The Emptiness Problem for Valence Automata or: Another Decidable Extension of Petri Nets

Georg Zetzsche

Concurrency Theory Group
Fachbereich Informatik
Technische Universität Kaiserslautern
zetzsche@cs.uni-kl.de

Abstract. This work studies which storage mechanisms in automata permit decidability of the reachability problem. The question is formalized using valence automata, an abstract model that generalizes automata with storage. For each of a variety of storage mechanisms, one can choose a (typically infinite) monoid M such that valence automata over M are equivalent to (one-way) automata with this type of storage. In fact, many interesting storage mechanisms can be realized by monoids defined by finite graphs, called graph monoids. Hence, we study for which graph monoids the emptiness problem for valence automata is decidable. A particular model realized by graph monoids is that of Petri nets with a pushdown stack. For these, decidability is a long-standing open question and we do not answer it here.
However, if one excludes subgraphs corresponding to this model, a characterization can be achieved. This characterization yields a new extension of Petri nets with a decidable reachability problem. Moreover, we provide a description of those storage mechanisms for which decidability remains open. This leads to a natural model that generalizes both pushdown Petri nets and priority multicounter machines.

1 Introduction

For each storage mechanism in one-way automata, it is an important question whether the reachability problem is decidable. It therefore seems prudent to aim for general insights into which properties of storage mechanisms are responsible for decidability or undecidability.

Our approach to obtain such insights is the model of valence automata. These consist of a finite-state control and a (typically infinite) monoid that represents a storage mechanisms. The edge inscriptions consist of an input word and an element of the monoid. Then, a computation is accepting if it arrives in a final state and composing the encountered monoid elements yields the neutral element. This way, by choosing a suitable monoid, one can realize a variety of storage mechanisms as valence automata. Hence, our question becomes: *For which monoids M is the emptiness problem for valence automata over M decidable?*

We address this question for a class of monoids that was introduced in [12] and accommodates a number of storage mechanisms that have been studied in

automata theory. Examples include *pushdown stacks*, *partially blind counters* (which behave like Petri net places), and *blind counters* (which may attain negative values; these are in most situations interchangeable with reversal-bounded counters), and combinations thereof. These monoids are defined by graphs and thus called *graph monoids*[1].

A particular type of storage mechanism that can be realized by graph monoids are partially blind counters that can be used simultaneously with a pushdown stack. Automata with such a storage are equivalent to *pushdown Petri nets (PPN)*, i.e. Petri nets where the transitions can also operate on a pushdown stack. This means, a complete characterization of graph monoids with a decidable emptiness problem would entail an answer to the long-standing open question of whether reachability is decidable for this Petri net extension [8].

Contribution. While this work does not answer this open question concerning PPN, it does provide a characterization among all graph monoids that avoid this elusive storage type. More precisely, we identify a set of graphs, 'PPN-graphs', each of which corresponds precisely to PPN. Then, among all graphs Γ avoiding PPN-graphs as induced subgraphs, we characterize those for which the graph monoid $\mathbb{M}\Gamma$ results in a decidable emptiness problem. Furthermore, we provide a simple, more mechanical (as opposed to algebraic) description of (i) the storage mechanism emerging as the most general decidable case and (ii) a mechanism subsuming the cases we leave open. The model (i) is a new extension of partially blind counter automata (i.e. Petri nets). While the decidability proof employs a reduction to Reinhardt's priority multicounter machines [8], the model (i) seems to be expressively incomparable to Reinhardt's model. The model (ii) is a class of mechanisms whose simplest instance are the pushdown Petri nets and which also subsumes priority multicounter machines.

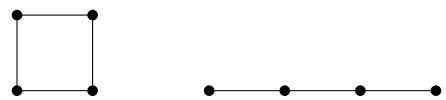
Hence, although the results here essentially combine previous work, the perspective of valence automata allows us to identify two natural storage mechanisms that (i) push the frontier of decidable reachability and (ii) let us interpret PPN and priority multicounter machines as special cases of a more powerful model that might enjoy decidability, respectively.

2 Preliminaries

A *monoid* is a set M together with a binary associative operation such that M contains a neutral element. Unless the monoid at hand warrants a different notation, we will denote the neutral element by 1 and the product of $x, y \in M$ by xy. If X is a set of symbols, X^* denoted the set of words over X. An *alphabet* is a finite set of symbols. The empty word is denoted by $\varepsilon \in X^*$.

Valence automata As a framework for studying which storage mechanisms permit decidability of the reachability problem, we employ valence automata. They

[1] They are not to be confused with the closely related, but different concept of *trace monoids* [2], i.e. monoids of Mazurkiewicz traces, which some authors also call graph monoids.



(a) C_4 (b) P_4

Fig. 1: Graphs C_4 and P_4.

feature a monoid that dictates which computations are valid. Hence, by an appropriate choice of the monoid, valence automata can be instantiated to be equivalent to a concrete automata model with storage. For the purposes of this work, *equivalent* is meant with respect to accepted languages. Therefore, we regard valence automata as language accepting devices.

Let M be a monoid and X an alphabet. A *valence automaton over* M is a tuple $A = (Q, X, M, E, q_0, F)$, in which (i) Q is a finite set of *states*, (ii) E is a finite subset of $Q \times X^* \times M \times Q$, called the set of *edges*, (iii) $q_0 \in Q$ is the *initial state*, and (iv) $F \subseteq Q$ is the set of *final states*. For $q, q' \in Q$, $w, w' \in X^*$, and $m, m' \in M$, we write $(q, w, m) \to_A (q', w', m')$ if there is an edge $(q, v, n, q') \in E$ such that $w' = wv$ and $m' = mn$. The language *accepted* by A is then

$$\mathsf{L}(A) = \{w \in X^* \mid (q_0, \varepsilon, 1) \to_A^* (f, w, 1) \text{ for some } f \in F\}.$$

The class of languages accepted by valence automata over M is denoted by $\mathsf{VA}(M)$. If $\mathcal{M}$ is a class of monoids, we write $\mathsf{VA}(\mathcal{M})$ for $\bigcup_{M \in \mathcal{M}} \mathsf{VA}(M)$.

Graphs A *graph* is a pair $\Gamma = (V, E)$ where V is a finite set and E is a subset of $\{S \subseteq V \mid 1 \leq |S| \leq 2\}$. The elements of V are called *vertices* and those of E are called *edges*. Vertices $v, w \in V$ are *adjacent* if $\{v, w\} \in E$. If $\{v\} \in E$ for some $v \in V$, then v is called a *looped* vertex, otherwise it is *unlooped*. A *subgraph* of Γ is a graph (V', E') with $V' \subseteq V$ and $E' \subseteq E$. Such a subgraph is called *induced (by V')* if $E' = \{S \in E \mid S \subseteq V'\}$, i.e. E' contains all edges from E incident to vertices in V'. By $\Gamma \setminus \{v\}$, for $v \in V$, we denote the subgraph of Γ induced by $V \setminus \{v\}$. By C_4 (P_4), we denote a graph that is a cycle (path) on four vertices; see Figs. 1a and 1b. Moreover, a *clique* is a loop-free graph in which any two distinct vertices are adjacent. Finally, Γ^- denotes the graph obtained from Γ by deleting all loops: We have $\Gamma^- = (V, E^-)$, where $E^- = \{S \in E \mid |S| = 2\}$.

Graph monoids Let A be a (not necessarily finite) set of symbols and R be a subset of $A^* \times A^*$. The pair (A, R) is called a *(monoid) presentation*. The smallest congruence of A^* containing R is denoted by $\equiv_R$ and we will write $[w]_R$ for the congruence class of $w \in A^*$. The *monoid presented by* (A, R) is defined as $A^*/{\equiv_R}$. Note that since we did not impose a finiteness restriction on A, up to isomorphism, every monoid has a presentation. Furthermore, for monoids M_1, M_2 we can find presentations (A_1, R_1) and (A_2, R_2) such that $A_1 \cap A_2 = \emptyset$. We define the *free product* $M_1 * M_2$ to be presented by $(A_1 \cup A_2, R_1 \cup R_2)$. Note that $M_1 * M_2$ is well-defined up to isomorphism. In analogy to the n-fold direct product, we write $M^{(n)}$ for the n-fold free product of M.

A presentation (A, R) in which A is a finite alphabet is a *Thue system*. To each graph $\Gamma = (V, E)$, we associate the Thue system $T_\Gamma = (X_\Gamma, R_\Gamma)$ over the alphabet $X_\Gamma = \{a_v, \bar{a}_v \mid v \in V\}$. R_Γ is defined as

$$R_\Gamma = \{(a_v\bar{a}_v, \varepsilon) \mid v \in V\} \cup \{(xy, yx) \mid x \in \{a_v, \bar{a}_v\},\ y \in \{a_w, \bar{a}_w\},\ \{v, w\} \in E\}.$$

In particular, we have $(a_v\bar{a}_v, \bar{a}_v a_v) \in R_\Gamma$ whenever $\{v\} \in E$. To simplify notation, the congruence $\equiv_{R_\Gamma}$ is then also denoted by $\equiv_\Gamma$. We are now ready to define graph monoids. To each graph Γ, we associate the monoid

$$\mathbb{M}\Gamma \quad = \quad X_\Gamma^*/\equiv_\Gamma.$$

The monoids of the form $\mathbb{M}\Gamma$ are called *graph monoids*.

Let us briefly discuss how to realize storage mechanisms by graph monoids. First, suppose Γ_0 and Γ_1 are disjoint graphs. If Γ is the union of Γ_0 and Γ_1, then $\mathbb{M}\Gamma \cong \mathbb{M}\Gamma_0 * \mathbb{M}\Gamma_1$ by definition. Moreover, if Γ is obtained from Γ_0 and Γ_1 by drawing an edge between each vertex of Γ_0 and each vertex of Γ_1, then $\mathbb{M}\Gamma \cong \mathbb{M}\Gamma_0 \times \mathbb{M}\Gamma_1$.

If Γ consists of one vertex v and has no edges, the only rule in the Thue system is $(a_v\bar{a}_v, \varepsilon)$. In this case, $\mathbb{M}\Gamma$ is also denoted as $\mathbb{B}$ and we will refer to it as the *bicyclic monoid*. The generators a_v and $\bar{a}_v$ are then also written a and $\bar{a}$, respectively. It is not hard to see that $\mathbb{B}$ corresponds to a *partially blind counter*, i.e. one that attains only non-negative values and has to be zero at the end of the computation. Moreover, if Γ consists of one looped vertex, then $\mathbb{M}\Gamma$ is isomorphic to $\mathbb{Z}$ and thus realizes a *blind counter*, which can go below zero and also zero-tested in the end.

If one storage mechanism is realized by a monoid M, then the monoid $\mathbb{B} * M$ corresponds to the mechanism that *builds stacks*: A configuration of this new mechanism consists of a sequence $c_0ac_1 \cdots ac_n$, where $c_0, \ldots, c_n$ are configurations of the mechanism realized by M. We interpret this as a stack with the entries $c_0, \ldots, c_n$. One can open a new stack entry on top (by multiplying $a \in \mathbb{B}$), remove the topmost entry if empty (by multiplying $\bar{a} \in \mathbb{B}$) and operate on the topmost entry using the old mechanism (by multiplying elements from M). In particular, $\mathbb{B} * \mathbb{B}$ describes a pushdown stack with two stack symbols. See [12] for details and more examples.

As a final example, note that if Γ is one edge short of being a clique, then $\mathbb{M}\Gamma \cong \mathbb{B}^{(2)} \times \mathbb{B}^{n-2}$, where n is the number of vertices in Γ. Then, by the observations above, valence automata over $\mathbb{M}\Gamma$ are equivalent to Petri nets with n unbounded places and access to a pushdown stack. Hence, for our purposes, a *pushdown Petri net* is a valence automaton over $\mathbb{B}^{(2)} \times \mathbb{B}^n$ for some $n \in \mathbb{N}$.

3 Results

As a first step, we exhibit graphs Γ for which $\mathsf{VA}(\mathbb{M}\Gamma)$ includes the recursively enumerable languages. Unfortunately, the space restrictions do not permit an inclusion of a proof for the following theorem.

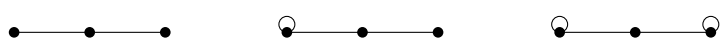
Theorem 1. *Let Γ be a graph such that Γ^- contains C_4 or P_4 as an induced subgraph. Then $\mathsf{VA}(\mathbb{M}\Gamma)$ is the class of recursively enumerable languages. In particular, the emptiness problem is undecidable for valence automata over $\mathbb{M}\Gamma$.*

This unifies and slightly strengthens a few undecidability results concerning valence automata over graph monoids. The case that all vertices are looped was shown by Lohrey and Steinberg [7] (see also the discussion of Theorem 4). Another case appeared in [12].

It is not clear whether Theorem 1 describes all Γ for which $\mathsf{VA}(\mathbb{M}\Gamma)$ exhausts the recursively enumerable languages. For example, as mentioned above, if Γ is one edge short of being a clique, then valence automata over $\mathbb{M}\Gamma$ are pushdown Petri nets. In particular, the emptiness problem for valence automata is equivalent to the reachability problem of this model, for which decidability is a long-standing open question [8]. In fact, it is already open whether reachability is decidable in the case of $\mathbb{B}^{(2)} \times \mathbb{B}$, although Leroux, Sutre, and Totzke have recently made progress on this case [5]. Therefore, characterizing those Γ with a decidable emptiness problem for valence automata over $\mathbb{M}\Gamma$ would very likely settle these open questions[2].

However, we will show that if we steer clear of pushdown Petri nets, we can achieve a characterization. More precisely, we will present a set of graphs that entail the behavior of pushdown Petri nets. Then, we show that among those graphs that do not contain these as induced subgraphs, the absence of P_4 and C_4 already characterizes decidability.

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



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

Of course, the abbreviation 'PPN' refers to 'pushdown Petri nets'. This is justified by the following fact.

Proposition 2. *If Γ is a PPN-graph, then $\mathsf{VA}(\mathbb{M}\Gamma) = \mathsf{VA}(\mathbb{B}^{(2)} \times \mathbb{B})$.*

Transitive forests In order to exploit the absence of P_4 and C_4 as induced subgraphs, we will employ a characterization of such graphs as transitive forests. The *comparability graph* of a tree t is a simple graph with the same vertices as t, but has an edge between two vertices whenever one is a descendent of the other in t. A simple graph is a *transitive forest* if it is the disjoint union of comparability graphs of trees. For an example of a transitive forest, see Fig. 2.

Let DEC denote the smallest isomorphism-closed class of monoids such that

[2] Technically, it is conceivable that there is a decision procedure for each $\mathbb{B}^{(2)} \times \mathbb{B}^n$, but no uniform one that works for all n. However, this seems unlikely.

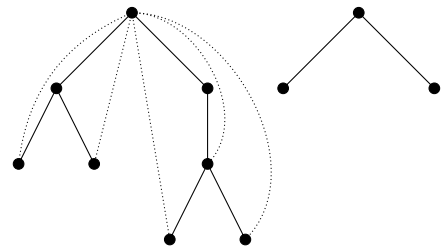


Fig. 2: Example of a transitive forest. The solid edges are part of the trees whose comparability graphs make up the graph.

1. for each $n \geq 0$, we have $\mathbb{B}^n \in \mathsf{DEC}$ and
2. for $M, N \in \mathsf{DEC}$, we also have $M * N \in \mathsf{DEC}$ and $M \times \mathbb{Z} \in \mathsf{DEC}$.

Our main result characterizes those PPN-free Γ for which valence automata over $\mathbb{M}\Gamma$ have a decidable emptiness problem.

Theorem 3. *Let Γ be PPN-free. Then the following conditions are equivalent:*

1. *Emptiness is decidable for valence automata over $\mathbb{M}\Gamma$.*
2. *Γ^- contains neither C_4 nor P_4 as an induced subgraph.*
3. *Γ^- is a transitive forest.*
4. $\mathbb{M}\Gamma \in \mathsf{DEC}$.

Note that this generalizes the fact that emptiness is decidable for pushdown automata (i.e. graphs with no edges) and partially blind multicounter automata (i.e. cliques), or equivalently, reachability in Petri nets.

Note that if Γ has a loop on every vertex, then $\mathbb{M}\Gamma$ is a group. Groups that arise in this way are called *graph groups*. In general, if a monoid M is a group, then emptiness for valence automata over M is decidable if and only if the *rational subset membership problem* is decidable for M [4]. The latter problem asks, given a rational set R over M and an element $m \in M$, whether $m \in R$; see [6] for more information. Therefore, Theorem 3 extends the following result of Lohrey and Steinberg [7], which characterizes those graph groups for which the rational subset membership problem is decidable.

Theorem 4 (Lohrey and Steinberg [7]). *Let Γ be a graph in which every vertex is looped. Then the rational subset membership problem for the group $\mathbb{M}\Gamma$ is decidable if and only if Γ^- is a transitive forest.*

Lohrey and Steinberg show decidability by essentially proving that $\mathsf{VA}(\mathbb{M}\Gamma)$ is semilinear in their case. Here, we extend this argument by showing that in the equivalent cases of Theorem 3, the Parikh images of $\mathsf{VA}(\mathbb{M}\Gamma)$ are those of languages accepted by priority multicounter machines. The latter were introduced and shown to have a decidable reachability problem by Reinhardt [8].

Intuition for decidable cases In order to provide an intuition for those storage mechanisms (not containing a pushdown Petri net) with a decidable emptiness problem, we present an equally expressive class of monoids for which the corresponding storage mechanisms are easier to grasp. Let $\mathsf{SC}^{\pm}$ be the smallest isomorphism-closed class of monoids with

1. for each $n \in \mathbb{N}$, we have $\mathbb{B}^n \in \mathsf{SC}^{\pm}$,
2. for each $M \in \mathsf{SC}^{\pm}$, we also have $\mathbb{B} * M \in \mathsf{SC}^{\pm}$ and $M \times \mathbb{Z} \in \mathsf{SC}^{\pm}$.

Thus, $\mathsf{SC}^{\pm}$ realizes those storage mechanisms that can be constructed from a finite set of *partially blind counters* ($\mathbb{B}^n$) by *building stacks* ($M \mapsto \mathbb{B} * M$) and *adding blind counters* ($M \mapsto M \times \mathbb{Z}$). Then, in fact, the monoids in $\mathsf{SC}^{\pm}$ produce the same languages as those in DEC.

Proposition 5. $\mathsf{VA}(\mathsf{DEC}) = \mathsf{VA}(\mathsf{SC}^{\pm})$.

While our decidability proof for $\mathsf{SC}^{\pm}$ is a reduction to priority multicounter machines, it should be stressed that the mechanisms realized by $\mathsf{SC}^{\pm}$ are quite different from priority counters and very likely not subsumed by them in terms of accepted languages. For example, $\mathsf{SC}^{\pm}$ contains pushdown stacks ($\mathbb{B} * \mathbb{B}$)—if the priority multicounter machines could accept all context-free languages (or even just the semi-Dyck language over two pairs of parentheses), this would easily imply decidability for pushdown Petri nets. Indeed, $\mathsf{SC}^{\pm}$ can even realize stacks where each entry consists of n partially blind counters (since $\mathbb{B} * (\mathbb{B}^n) \in \mathsf{SC}^{\pm}$). On the other hand, priority multicounter machines do not seem to be subsumed by $\mathsf{SC}^{\pm}$ either: After building stacks once, $\mathsf{SC}^{\pm}$ only allows adding *blind* counters (and building stacks again). It therefore seems unlikely that a mechanism in $\mathsf{SC}^{\pm}$ can accept the languages even of a priority 2-counter machine.

Intuition for open cases We also want to provide an intuition for the remaining storage mechanisms, i.e. those defined by monoids $\mathbb{M}\Gamma$ about which Theorems 1 and 3 make no statement. To this end, we describe a class of monoids that are expressively equivalent to these remaining cases. The remaining cases are given by those graphs Γ where Γ^- does not contain C_4 or P_4, but Γ contains a PPN-graph. Let REM denote the class of monoids $\mathbb{M}\Gamma$, where Γ is such a graph. Let SC^{+} be the smallest isomorphism-closed class of monoids with

1. $\mathbb{B}^{(2)} \times \mathbb{B} \in \mathsf{SC}^{+}$ and
2. for each $M \in \mathsf{SC}^{+}$, we also have $\mathbb{B} * M \in \mathsf{SC}^{+}$ and $M \times \mathbb{B} \in \mathsf{SC}^{+}$.

This means, SC^{+} realizes those storage mechanisms that are obtained from *a pushdown stack, together with one partially blind counter* ($\mathbb{B}^{(2)} \times \mathbb{B}$) by the transformations of *building stacks* ($M \mapsto \mathbb{B} * M$) and *adding partially blind counters* ($M \mapsto M \times \mathbb{B}$).

Proposition 6. $\mathsf{VA}(\mathsf{REM}) = \mathsf{VA}(\mathsf{SC}^{+})$.

Of course, SC^{+} generalizes pushdown Petri nets, which correspond to monoids $\mathbb{B}^{(2)} \times \mathbb{B}^n$ for $n \in \mathbb{N}$. Moreover, SC^{+} also subsumes priority multicounter machines

(see p. 10): Every time we build stacks, we can use the new pop operation to realize a zero test on all the counters we have added so far. Let $M_0 = \mathbf{1}$ and $M_{k+1} = \mathbb{B} * (M_k \times \mathbb{B})$. Then, priority k-counter machines correspond to valence automata over M_k where the stack heights never exceed 1.

4 Proof of the characterization

First, we mention existing results that are ingredients to our proofs. A *class of languages* is a collection of languages that contains at least one non-empty language. If $\mathcal{C}$ is a language class such that for languages $L \subseteq X^*$ in $\mathcal{C}$, homomorphisms $h\colon X^* \to Y^*$, $g\colon Z^* \to X^*$, and regular sets $R \subseteq X^*$, we have $h(L) \in \mathcal{C}$, $g^{-1}(L) \in \mathcal{C}$, and $L \cap R \in \mathcal{C}$, we call $\mathcal{C}$ a *full trio*.

Let $\mathcal{C}$ be a class of languages. A $\mathcal{C}$*-grammar* is a quadruple $G = (N, T, P, S)$ where N and T are disjoint alphabets and $S \in N$. P is a finite set of pairs (A, M) with $A \in N$ and $M \subseteq (N \cup T)^*$, $M \in \mathcal{C}$. We write $x \Rightarrow_G y$ if $x = uAv$ and $y = uwv$ for some $u, v, w \in (N \cup T)^*$ and $(A, M) \in P$ with $w \in M$. The *language generated by* G is $\mathsf{L}(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$. The class of all languages that are generated by $\mathcal{C}$-grammars is denoted $\mathsf{Alg}(\mathcal{C})$.

The well-known theorem of Chomsky and Schützenberger [1], expressed in terms of valence automata, states that $\mathsf{VA}(\mathbb{Z} * \mathbb{Z})$ is the class of context-free languages. This formulation, along with a new proof, is due to Kambites [3]. Let Reg and CF denote the class of regular and context-free languages, respectively. Then we have $\mathsf{Reg} = \mathsf{VA}(\mathbf{1})$ and $\mathsf{CF} = \mathsf{Alg}(\mathsf{Reg})$. Here, $\mathbf{1}$ denotes the trivial monoid $\{1\}$. Since furthermore valence automata over $\mathbb{B} * \mathbb{B}$ are equivalent to pushdown automata, we have in summary:

$$\mathsf{CF} = \mathsf{VA}(\mathbb{B} * \mathbb{B}) = \mathsf{Alg}(\mathsf{VA}(\mathbf{1})) = \mathsf{Alg}(\mathsf{CF}) = \mathsf{VA}(\mathbb{Z} * \mathbb{Z}).$$

In order to work with general free products, we use the following result, which expresses the languages in $\mathsf{VA}(M_0 * M_1)$ in terms of $\mathsf{VA}(M_0)$ and $\mathsf{VA}(M_1)$.

Proposition 7 ([12]). *Let M_0 and M_1 be monoids. Then $\mathsf{VA}(M_0 * M_1)$ is included in $\mathsf{Alg}(\mathsf{VA}(M_0) \cup \mathsf{VA}(M_1))$.*

As a partial converse to Proposition 7, we have the following. For a monoid M, we define $R_1(M) = \{x \in M \mid \exists y \in M\colon xy = 1\}$. Observe that the set $R_1(M)$ can be thought of as the storage contents that can be reset.

Proposition 8 ([11]). *For every monoid M, $\mathsf{VA}(\mathbb{B} * \mathbb{B} * M) = \mathsf{Alg}(\mathsf{VA}(M))$. Moreover, if $R_1(M) \neq \{1\}$, then $\mathsf{VA}(\mathbb{B} * M) = \mathsf{Alg}(\mathsf{VA}(M))$.*

Since valence automata over $\mathbb{B} * \mathbb{B}$ are essentially pushdown automata and since $\mathsf{Alg}(\mathsf{VA}(\mathbf{1})) = \mathsf{Alg}(\mathsf{Reg})$, the equality $\mathsf{VA}(\mathbb{B} * \mathbb{B} * M) = \mathsf{Alg}(\mathsf{VA}(M))$ generalizes the equivalence between pushdown automata and context-free grammars.

We will also use the fact that if $\mathsf{VA}(M_i) \subseteq \mathsf{VA}(N_i)$ for $i \in \{0, 1\}$, then $\mathsf{VA}(M_0 \times M_1) \subseteq \mathsf{VA}(N_0 \times N_1)$. This can be deduced from a characterization of $\mathsf{VA}(M_0 \times M_1)$ in terms of $\mathsf{VA}(M_0)$ and $\mathsf{VA}(M_1)$ by Kambites [3].

Proof (Proposition 2). By definition, we have $\mathbb{M}\Gamma \cong \mathbb{B} \times (M_0 * M_1)$, where $M_i \cong \mathbb{B}$ or $M_i \cong \mathbb{Z}$ for $i \in \{0,1\}$. We show that $\mathsf{VA}(M_0 * M_1) = \mathsf{VA}(\mathbb{B} * \mathbb{B})$ in any case. This suffices, since it clearly implies $\mathsf{VA}(\mathbb{M}\Gamma) = \mathsf{VA}(\mathbb{B}^{(2)} \times \mathbb{B})$. If $M_0 \cong M_1 \cong \mathbb{B}$, the equality $\mathsf{VA}(M_0 * M_1) = \mathsf{VA}(\mathbb{B} * \mathbb{B})$ is trivial, so we may assume $M_0 \cong \mathbb{Z}$.

If $M_1 \cong \mathbb{Z}$, then $M_0 * M_1 \cong \mathbb{Z} * \mathbb{Z}$, meaning that $\mathsf{VA}(M_0 * M_1)$ is the class of context-free languages and thus $\mathsf{VA}(M_0 * M_1) = \mathsf{VA}(\mathbb{B} * \mathbb{B})$.

If $M_1 \cong \mathbb{B}$, then $\mathsf{VA}(\mathbb{Z} * \mathbb{B}) = \mathsf{Alg}(\mathsf{VA}(\mathbb{Z}))$ by Proposition 8. Since $\mathsf{VA}(\mathbb{Z})$ is included in the context-free languages, we have $\mathsf{Alg}(\mathsf{VA}(\mathbb{Z})) = \mathsf{VA}(\mathbb{B} * \mathbb{B})$. □

We shall now prove Theorem 3. Note that the implication "1 ⇒ 2" immediately follows from Theorem 1. The implication "2 ⇒ 3" is an old graph-theoretic result of Wolk.

Theorem 9 (Wolk [10]). *A simple graph Γ is a transitive forest if and only if Γ does not contain C_4 or P_4 as an induced subgraph.*

The implication "3 ⇒ 4" is a simple combinatorial observation. An analogous fact is part of Lohrey and Steinberg's proof of Theorem 4.

Lemma 10. *If Γ is PPN-free and Γ^- is a transitive forest, then $\mathbb{M}\Gamma \in \mathsf{DEC}$.*

Proof. Let $\Gamma = (V, E)$. We proceed by induction on $|V|$. Observe that by Theorem 9, every induced subgraph of a transitive forest is again a transitive forest. Since furthermore every induced proper subgraph Δ of Γ is again PPN-free, our induction hypothesis implies $\mathbb{M}\Delta \in \mathsf{DEC}$ for such graphs. If Γ is empty, then $\mathbb{M}\Gamma \cong \mathbf{1} \cong \mathbb{B}^0 \in \mathsf{DEC}$. Hence, we assume that Γ is non-empty. If Γ is not connected, then Γ is the disjoint union of graphs Γ_1, Γ_2, for which $\mathbb{M}\Gamma_1, \mathbb{M}\Gamma_2 \in \mathsf{DEC}$ by induction. Hence, $\mathbb{M}\Gamma \cong \mathbb{M}\Gamma_1 * \mathbb{M}\Gamma_2 \in \mathsf{DEC}$.

Suppose Γ is connected. Since Γ^- is a transitive forest, there is a vertex $v \in V$ that is adjacent to every vertex in $V \setminus \{v\}$. We distinguish two cases.

- If v is a looped vertex, then $\mathbb{M}\Gamma \cong \mathbb{Z} \times \mathbb{M}(\Gamma \setminus v)$, and $\mathbb{M}(\Gamma \setminus v) \in \mathsf{DEC}$ by induction.
- If v is an unlooped vertex, then Γ being PPN-free means that in $\Gamma \setminus v$, any two distinct vertices are adjacent. Hence, $\mathbb{M}\Gamma \cong \mathbb{B}^m \times \mathbb{Z}^n$, where m and n are the number of unlooped and looped vertices in Γ, respectively. Therefore, $\mathbb{M}\Gamma \in \mathsf{DEC}$. □

In light of Theorems 1 and 9 and Lemma 10, it remains to be shown that emptiness is decidable for valence automata over monoids in DEC. This will involve two facts (Theorem 11 and Proposition 12) about the languages arising from monoids in DEC.

The following generalization of Parikh's theorem by van Leeuwen will allow us to exploit our description of free products by algebraic extensions. If X is an alphabet, $X^{\oplus}$ denotes the set of maps $\alpha\colon X \to \mathbb{N}$. The elements of $X^{\oplus}$ are called *multisets*. The *Parikh map* is the map $\Psi\colon X^* \to X^{\oplus}$ where $\Psi(w)(x)$ is the number of occurrences of x in w. By $\mathcal{P}(S)$, we denote the power set of the set

S. A *substitution* is a map $\sigma\colon X \to \mathcal{P}(Y^*)$ and given $L \subseteq X^*$, we write $\sigma(L)$ for the set of all words $v_1 \cdots v_n$, where $v_i \in \sigma(x_i)$, $1 \leq i \leq n$, for $x_1 \cdots x_n \in L$ and $x_1, \ldots, x_n \in X$. If $\sigma(x)$ belongs to $\mathcal{C}$ for each $x \in X$, then σ is a $\mathcal{C}$-*substitution*. The class $\mathcal{C}$ is said to be *substitution closed* if $\sigma(L) \in \mathcal{C}$ for every member L of $\mathcal{C}$ and every $\mathcal{C}$-substitution σ.

Theorem 11 (van Leeuwen [9]). *For each substitution closed full trio $\mathcal{C}$, we have $\Psi(\mathsf{Alg}(\mathcal{C})) = \Psi(\mathcal{C})$.*

For $\alpha, \beta \in X^{\oplus}$, let $\alpha + \beta \in X^{\oplus}$ be defined by $(\alpha + \beta)(x) = \alpha(x) + \beta(x)$. With this operation, $X^{\oplus}$ is a monoid. For a subset $S \subseteq X^{\oplus}$, we write $S^{\oplus}$ for the smallest submonoid of $X^{\oplus}$ containing S. A subset of the form $\mu + F^{\oplus}$ for $\mu \in X^{\oplus}$ and a finite $F \subseteq X^{\oplus}$ is called *linear*. A finite union of linear sets is called *semilinear*. By $\mathsf{SLI}(\mathcal{C})$ we denote the class of languages $h(L \cap \Psi^{-1}(S))$, where $h\colon X^* \to Y^*$ is a morphism, L belongs to $\mathcal{C}$, and $S \subseteq X^{\oplus}$ is semilinear.

Proposition 12 ([11]). *For each monoid M, $\mathsf{SLI}(\mathsf{VA}(M)) = \bigcup_{i \geq 0} \mathsf{VA}(M \times \mathbb{Z}^n)$.*

We will prove decidability for DEC by reducing the problem to the reachability problem of priority multicounter machines, whose decidability has been established by Reinhardt [8]. Priority multicounter machines are an extension of Petri nets with one inhibitor arc. Intuitively, a priority multicounter machine is a partially blind multicounter machine with the additional capability of restricted zero tests: The counters are numbered from 1 to k and for each $\ell \in \{1, \ldots, k\}$, there is a zero test instruction that checks whether counters 1 through ℓ are zero. Let us define priority multicounter machines formally.

A *priority k-counter machine* is a tuple $A = (Q, X, E, q_0, F)$, where (i) X is an alphabet, (ii) Q is a finite set of states, (iii) E is a finite subset of $Q \times X^* \times \{0, \ldots, k\} \times \mathbb{Z}^k \times Q$, and its elements are called *edges* or *transitions*, (iv) $q_0 \in Q$ is the *initial state*, and (v) $F \subseteq Q$ is the set of *final states*. For triples (q, u, μ) and (q', u', μ') with $q, q' \in Q$ and $\mu, \mu' \in \mathbb{N}^k$, with $\mu = (m_1, \ldots, m_k)$, we write $(q, u, \mu) \to_A (q', u', \mu')$ if for some $(q, x, \ell, \nu, q') \in E$, we have $u' = ux$, $\mu' = \mu + \nu$, and $m_i = 0$ for $1 \leq i \leq \ell$. The *language accepted by A* is defined as

$$\mathsf{L}(A) = \{w \in X^* \mid (q_0, \varepsilon, 0) \to_A^* (f, w, 0) \text{ for some } f \in F\}.$$

A *priority multicounter machine* is a priority k-counter machine for some $k \in \mathbb{N}$. The class of languages accepted by priority multicounter machines is denoted by Prio. Reinhardt has shown that the reachability problem for priority multicounter machines is decidable [8], which can be reformulated as follows.

Theorem 13 (Reinhardt [8]). *Emptiness is decidable for priority multicounter machines.*

The idea of the proof of "4 $\Rightarrow$ 1" is, given a valence automaton over some $M \in \mathsf{DEC}$, to construct a Parikh-equivalent priority multicounter machine. This construction makes use of the following simple fact. A full trio $\mathcal{C}$ is said to be *Presburger closed* if $\mathsf{SLI}(\mathcal{C}) \subseteq \mathcal{C}$.

Lemma 14. Prio *is a Presburger closed full trio and closed under substitutions.*

Proof. The fact that Prio is a full trio can be shown by standard automata constructions. Given a priority multicounter machine A and a semilinear set $S \subseteq X^{\oplus}$, we add $|X|$ counters to A that ensure that the input is contained in $\mathsf{L}(A) \cap \Psi^{-1}(S)$. This proves that Prio is Presburger closed.

Suppose $\sigma\colon X \to \mathcal{P}(Y^*)$ is a Prio-substitution. Furthermore, let A be a priority k-counter machine and let $\sigma(x)$ be given by a priority ℓ-counter machine for each $x \in X$. We construct a priority $(\ell + k)$-counter machine B from A by adding ℓ counters. B simulates A on counters $\ell+1, \ldots, \ell+k$. Whenever A reads x, B uses the first ℓ counters to simulate the priority ℓ-counter machine for $\sigma(x)$. Using the zero test on the first ℓ counters, it makes sure that the machine for $\sigma(x)$ indeed ends up in a final configuration. Then clearly $\mathsf{L}(B) = \sigma(\mathsf{L}(A))$. $\square$

In order to show that every $L \in \mathsf{VA}(M)$ for $M \in \mathsf{DEC}$ has a Parikh equivalent in Prio, we use Proposition 12 and Proposition 7. By induction with respect to the definition of DEC, it suffices to prove that

$$\Psi(\mathsf{VA}(M)), \Psi(\mathsf{VA}(N)) \subseteq \Psi(\mathsf{Prio}) \quad \text{implies} \quad \begin{aligned} &\Psi(\mathsf{VA}(M \times \mathbb{Z})) \subseteq \Psi(\mathsf{Prio}) \text{ and} \\ &\Psi(\mathsf{VA}(M * N)) \subseteq \Psi(\mathsf{Prio}). \end{aligned}$$

According to Proposition 12 and Proposition 7, this boils down to showing that we have $\Psi(\mathsf{SLI}(\mathsf{Prio})) \subseteq \Psi(\mathsf{Prio})$ and $\Psi(\mathsf{Alg}(\mathsf{Prio})) \subseteq \Psi(\mathsf{Prio})$. The former is a consequence of Lemma 14 and the latter is an instance of Theorem 11.

Lemma 15. *We have the effective inclusion* $\Psi(\mathsf{VA}(\mathsf{DEC})) \subseteq \Psi(\mathsf{Prio})$*. More precisely, given* $M \in \mathsf{DEC}$ *and* $L \in \mathsf{VA}(M)$*, one can construct an* $L' \in \mathsf{Prio}$ *with* $\Psi(L') = \Psi(L)$*.*

Proof. We proceed by induction with respect to the definition of DEC. In the case $M = \mathbb{B}^n$, we have $\mathsf{VA}(M) \subseteq \mathsf{Prio}$, because priority multicounter machines generalize partially blind multicounter machines.

Suppose $M = N \times \mathbb{Z}$ and $\Psi(\mathsf{VA}(N)) \subseteq \Psi(\mathsf{Prio})$ and let $L \in \mathsf{VA}(M)$. By Proposition 12, we have $L = h(K \cap \Psi^{-1}(S))$ for some semilinear set S, a morphism h, and $K \in \mathsf{VA}(N)$. Hence, there is a $\bar{K} \in \mathsf{Prio}$ with $\Psi(\bar{K}) = \Psi(K)$. With this, we have $\Psi(L) = \Psi(h(\bar{K} \cap \Psi^{-1}(S)))$ and since Prio is Presburger closed, we have $h(\bar{K} \cap \Psi^{-1}(S)) \in \mathsf{Prio}$ and thus $\Psi(L) \in \Psi(\mathsf{Prio})$.

Suppose $M = M_0 * M_1$ and $\Psi(\mathsf{VA}(M_i)) \subseteq \Psi(\mathsf{Prio})$ for $i \in \{0, 1\}$. Let L be a member of $\mathsf{VA}(M)$. According to Proposition 7, this means L belongs to $\mathsf{Alg}(\mathsf{VA}(M_0) \cup \mathsf{VA}(M_1))$. Since $\Psi(\mathsf{VA}(M_0) \cup \mathsf{VA}(M_1)) \subseteq \Psi(\mathsf{Prio})$, we can construct a Prio-grammar G with $\Psi(\mathsf{L}(G)) = \Psi(L)$. By Theorem 11 and Lemma 14, this implies $\Psi(L) \in \Psi(\mathsf{Prio})$. $\square$

The following lemma is a direct consequence of Lemma 15 and Theorem 13: Given a valence automaton over M with $M \in \mathsf{DEC}$, we construct a priority multicounter machine accepting a Parikh-equivalent language. The latter can then be checked for emptiness.

Lemma 16. *For each $M \in \mathsf{DEC}$, the emptiness problem for valence automata over M is decidable.*

This completes the proof of Theorem 3. Let us now prove the expressive equivalences, Propositions 5 and 6.

Proof (Proposition 5). Since $\mathsf{SC}^{\pm} \subseteq \mathsf{DEC}$, the inclusion "$\supseteq$" is immediate. We show by induction with respect to the definition of DEC that for each $M \in \mathsf{DEC}$, there is an $M' \in \mathsf{SC}^{\pm}$ with $\mathsf{VA}(M) \subseteq \mathsf{VA}(M')$. This is trivial if $M = \mathbb{B}^n$, so suppose $\mathsf{VA}(M) \subseteq \mathsf{VA}(M')$ and $\mathsf{VA}(N) \subseteq \mathsf{VA}(N')$ for $M, N \in \mathsf{DEC}$ and $M', N' \in \mathsf{SC}^{\pm}$. Observe that by induction on the definition of $\mathsf{SC}^{\pm}$, one can show that there is a common $P \in \mathsf{SC}^{\pm}$ with $\mathsf{VA}(M') \subseteq \mathsf{VA}(P)$ and $\mathsf{VA}(N') \subseteq \mathsf{VA}(P)$. Of course, we may assume that $R_1(P) \neq \{1\}$. Then we have

$$\begin{aligned}\mathsf{VA}(M * N) &\subseteq \mathsf{Alg}(\mathsf{VA}(M) \cup \mathsf{VA}(N)) \subseteq \mathsf{Alg}(\mathsf{VA}(M') \cup \mathsf{VA}(N')) \\ &\subseteq \mathsf{Alg}(\mathsf{VA}(P)) = \mathsf{VA}(\mathbb{B} * P),\end{aligned}$$

in which the first inclusion is due to Proposition 7 and the equality in the end is provided by Proposition 8. Since $\mathbb{B} * P \in \mathsf{SC}^{\pm}$, this completes the proof for $M * N$. Moreover, $\mathsf{VA}(M) \subseteq \mathsf{VA}(M')$ implies $\mathsf{VA}(M \times \mathbb{Z}) \subseteq \mathsf{VA}(M' \times \mathbb{Z})$ and we have $M' \times \mathbb{Z} \in \mathsf{SC}^{\pm}$. □

Proof (Proposition 6). By induction, it is easy to see that each $M \in \mathsf{SC}^{+}$ is isomorphic to some $\mathbb{M}\Gamma$ where Γ contains a PPN-graph and Γ^{-} is a transitive forest. By Theorem 9, this means Γ^{-} contains neither C_4 nor P_4. This proves the inclusion "$\supseteq$".

Because of Theorem 9, for the inclusion "$\subseteq$", it suffices to show that if Γ^{-} is a transitive forest, then there is some $M \in \mathsf{SC}^{+}$ with $\mathsf{VA}(\mathbb{M}\Gamma) \subseteq \mathsf{VA}(M)$. We prove this by induction on the number of vertices in $\Gamma = (V, E)$. As in the proof of Lemma 10, we may assume that for every induced proper subgraph Δ of Γ, we find an $M \in \mathsf{SC}^{+}$ with $\mathsf{VA}(\mathbb{M}\Gamma) \subseteq \mathsf{VA}(M)$. If Γ is empty, then $\mathbb{M}\Gamma \cong \mathbf{1}$ and $\mathsf{VA}(\mathbb{M}\Gamma) \subseteq \mathsf{VA}(\mathbb{B}^{(2)} \times \mathbb{B})$. Hence, we may assume that Γ is non-empty.

If Γ is not connected, then $\Gamma = \Gamma_1 \uplus \Gamma_2$ with graphs Γ_1, Γ_2 such that there are $M_1, M_2 \in \mathsf{SC}^{+}$ with $\mathsf{VA}(\mathbb{M}\Gamma_i) \subseteq \mathsf{VA}(M_i)$ for $i \in \{1, 2\}$. By induction with respect to the definition of SC^{+}, one can show that there is a common $N \in \mathsf{SC}^{+}$ with $\mathsf{VA}(M_i) \subseteq \mathsf{VA}(N)$ for $i \in \{1, 2\}$. Since then $R_1(N) \neq \{1\}$, we have

$$\begin{aligned}\mathsf{VA}(\mathbb{M}\Gamma) &= \mathsf{VA}(\mathbb{M}\Gamma_1 * \mathbb{M}\Gamma_2) \subseteq \mathsf{Alg}(\mathsf{VA}(\mathbb{M}\Gamma_1) \cup \mathsf{VA}(\mathbb{M}\Gamma_2)) \\ &\subseteq \mathsf{Alg}(\mathsf{VA}(M_1) \cup \mathsf{VA}(M_2)) \subseteq \mathsf{Alg}(\mathsf{VA}(N)) = \mathsf{VA}(\mathbb{B} * N)\end{aligned}$$

and $\mathbb{B} * N \in \mathsf{SC}^{+}$ as in the proof of Proposition 5.

Suppose Γ is connected. Since Γ^{-} is a transitive forest, there is a vertex $v \in V$ that is adjacent to every vertex in $V \setminus \{v\}$. By induction, there is an $M \in \mathsf{SC}^{+}$ with $\mathsf{VA}(\mathbb{M}(\Gamma \setminus v)) \subseteq \mathsf{VA}(M)$. Depending on whether v is looped or not, we have $\mathbb{M}\Gamma \cong \mathbb{M}(\Gamma \setminus v) \times \mathbb{Z}$ or $\mathbb{M}\Gamma \cong \mathbb{M}(\Gamma \setminus v) \times \mathbb{B}$. Since $\mathsf{VA}(\mathbb{Z}) \subseteq \mathsf{VA}(\mathbb{B} \times \mathbb{B})$ (one blind counter can easily be simulated by two partially blind counters), this yields $\mathsf{VA}(\mathbb{M}\Gamma) \subseteq \mathsf{VA}(\mathbb{M}(\Gamma \setminus v) \times \mathbb{B} \times \mathbb{B}) \subseteq \mathsf{VA}(M \times \mathbb{B} \times \mathbb{B})$ and the fact that $M \times \mathbb{B} \times \mathbb{B} \in \mathsf{SC}^{+}$ completes the proof. □

Conclusion Of course, an intriguing open question is whether the storage mechanisms corresponding to SC^+ have a decidable reachability problem. First, since their simplest instance are pushdown Petri nets, this extends the open question concerning the latter's reachability. Second, they subsume the priority multi-counter machines of Reinhardt. This makes them a candidate for being a quite powerful model for which reachability might be decidable.

Observe that if these storage mechanisms turn out to exhibit decidability, this would mean that Lohrey and Steinberg's characterization (Theorem 4) remains true for all graph monoids. This can be interpreted as evidence for decidability.

Acknowledgements The author is grateful to the anonymous referees, whose helpful comments improved the presentation of this work.

References

[1] J. Berstel. *Transductions and Context-Free Languages*. Stuttgart: Teubner, 1979.

[2] V. Diekert and G. Rozenberg, eds. *The Book of Traces*. Singapore: World Scientific, 1995.

[3] M. Kambites. "Formal Languages and Groups as Memory". In: *Commun. Algebra* 37 (1 2009), pp. 193–208.

[4] M. Kambites, P. V. Silva, and B. Steinberg. "On the rational subset problem for groups". In: *J. Algebra* 309 (2007), pp. 622–639.

[5] J. Leroux, G. Sutre, and P. Totzke. "On the Coverability Problem for Pushdown Vector Addition Systems in One Dimension". In: *Proc. of ICALP 2015*.

[6] M. Lohrey. "The rational subset membership problem for groups: A survey". To appear.

[7] M. Lohrey and B. Steinberg. "The submonoid and rational subset membership problems for graph groups". In: *J. Algebra* 320.2 (2008), pp. 728–755.

[8] K. Reinhardt. "Reachability in Petri Nets with Inhibitor Arcs". In: *Proc. of RP 2008*.

[9] J. van Leeuwen. "A generalisation of Parikh's theorem in formal language theory". In: *Proc. of ICALP 1974*.

[10] E. S. Wolk. "A Note on 'The Comparability Graph of a Tree'". In: *P. Am. Math. Soc.* 16.1 (1965), pp. 17–20.

[11] G. Zetzsche. "Computing downward closures for stacked counter automata". In: *Proc. of STACS 2015*.

[12] G. Zetzsche. "Silent Transitions in Automata with Storage". In: *Proc. of ICALP 2013*.