburger arithmetic with stars, rational subsets of graph groups, and nested zero tests. In: Proceeding of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2019), pp. 1–14. IEEE (2019). https://doi.org/10.1109/LICS.2019.8785850
15. Hague, M., Lin, A.W.: Model checking recursive programs with numeric data types. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 743–759. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_60
16. Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. J. ACM **25**(1), 116–133 (1978). https://doi.org/10.1145/322047.322058
17. Ibarra, O.H., Sahni, S.K., Kim, C.E.: Finite automata with multiplication. Theoret. Comput. Sci. **2**(3), 271–294 (1976). https://doi.org/10.1016/0304-3975(76)90081-5
18. Kambites, M.: Formal languages and groups as memory. Comm. Algebra **37**, 193–208 (2009). https://doi.org/10.1080/00927870802243580
19. Kambites, M., Silva, P.V., Steinberg, B.: On the rational subset problem for groups. J. Algebra **309**, 622–639 (2007). https://doi.org/10.1016/j.jalgebra.2006.05.020
20. Leroux, J., Schmitz, S.: Reachability in vector addition systems is primitive-recursive in fixed dimension. In: Proceeding of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2019), pp. 1–13. IEEE (2019). https://doi.org/10.1109/LICS.2019.8785796
21. Leroux, J., Sutre, G., Totzke, P.: On the coverability problem for pushdown vector addition systems in one dimension. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015. LNCS, vol. 9135, pp. 324–336. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47666-6_26
22. Lohrey, M., Steinberg, B.: The submonoid and rational subset membership problems for graph groups. J. Algebra **320**(2), 728–755 (2008)
23. Lu, S., Park, S., Seo, E., Zhou, Y.: Learning from mistakes: A comprehensive study on real world concurrency bug characteristics. In: Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2008), pp. 329–339. ACM (2008). https://doi.org/10.1145/1346281.1346323
24. Mayr, R.: Undecidable problems in unreliable computations. Theoret. Comput. Sci. **297**(1–3), 337–354 (2003). https://doi.org/10.1016/S0304-3975(02)00646-1
25. Meyer, R., Muskalla, S., Zetzsche, G.: Bounded context switching for valence systems. In: Proceeding of the 29th International Conference on Concurrency Theory (CONCUR 2018). LIPIcs, vol. 118, pp. 12:1–12:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.CONCUR.2018.12
26. Mitrana, V., Stiebe, R.: Extended finite automata over groups. Discret. Appl. Math. **108**(3), 287–300 (2001). https://doi.org/10.1016/S0166-218X(00)00200-6
27. Musuvathi, M., Qadeer, S.: Iterative context bounding for systematic testing of multithreaded programs. In: Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2007), pp. 446–455. ACM (2007). https://doi.org/10.1145/1273442.1250785
28. Ong, L.: Higher-order model checking: an overview. In: Proceeding of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2015), pp. 1–15. IEEE Computer Society (2015). https://doi.org/10.1109/LICS.2015.9
29. Păun, G.: A new generative device: Valence grammars. Rev. Roumaine Math. Pures Appl. **25**, 911–924 (1980)
30. Piskac, R., Kuncak, V.: Linear arithmetic with stars. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 268–280. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70545-1_25

31. Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 93–107. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31980-1_7
32. Ramalingam, G.: Context-sensitive synchronization-sensitive analysis is undecidable. ACM Transactions on Programming languages and Systems (TOPLAS) **22**(2), 416–430 (2000). https://doi.org/10.1145/349214.349241
33. Red'ko, V., Lisovik, L.: Regular events in semigroups. Probl. Cybern. **37**, 155–184 (1980). in Russian
34. Shetty, A.K., Krishna, S.N., Zetzsche, G.: Scope-bounded reachability in valence systems. In: Proceeding of the 32nd International Conference on Concurrency Theory (CONCUR 2021). LIPIcs, vol. 203, pp. 29:1–29:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.CONCUR.2021.29
35. Torre, S.L., Napoli, M., Parlato, G.: Reachability of scope-bounded multistack pushdown systems. Inf. Comput. **275**, 104588 (2020). https://doi.org/10.1016/j.ic.2020.104588
36. Zetzsche, G.: Silent transitions in automata with storage. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013. LNCS, vol. 7966, pp. 434–445. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39212-2_39
37. Zetzsche, G.: Monoids as Storage Mechanisms. Ph.D. Thesis, Technische Universität Kaiserslautern (2016). https://kluedo.ub.uni-kl.de/frontdoor/index/index/docId/4400
38. Zetzsche, G.: The emptiness problem for valence automata over graph monoids. Inf. Comput. **277**, 104583 (2021). https://doi.org/10.1016/j.ic.2020.104583