Automata Column: The Complexity of Reachability in Vector Addition Systems

Sylvain Schmitz

Automata Column: The Complexity of Reachability in Vector Addition Systems

SYLVAIN SCHMITZ, LSV, ENS Cachan & CNRS & INRIA, Université Paris-Saclay

The program of the 30th Symposium on Logic in Computer Science held in 2015 in Kyoto included two contributions on the computational complexity of the *reachability problem* for vector addition systems: Blondin, Finkel, Göller, Haase, and McKenzie [2015] attacked the problem by providing the first tight complexity bounds in the case of dimension 2 systems with states, while Leroux and Schmitz [2015] proved the first complexity upper bound in the general case. The purpose of this column is to present the main ideas behind these two results, and more generally survey the current state of affairs.

1. INTRODUCTION

Vector addition systems with states (VASS), or equivalently Petri nets, find a wide range of applications in the modelling of concurrent, chemical, biological, or business processes. Maybe more importantly for this column, their algorithmics, and in particular the decidability of their *reachability problem* [Mayr 1981; Kosaraju 1982; Lambert 1992; Leroux 2011], is the cornerstone of many decidability results in logic, automata, verification, etc.—see Section 5 for a few examples.

In spite of its importance, fairly little is known about the computational complexity of the reachability problem. Regarding the general case, the inclusive surveys on the complexity of decision problems on VASS by Esparza and Nielsen [1994] and Esparza [1998] could only point to the EXPSPACE lower bound of Lipton [1976] and to the fact that the running time of the known algorithms is not primitive recursive: *no* complexity upper bound was known, besides decidability first proven in 1981 by Mayr. When turning to restricted versions of the problem, the 2-dimensional case was only known to be in 2-EXP [Howell, Rosier, Huynh, and Yen 1986] and NP-hard [Rosier and Yen 1986].

This state of affair has very recently improved with two articles:

— Leroux and Schmitz [2015] have shown that reachability has a 'cubic Ackermann' upper bound, i.e. is in F_{ω^3}, by analysing the complexity of the classical algorithm developed and refined by Mayr [1981], Kosaraju [1982], and Lambert [1992]. Here, F_{ω^3} is a non primitive-recursive complexity class, but among the lower multiply-recursive ones. The main ingredients for this analysis are the fast-growing complexity bounds for termination proofs by well-quasi-orders and ordinal ranking functions from [Figueira et al. 2011; Schmitz 2014].

— Blondin, Finkel, Göller, Haase, and McKenzie [2015] have shown that reachability in 2-dimensional VASS is PSPACE-complete by a careful analysis of the complexity of the 'flattenings' of Leroux and Sutre [2004] for the upper bound, and by applying recent results on bounded one-counter automata by Fearnley and Jurdziński [2015] for the lower bound.

Organisation of the Column. The main focus of the column is the complexity of the algorithm of Mayr [1981], Kosaraju [1982], and Lambert [1992]. Section 3 presents it in an informal manner on an example before explaining the main points of its complexity analysis following Leroux and Schmitz [2015].

This cubic Ackermann upper bound leaves a considerable gap with the EXPSPACE lower bound of Lipton [1976]. Rather than attacking this complexity gap directly, it makes sense to try to obtain tight complexity bounds on restrictions of the general reachability problem, and we shall see one such restriction in Section 4: the 2-dimensional case and its tight PSPACE-completeness proven by Blondin et al. [2015].

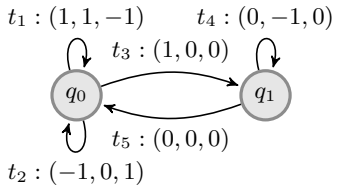

Fig. 1: A 3-dimensional VASS.

The last two sections of the column are more in the spirit of a survey. To better emphasise the importance of the reachability problem, Section 5 provides a glimpse of the many problems known to be interreducible with reachability in VASS. Section 6 finally presents a small selection of VASS extensions and what is known about their reachability problems, pointing to several open problems.

2. VECTOR ADDITION SYSTEMS WITH STATES

For the needs of the automata column, it is natural to first present vector addition systems *with states* (VASS) [Hopcroft and Pansiot 1979], which are essentially finite-state transition systems with d-dimensional vectors of weights in $\mathbb{Z}^d$ attached to their transitions. Formally, a VASS is a tuple $\mathcal{V} = \langle Q, d, T \rangle$ where Q is a finite set of 'control' states, d in $\mathbb{N}$ is a non-negative dimension, and $T \subseteq Q \times \mathbb{Z}^d \times Q$ is a finite set of transitions.

Semantics and Runs. The operational semantics of such a system is captured by an infinite transition system $\mathcal{S}_\mathcal{V}$ over the set of *configurations* $Confs_\mathcal{V} \stackrel{\text{def}}{=} Q \times \mathbb{N}^d$, with a *step* $(q, \boldsymbol{u}) \xrightarrow{t}_\mathcal{V} (q', \boldsymbol{u} + \boldsymbol{a})$ defined whenever $t = (q, \boldsymbol{a}, q')$ belongs to T; note that $\boldsymbol{u} + \boldsymbol{a}$ must belong to $\mathbb{N}^d$ for such a step to be possible. A *run* from a configuration c_0 to a configuration c_ℓ is a finite sequence of steps $c_0 \xrightarrow{t_1}_\mathcal{V} c_1 \xrightarrow{t_2}_\mathcal{V} c_2 \cdots c_{\ell-1} \xrightarrow{t_\ell}_\mathcal{V} c_\ell$, which can also be written $c_0 \xrightarrow{t_1 \cdots t_\ell}_\mathcal{V} c_\ell$. Finally, let us write $c_0 \to^*_\mathcal{V} c_\ell$ if there exists a finite sequence of transitions $\sigma \in T^*$ such that $c_0 \xrightarrow{\sigma}_\mathcal{V} c_\ell$.

Reachability. The *reachability problem* refers to reachability in the infinite system $\mathcal{S}_\mathcal{V}$:

input: a VASS $\mathcal{V}$ and two configurations c and c' in $Confs_\mathcal{V}$,
question: can c reach c', i.e. does $c \to^*_\mathcal{V} c'$?

This problem was famously shown to be decidable by Mayr [1981], Kosaraju [1982], Lambert [1992], and Leroux [2011] (see Section 3 for more details):

THEOREM 2.1 (DECIDABILITY THEOREM). *Reachability in VASS is decidable.*

Example 2.2. Consider for instance the 3-dimensional VASS of Figure 1 with $Q \stackrel{\text{def}}{=} \{q_0, q_1\}$ and $T \stackrel{\text{def}}{=} \{t_1, t_2, t_3, t_4, t_5\}$. One can check that $(q_0, 1, 0, 1)$ reaches $(q_1, 2, 2, 1)$, for instance by the run

$$(q_0, 1, 0, 1) \xrightarrow{t_1} (q_0, 2, 1, 0) \xrightarrow{t_2} (q_0, 1, 1, 1) \xrightarrow{t_1} (q_0, 2, 2, 0) \xrightarrow{t_2} (q_0, 1, 2, 1) \xrightarrow{t_3} (q_1, 2, 2, 1)\ . \quad (1)$$

This is just one example of a run witnessing reachability; observe that any sequence of transitions in $\{t_1 t_2, t_2 t_1\}^{n+2} t_3 t_4^n$ for $n \geq 0$ would similarly do.

Binary Encoding. Regarding complexity, one typically assumes a binary encoding of the integers of a VASS and of the source and target configurations: let $\|\boldsymbol{a}\| \stackrel{\text{def}}{=}$

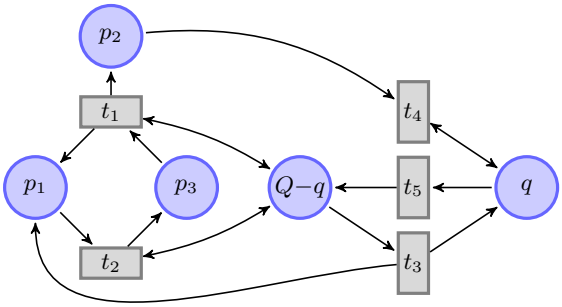

Fig. 2: A Petri net equivalent to the VASS of Figure 1.

$\max_{1\leq i\leq d} |\boldsymbol{a}(i)|$ denote the infinity norm of a vector $\boldsymbol{a}$ in $\mathbb{Z}^d$; then $\|T\| \stackrel{\text{def}}{=} \max_{(q,\boldsymbol{a},q')\in T} \|\boldsymbol{a}\|$ can be exponential in the size of a VASS $\mathcal{V} = \langle Q, d, T\rangle$. The choice of a binary rather than a unary encoding has no impact in the general case—because there is a LOGSPACE reduction to the case where $T \subseteq Q \times \{-1, 0, 1\}^d \times Q$ (at the expense of increasing the dimension) and $c = (q, \mathbf{0})$ and $c' = (q', \mathbf{0})$ for some states q, q'—, but will be important in Section 4 for the 2-dimensional case.

2.1. Closely Related Models

Historically, VASS do not seem to have been studied before the works of Greibach [1978, see Section 5.1] and Hopcroft and Pansiot [1979]. Nevertheless, equivalent models had been investigated before, in particular the *Petri nets* of Petri [1962] and *vector addition systems* (VAS) of Karp and Miller [1969]. The absence of explicit control states makes these two classes of models rather convenient for the modelling of concurrent or distributed systems.

2.1.1. Petri Nets. A Petri net is a tuple $\mathcal{N} = \langle P, T, W\rangle$ where P is a finite set of *places*, T is a finite set of *transitions*, and $W\colon (P \times T) \cup (T \times P) \to \mathbb{N}$ is a (weighted) *flow* function. It defines a transition system with configurations in $\mathbb{N}^P$—i.e. multisets of places, also called *markings*—and steps $m \xrightarrow{t} m'$ whenever $m(p) \geq W(p,t)$ and $m'(p) = m(p) - W(p,t) + W(t,p)$ for all p in P. A Petri net can be encoded as an equivalent $|P|$-dimensional VASS with $|T|+1$ states, and conversely a d-dimensional VASS can be encoded as an equivalent Petri net with $d+2$ places (see Figure 2 for the result of this construction on the VASS of Figure 1, where places are depicted as circles, transitions as rectangles, and flows as arrows)—'equivalence' here should be understood as far as the decision problems like reachability are concerned.

2.1.2. Vector Addition Systems. A VAS is a pair $\langle d, \boldsymbol{A}\rangle$ where $\boldsymbol{A}$ is a finite subset of *actions* in $\mathbb{Z}^d$ [Karp and Miller 1969]. It defines a transition system with configurations $\boldsymbol{u}$ in $\mathbb{N}^d$ and steps $\boldsymbol{u} \to \boldsymbol{u} + \boldsymbol{a}$ for $\boldsymbol{a}$ in $\boldsymbol{A}$, again implicitly checking that $\boldsymbol{u} + \boldsymbol{a} \geq \mathbf{0}$. Put differently, a VAS can be seen as a VASS with a singleton state set. Conversely, the finite control of a d-dimensional VASS can be encoded in an equivalent VAS by increasing the system's dimension to $d+3$ [Hopcroft and Pansiot 1979, Lemma 2.1].

3. DECIDING REACHABILITY

Considered as one of the great achievements of theoretical computer science, the seminal 1981 decidability proof for the reachability problem by Mayr [1981] is the culmination of more than a decade of research into the topic, and builds notably on an

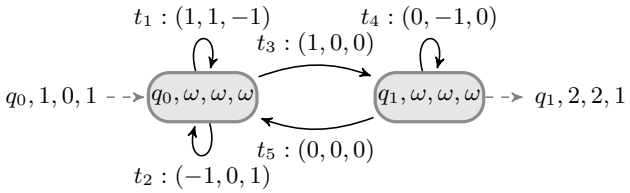

Fig. 3: The initial marked witness graph sequence $\xi_0 = M_0$.

incomplete proof by Sacerdote and Tenney [1977]. This proof has been simplified two times since: one year later by Kosaraju [1982], and another ten years later by Lambert [1992]. At the heart of these three proofs lies a *decomposition technique*, which is called the Kosaraju-Lambert-Mayr-Sacerdote-Tenney (KLMST) decomposition. In a nutshell, the KLMST decomposition defines both

— a structure (resp. *regular constraint graphs* for Mayr, *generalised VASS* for Kosaraju, and *marked graph-transition sequences* for Lambert) and
— a condition for this structure to represent in some way the set of all runs witnessing reachability (resp. *consistent marking*, the θ *condition*, and the *perfect condition*).

The algorithms advanced by Mayr, Kosaraju, and Lambert compute this decomposition by successive refinements of the structure until the condition is fulfilled, by which time the existence of a run becomes trivial.

The KLMST decomposition has also been employed by Leroux [2010] to derive a new, very simple algorithm for reachability based on Presburger inductive invariants. Leroux [2011] then re-proved the correctness of this new algorithm *without* referring to the KLMST decomposition, yielding an independent, compact self-contained decidability proof for VASS reachability.

In the following we will however focus on the 'classical' decomposition algorithm, and present the following result from [Leroux and Schmitz 2015]:

THEOREM 3.1 (UPPER BOUND THEOREM). *Reachability in VASS is in* F_{ω^3}.

We will see in Section 3.2 what is F_{ω^3}. But let us first have a look at the KLMST decomposition algorithm.

3.1. An Example of a KLMST Decomposition

The reader is referred to the original article, and to the excellent accounts by Müller [1985] and Reutenauer [1990] for examples and details on the KLMST decomposition as defined by Kosaraju [1982]. Here we shall keep the description at an informal level, and see how the decomposition algorithm works in the case of Example 2.2 without entering its details.

3.1.1. Marked Witness Graph Sequences. Let us first complete $\mathbb{N}$ with a top element ω and write $\mathbb{N}_\omega \stackrel{\text{def}}{=} \mathbb{N} \uplus \{\omega\}$ for the result; also let $\omega + z = z + \omega = \omega$ for all z in $\mathbb{Z}$.

A *witness graph* is a finite strongly connected directed graph $G = (S, E)$ with vertices $S \subseteq Q \times \mathbb{N}_\omega^d$, and labelled edges $E \subseteq S \times T \times S$, such that the edge labels from T are consistent with the vertices from S. This means that, if (s, t, s') is an edge in E with transition $t = (q, \boldsymbol{a}, q')$ from T as label, then $s = (q, \boldsymbol{u})$ for some $\boldsymbol{u}$ in $\mathbb{N}_\omega^d$ and $s' = (q', \boldsymbol{u} + \boldsymbol{a})$. Note that these conditions together imply that all the vertices in the graph share the same set $I \subseteq \{1, \ldots, d\}$ of ω-components.

A *marked witness graph* $M = (G, c^{\text{in}}, s^{\text{in}}, c^{\text{out}}, s^{\text{out}})$ is further endowed with distinguished input and output vertices s^{in} and s^{out} from S, along with input and output

constraints c^{in} and c^{out} taken from $Q \times \mathbb{N}^d_\omega$, such that for all $1 \leq i \leq d$, $s^{\text{in}}(i) \neq \omega$ implies $c^{\text{in}}(i) = s^{\text{in}}(i)$, and similarly for the output vertex and constraint. In other words, s^{in} and c^{in} agree on their finite components. This entails that I^{in} the set of ω-components of c^{in} is a subset of I the set of ω-components of s^{in}, and similarly $I^{\text{out}} \subseteq I$.

Finally, a *marked witness graph sequence* ξ is a sequence

$$\xi = M_0, t_1, M_1, \ldots, t_k, M_k \tag{2}$$

that alternates between marked witness graphs $M_0, \ldots, M_k$ and transitions $t_1, \ldots, t_k$ taken from T. Let us write $M_j = (G_j, c^{\text{in}}_j, s^{\text{in}}_j, c^{\text{out}}_j, s^{\text{out}}_j)$ and $t_j = (q_j, \boldsymbol{a}_j, q'_j)$ for all j. It is also required that, for all $1 \leq j \leq k$, $c^{\text{out}}_{j-1} = (q_j, \boldsymbol{u}_j)$ for some $\boldsymbol{u}_j$ and $c^{\text{in}}_j = (q'_j, \boldsymbol{u}'_j)$ for some $\boldsymbol{u}'_j$. In such a sequence, c^{in}_0 is the *source* and c^{out}_k is the *target*.

Figure 3 displays a marked witness graph for the VASS of Example 2.2, with input constraint $(q_0, 1, 0, 1)$ on the input vertex $(q_0, \omega, \omega, \omega)$ and output constraint $(q_1, 2, 2, 1)$ on the output vertex $(q_1, \omega, \omega, \omega)$.

3.1.2. The Decomposition Algorithm. The KLMST decomposition algorithm builds a sequence $\Xi_0, \Xi_1, \Xi_2, \ldots$ of finite sets of marked witness graph sequences. At step n, it checks whether all the sequences ξ in Ξ_n are *perfect* (in the sense of Lambert [1992], or equivalently fulfil the θ-condition of Kosaraju [1982]) and stops if this is the case; then either Ξ_n is empty and the algorithm answers 'not reachable', or Ξ_n is not empty and the algorithm answers 'reachable'.

If however some sequence ξ from Ξ_n is not perfect, then it is *decomposed* into a finite set of marked witness graph sequences $\mathrm{dec}(\xi)$—which is possibly empty. Then we let

$$\Xi_{n+1} \stackrel{\text{def}}{=} (\Xi \setminus \{\xi\}) \cup \mathrm{dec}(\xi) \tag{3}$$

and the algorithm proceeds to the next step.

The perfectness condition comprises two sub-conditions, along with the corresponding ways of decomposing marked witness graph sequences when the sub-conditions are violated. We are going to illustrate these two sub-conditions in the upcoming §3.1.3 and §3.1.4, in the case of Example 2.2, and starting from $\Xi_0 = \{\xi_0\}$.

3.1.3. Flow Constraints. Consider a path from the source to the target in the graph of Figure 3: denoting by z_j the number of times transition t_j is used along this path for $j \in \{1, \ldots, 5\}$, we can see that

$$z_3 = z_5 + 1 \ . \tag{4}$$

Consider now a run in the VASS of Figure 1, which follows a path in the marked witness graph of Figure 3 from $(q_0, 1, 0, 1)$ to $(q_1, 2, 2, 1)$, i.e. with overall effect $(1, 2, 0)$. Then, considering the effect of these transitions for each coordinate i in $\{1, 2, 3\}$,

$$\begin{aligned} z_1 + z_3 &= z_2 + 1 \ , \\ z_1 &= z_4 + 2 \ , \\ z_1 &= z_2 \ . \end{aligned} \tag{5}$$

The system of equations (4–5) requires $z_3 = 1$ and $z_5 = 0$; z_1, z_2 and z_4 are on the other hand unbounded.

This shows that the marked witness graph sequence ξ_0 of Figure 3 is too permissive, allowing to follow paths that do not bring the source of the sequence to its target. We therefore decompose it, using the fact that t_3 must be employed exactly once and that t_5 is never employed: $\mathrm{dec}(\xi_0) = \{\xi_1\}$ where the new sequence ξ_1 is depicted in Figure 4; it contains two marked witness graphs M'_0 and M'_1 connected by a transition t_3.

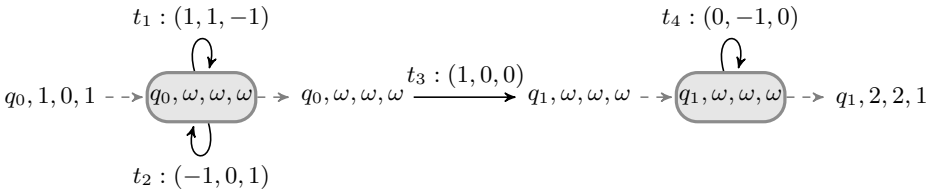

Fig. 4: The next marked witness graph sequence $\xi_1 = M'_0, t_3, M'_1$.

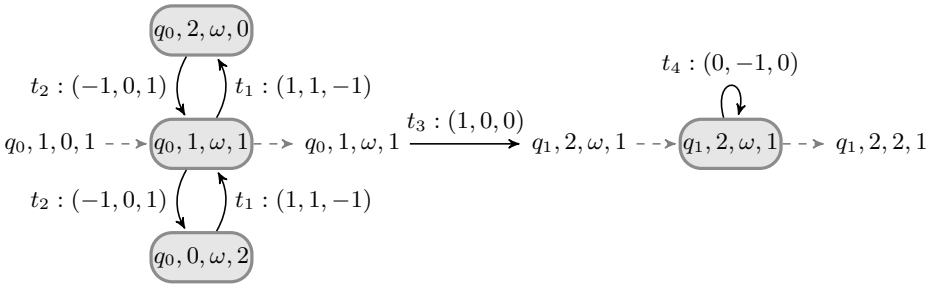

Fig. 5: The final marked witness graph sequence $\xi_2 = M''_0, t_3, M''_1$.

3.1.4. Pumpability. A marked witness graph M is *forward pumpable* if there exist runs in the VASS following paths of M and starting from the input vertex which, when applied to the input constraint, allow to 'pump' arbitrarily high values in the (necessarily common) components labelled ω in the vertices of the graph.

On the one hand, M'_0 in Figure 4 is not forward pumpable: any run of the VASS of Figure 1 starting from $(q_0, 1, 0, 1)$ and using only t_1 and t_2 can indeed reach arbitrarily high values on the second component, but the first and third components are bounded.

On the other hand, M'_1 is forward pumpable, but not *backward pumpable*: starting from $(q_1, 2, 2, 1)$ and applying t_4 in reverse allows to reach arbitrarily high values on the second component, but the first and third components are again bounded.

Again, the decomposition algorithm will observe that the current marked witness graph sequence over-approximates the possible behaviours of the VASS, and refine M'_0 and M'_1 using their bounded components; the values of these bounds can be computed in practice using the coverability tree construction of Karp and Miller [1969]. Propagating the flow constraints, we obtain the final marked witness sequence depicted in Figure 5. This sequence is perfect, and captures in some sense[1] all the runs from $(q_0, 1, 0, 1)$ to $(q_1, 2, 2, 1)$ in the VASS of Example 2.2.

3.1.5. Termination. The termination of the KLMST decomposition algorithm relies on a *ranking function* r mapping marked witness graph sequences to elements of a well-order, and ensuring $r(\xi) > r(\xi')$ whenever ξ' belongs to $\mathrm{dec}(\xi)$ [Kosaraju 1982]. More precisely, the ranking function r associates to ξ a multiset of triples of natural numbers, one triple for each marked witness graph in the sequence. These triples consist of (1) $|I|$, the number of ω-components of the marked witness graph, (2) $|E|$, the number of transitions of the marked witness graph, and (3) $|I^{\mathrm{in}}| + |I^{\mathrm{out}}|$, the number of ω-components in the input and output constraints. This results for the sequences ξ_0, ξ_1,

[1] It represents exactly the *downward closure* of the set of runs from $(q_0, 1, 0, 1)$ to $(q_1, 2, 2, 1)$; see the Decomposition Theorem of Leroux and Schmitz [2015], which might also help the reader build an intuition about marked witness graph sequences and the KLMST decomposition algorithm.

and ξ_2 of our example in the multisets

$$r(\xi_0) = \{(3,5,0)\}\ , \qquad r(\xi_1) = \{(3,2,3),(3,1,3)\}\ , \qquad r(\xi_2) = \{(1,4,1),(1,1,1)\}\ . \tag{6}$$

Let us consider the lexicographic ordering over $\mathbb{N}^3$; finite multisets of triples in $\mathbb{N}^3$ are then well-ordered using the ordering of Dershowitz and Manna [1979].

Observe that we can see the KLMST algorithm as building in general a forest of marked witness graph sequences, with the elements of Ξ_0 as its finitely many roots, and where each imperfect marked witness graph sequence ξ is the parent of the sequences in $\mathrm{dec}(\xi)$. The ranking function r then shows that the trees in this forest are of finite height; since $\mathrm{dec}(\xi)$ is finite for all ξ, they are also of finite branching degree, hence the trees are finite by Kőnig's Lemma and the algorithm terminates.

3.2. Fast-Growing Upper Bounds

Hopefully, the reader has now some vague intuition about the KLMST decomposition algorithm. The key point for complexity considerations is the termination argument by a ranking function explained in §3.1.5: we know that any sequence $\xi_0, \xi_1, \xi_2, \dots$ of marked witness graph sequences with $\xi_{n+1} \in \mathrm{dec}(\xi_n)$ is finite since

$$r(\xi_0) > r(\xi_1) > r(\xi_2) > \cdots \tag{7}$$

is a decreasing sequence in the well-order of multisets of triples of naturals. In order to bound the complexity of the KLMST decomposition algorithm, we are going to bound the length L of such sequences. We shall relate this length with the order type of the ranking function.

3.2.1. Order Types. Recall that an ordinal $\alpha < \varepsilon_0$ (these are rather small, computable, countable ordinals) can be written uniquely in Cantor normal form (CNF) as an *ordinal term*

$$\alpha = \omega^{\alpha_1} + \cdots + \omega^{\alpha_n} \tag{8}$$

where the ω-exponents $\alpha > \alpha_1 \geq \cdots \geq \alpha_n$ are written themselves in CNF; the case where $n = 0$ then corresponds to the ordinal 0. Equivalently, when gathering summands with the same ω-exponent α_i, this can be written as

$$\alpha = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_n} \cdot c_n \tag{9}$$

with $\alpha > \alpha_1 > \cdots > \alpha_n$ and finite *coefficients* $0 < c_i < \omega$.

One can compare two ordinals $\alpha = \omega^{\alpha_1} + \cdots + \omega^{\alpha_n}$ and $\beta = \omega^{\beta_1} + \cdots + \omega^{\beta_m}$ *syntactically* based on their CNFs (8): $\alpha < \beta$ if and only if there exists $k \leq m$ such that $\alpha_j = \beta_j$ for all $1 \leq j < k$ with $j \leq n$, and $n < k$ or $\alpha_k < \beta_k$.

The order type of multisets of triples of natural numbers is ω^{ω^3} [Dershowitz and Manna 1979], and we can equivalently see the ranking function r of §3.1.5 as ranging over ordinals below ω^{ω^3}: the ranks in (7) then become

$$r(\xi_0) = \omega^{\omega^2 \cdot 3 + \omega \cdot 5}, \quad r(\xi_1) = \omega^{\omega^2 \cdot 3 + \omega \cdot 2 + 3} + \omega^{\omega^2 \cdot 3 + \omega + 3}, \quad r(\xi_2) = \omega^{\omega^2 + \omega \cdot 4 + 1} + \omega^{\omega^2 + \omega + 1}. \tag{10}$$

3.2.2. Controlled Sequences and Length Function Theorems. Although sequences like (7) are always finite, they can be of arbitrary length in general. For instance, the sequences

$$\omega > n > n-1 > n-2 > \cdots > 0 \tag{11}$$

are decreasing for all n, and have length $n + 2$.

Thankfully, the sequences of ranks built by the KLMST decomposition algorithm are not arbitrary. The ordinal terms appearing in (7) during the course of the KLMST decomposition algorithm are indeed *controlled*, meaning that 'jumps' to arbitrary high

coefficients as in (11) cannot occur. More formally, let us define a *norm* on ordinals below ε_0 as the maximal coefficient appearing in their CNF (9): given $\alpha = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha}_{n} \cdot c_n$ with $\alpha > \alpha_1 > \cdots > \alpha_n$, $N\alpha \stackrel{\text{def}}{=} \max_{1 \leq j \leq n}(c_j, N\alpha_j)$. For instance, $Nr(\xi_0) = 5$, $Nr(\xi_1) = 3$, and $Nr(\xi_2) = 4$ in (10). Let now $g \colon \mathbb{N} \to \mathbb{N}$ be a strictly increasing function and n be a natural in $\mathbb{N}$. A sequence $\alpha_0, \alpha_1, \ldots$ of ordinals below ε_0 is (g, n)-*controlled* if $N\alpha_j \leq g^j(n)$ the jth iterate of g. This means in particular that $N\alpha_0 \leq n$, while g bounds the amortised growth of the norm at each step.

The interest of (g, n)-controlled descending sequences of ordinals below some α is that they have a bounded length, which we shall see as a function $g_\alpha(n)$ of the initial norm n. When $n \geq N\alpha$, the *length function theorem* in [Schmitz 2014] shows that g_α is exactly the αth function in the *Cichoń hierarchy* [Cichoń and Tahhan Bittar 1998]—a consequence of general results on ordinal-recursive functions [e.g. Buchholz, Cichoń, and Weiermann 1994]. This provides a transfinite inductive definition of the function g_α:

$$g_0(n) = 0 \,, \qquad g_{\alpha+1}(n) = 1 + g_\alpha(g(n)) \,, \qquad g_\lambda(n) = g_{\lambda(n)}(n) \,, \tag{12}$$

where, for a limit ordinal λ, $\lambda(n)$ is the nth element of its fundamental sequence, also defined by transfinite induction:

$$(\gamma + \omega^{\beta+1})(n) \stackrel{\text{def}}{=} \gamma + \omega^\beta \cdot (n+1) \,, \qquad (\gamma + \omega^{\lambda'})(n) \stackrel{\text{def}}{=} \gamma + \omega^{\lambda(n)} \,. \tag{13}$$

For instance, $\omega(n) = n + 1$ and $\omega^{\omega^3}(n) = \omega^{\omega^3(n)} = \omega^{\omega^2 \cdot (n+1)}$; $g_k(n) = k$ for all $k, n \in \mathbb{N}$ and g, $g_\omega(n) = g_{n+1}(n) = n + 1$, $g_{\omega+1}(n) = 1 + g_\omega(g(n)) = 2 + g(n)$. With higher ordinal indices, these functions grow very fast: using $H(n) \stackrel{\text{def}}{=} n + 1$ as control function, $H_{\omega^2}(n) = H_{\omega \cdot (n+1)}(n) = (2^{n+1} - 1)(n + 1)$, H_{ω^3} is a non-elementary function akin to a tower of exponentials of height n, and H_{ω^ω} is a non-primitive-recursive function akin to the Ackermann function.

Going back to our main purpose, if we provide an initial norm $n \geq 3$ and a control function g for sequences like (7), we will obtain an

$$L = g_{\omega^{\omega^3}}(n) \tag{14}$$

bound on their lengths. Regarding the initial norm, the maximum of the sizes of the initial sequences in Ξ_0 will do, and is bounded by the size of the reachability instance. For the control function, it suffices to bound the size of the marked witness graph sequences in $\mathrm{dec}(\xi)$ compared to that of ξ; in the case of the flow conditions of §3.1.3, the blow-up is at most exponential, but in the case of the pumping conditions of §3.1.4, the blow-up is Ackermannian—i.e. in $\mathscr{F}_\omega$ in the *extended Grzegorczyk hierarchy* of Löb and Wainer [1970]—due to the use of coverability trees [Figueira et al. 2011]. Overall, an Ackermannian control function g fits.

The bound in (14) also provides a bound $g^L(n)$, i.e. of L iterations of the function g, on the size of the marked witness graph sequences constructed by the KLMST decomposition algorithm. Defining $g^\alpha(n) \stackrel{\text{def}}{=} g^{g_\alpha(n)}(n)$, we find a related hierarchy of functions called the *Hardy hierarchy*, which allows by Savitch's Theorem to bound the space required by the KLMST decomposition algorithm by a polynomial in

$$g^L(n) = g^{\omega^{\omega^3}}(n) \,. \tag{15}$$

3.2.3. Complexity Classes. The announced F_{ω^3} upper bound on the complexity of the reachability problem is then just a matter of finding a suitable complexity class. We shall use the *fast-growing complexity classes* from [Schmitz 2016], defined for $\alpha \geq 3$ by

$$\mathsf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{f \in \mathscr{F}_{<\alpha}} \mathsf{DTIME}(H^{\omega^\alpha}(f(n))) \tag{16}$$

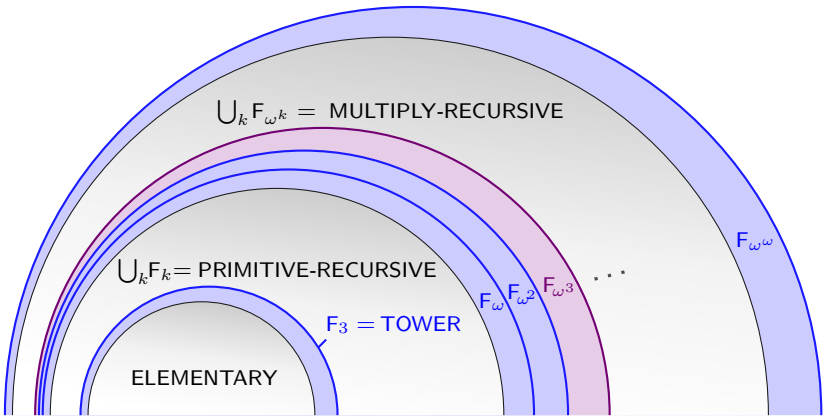

Fig. 6: Pinpointing F_{ω^3} among the complexity classes beyond ELEMENTARY.

where $\mathscr{F}_{<\alpha} = \bigcup_{\beta<\alpha} \mathscr{F}_\beta$ in the extended Grzegorczyk hierarchy. This means that F_α is the class of problems decidable in deterministic time H^{ω^α}—i.e. the ω^αth function in the Hardy hierarchy for $H(n) = n + 1$—of some 'smaller' function $f(n)$ of the size n of the instance. These classes are very robust, and F_{ω^3} captures computations in $g^{\omega^{\omega^3}}(n)$ space for an Ackermannian function g in $\mathscr{F}_\omega$ (see [Schmitz 2016, Section 4] for details). Figure 6 depicts the $(\mathsf{F}_\alpha)_\alpha$ classes and pinpoints F_{ω^3}'s position among them.

3.3. Discussion

3.3.1. Lower Bounds. Facing such a huge upper bound, it is natural to ask how it tight it might be. The best complexity lower bound currently known for the reachability problem is the following result of Lipton [1976] (see also the presentation given by Esparza [1998]):

THEOREM 3.2 (LOWER BOUND THEOREM). *Reachability in VASS is* EXPSPACE*-hard.*

This leaves an enormous gap between EXPSPACE and F_{ω^3}. Nevertheless, the upper bound is obtained with a specific algorithm, the KLMST decomposition algorithm, which—due to its use of coverability trees—is known to require at least an Ackermannian time in the worst case [Müller 1985], i.e. there is an F_ω lower bound for that particular algorithm. Hence, besides the main open question about the exact complexity of the reachability problem, there is a possibly easier open question about the complexity of the KLMST decomposition algorithm, with a smaller complexity gap between F_ω and F_{ω^3}.

3.3.2. Alternative Algorithms. The F_ω lower bound on the KLMST decomposition algorithm means that it might be worth looking for more efficient algorithms. Unfortunately, the other algorithm using Presburger inductive invariants by Leroux [2010, 2011] does not fare better: the 2010 proof using the KLMST decomposition is likely to yield essentially the same F_{ω^3} upper bound, while the alternative 2011 proof using almost semilinear sets does not easily lend itself to a complexity analysis.

4. REACHABILITY IN DIMENSION 2

While the exact complexity of the reachability problem is a long-lasting open problem, there are tight complexity bounds for several restricted variants. A natural way of

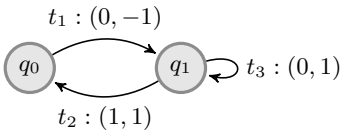

Fig. 7: A 2-dimensional VASS, from [Blondin et al. 2015].

restricting the reachability problem is to fix the dimension. For instance, in dimension one, VASS reachability is NP-complete [Haase et al. 2009] when integers are encoded in binary (and NL-complete when they are encoded in unary [Demri and Gascon 2009])—this is a subcase of reachability in one-counter automata.

In the 2-dimensional case, Hopcroft and Pansiot [1979] were the first to show decidability by showing that the reachability set from a given initial configuration c was effectively *semilinear*, i.e. that one could compute for each state q a representation of the set of vectors $\boldsymbol{u}$ such that $c \to_{\mathcal{V}}^* (q, \boldsymbol{u})$ as a finite union of *linear* sets

$$L(\boldsymbol{b}, \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_n\}) \stackrel{\text{def}}{=} \{\boldsymbol{b} + \lambda_1 \boldsymbol{p}_1 + \cdots + \lambda_n \boldsymbol{p}_n \mid \lambda_1, \dots, \lambda_n \in \mathbb{N}\} \tag{17}$$

defined by a *base* $\boldsymbol{b}$ in $\mathbb{Z}^d$ and a finite set of periods $\boldsymbol{p}_j$ in $\mathbb{Z}^d$—equivalently, these sets are definable in Presburger arithmetic $\mathrm{FO}(\mathbb{Z}, +, \leq)$. Hopcroft and Pansiot [1979, Lemma 2.8] also showed that there exists a 3-dimensional VASS with a non-semilinear reachability set, so that this result does not generalise to higher dimensions. Howell, Rosier, Huynh, and Yen [1986] then showed that this construction was in 2-NEXP and improved it to obtain a 2-EXP algorithm. This left a gap with the NP-hardness proven by Rosier and Yen the same year, which was only closed in 2015 by Blondin, Finkel, Göller, Haase, and McKenzie:

THEOREM 4.1 (2-DIMENSIONAL COMPLEXITY). *Reachability in 2-dimensional VASS is* PSPACE*-complete.*

4.1. Flattable VASS

The crux of the proof of Blondin et al. is a careful analysis and refinement of a result by Leroux and Sutre [2004]: 2-dimensional VASS are *flattable*. In general, a VASS is flattable if its *reachability relation*, which is for each pair of states q, q' the set $\{(\boldsymbol{u}, \boldsymbol{u}') \in \mathbb{N}^{2d} \mid (q, \boldsymbol{u}) \to_{\mathcal{V}}^* (q', \boldsymbol{u}')\}$, can be obtained in full when only following transition sequences taken from the language of a finite set S of regular expressions, each of the form

$$v_0 w_1^* v_1 \cdots w_k^* v_k \tag{18}$$

for some finite sequences $v_0, w_1, v_1, \dots, w_k, v_k$ of transitions in T. Such regular expressions are called *semilinear path schemes* (one will typically require the full sequence $v_0 w_1 v_1 \cdots w_k v_k$ to be a path in the VASS, and each w_j to be a cycle). Let us call $|v_0 w_1 v_1 \cdots w_k v_k|$ its *length* and k its *width*. A set S of semilinear path schemes is *complete* if, for all q, q' in Q and $\boldsymbol{u}, \boldsymbol{u}'$ in $\mathbb{N}^2$, $(q, \boldsymbol{u}) \to_{\mathcal{V}}^* (q', \boldsymbol{u}')$ if and only if there exist σ in the language of S such that $(q, \boldsymbol{u}) \xrightarrow{\sigma}_{\mathcal{V}} (q', \boldsymbol{u}')$.

By synchronising the VASS with the semilinear path schemes, one obtains a *flat* VASS, i.e. without nested loops when seen as a directed graph.

Example 4.2. Consider the 2-dimensional of Figure 7, taken from the article of Blondin et al. [2015]. Although it is has two nested loops t_3 and $t_2 t_1$ in q_1, it is nevertheless flattable. Observe indeed that the occurrences of t_3 can be commuted to occur before those of $t_2 t_1$. This results for the relation between q_0 and q_1 in a complete set $\{t_1 t_3^*, t_1 t_3^* t_2 (t_1 t_2)^* t_1\}$ (the first holding for the case where t_2 is never used).

A flat VASS has necessarily a semilinear reachability relation, and a fortiori a semilinar reachability set: given a semilinear path scheme as in (18), one can compute a linear set with basis the sum of the effects of the transitions in $v_0 v_1 \cdots v_k$, and a period for the effect of each cycle w_j. Perhaps more surprisingly, the converse is true: Leroux [2013a] showed that any VASS with a semilinear reachability set is flattable.

4.2. Complexity Bounds

The main technical result of Blondin et al. [2015] is their Theorem 1:

THEOREM 4.3 (2-DIMENSIONAL SEMILINEAR DECOMPOSITION). *Given a 2-dimensional VASS $\mathcal{V} = \langle Q, 2, T\rangle$, there exists a complete set S of semilinear path schemes, all of length bounded by $(|Q| + \|T\|)^C$ and width bounded by $C' \cdot |Q|^2$ for some constants C, C'.*

They employ then bounds from integer linear programming to show that, if $(q, \boldsymbol{u}) \to_{\mathcal{V}}^* (q', \boldsymbol{u}')$, and thus $(q, \boldsymbol{u}) \xrightarrow{\sigma}_{\mathcal{V}} (q', \boldsymbol{u})$ where $\sigma = v_0 w_1^{e_1} v_1 \cdots w_k^{e_k} v_k$ for some $e_1, \ldots, e_k$ in $\mathbb{N}$ and some semilinear path scheme $v_0 w_1^* v_1 \cdots w_k^* v_k$ from the complete set S provided by Theorem 4.3, then all the e_j can be bounded by a value exponential in $|Q|$ but polynomial in $\|T\|$, $\|\boldsymbol{u}\|$, and $\|\boldsymbol{u}'\|$. This run can thus be guessed nondeterministically in polynomial space, yielding the upper bound in Theorem 4.1.

The matching PSPACE lower bound is a consequence of the same bound [Fearnley and Jurdziński 2015] for reachability in 1-dimensional VASS, when one additionally requires the values to remain bounded by some B (given as input, in binary).

As a final note, this approach also yields an NP upper bound when integers are encoded in unary instead of binary. However, the best known lower bound in this case is NL-hardness, leaving a complexity gap [Blondin et al. 2015].

4.3. Discussion

Although there is no hope of seeing the approach through linear paths schemes be immediately generalised to arbitrary dimensions—because starting with dimension three, VASS reachability sets are no longer semilinear [Hopcroft and Pansiot 1979, Lemma 2.8]—there is nevertheless a promising open question: the result of Blondin et al. [2015] could be read as providing complexity bounds for flat 2-dimensional VASS: could it be generalised to flat VASS of arbitrary dimension?

Finally, the restriction of the reachability problem to 2-dimensional systems is far from being the only interesting one. The best known variant of the reachability problem is arguably the *coverability problem*, where one asks instead for the existence of a configuration c'' such that $c \to_{\mathcal{V}} c''$ and $c'' \geq c'$ for the product ordering over configurations.[2] Coverability in VASS is EXPSPACE-hard using the argument by Lipton [1976], but importantly, it is in EXPSPACE as shown by Rackoff [1978]. The coverability problem is sufficient in many cases, and unlike the reachability problem, there are several implementations, with increasing success at solving it on large practical instances [e.g. Kaiser et al. 2014; Esparza et al. 2014; Blondin et al. 2016].

The same tight EXPSPACE upper bound applies to the *boundedness problem*, which asks given a VASS $\mathcal{V}$ and a configuration c whether the set of configurations reachable from c is finite [Rackoff 1978]. Many more decision problems on VASS have been shown EXPSPACE-complete based on the techniques of Rackoff [e.g. Rosier and Yen 1986; Atig and Habermehl 2011; Blockelet and Schmitz 2011; Demri 2013; Leroux et al. 2013]. Notably, the *reversible reachability problem* is also EXPSPACE-complete [Leroux 2013b]:

[2]Over configurations, $(q, \boldsymbol{u}) \leq (q', \boldsymbol{u}')$ if and only if $q = q'$ and $\boldsymbol{u}(i) = \boldsymbol{u}'(i)$ for all $1 \leq i \leq d$.

this variant of reachability asks whether both $c \to_{\mathcal{V}}^* c'$ and $c' \to_{\mathcal{V}}^* c$, and the proof for the upper bound combines insights from both [Rackoff 1978] and the KLMST decomposition algorithm.

5. A FEW EQUIVALENT PROBLEMS

The centrality of the reachability problem was recognised early on; Hack [1975a] in particular identified its recursive equivalence with the liveness, single-place reachability, persistence, and language emptiness problems for Petri nets. What is remarkable however is the regularity with which decision problems—in seemingly unrelated areas—turn out to be interreducible with the reachability problem. In fact, given its importance in many fields, it would be no exaggeration to define a complexity class REACHABILITY for the class of problems reducible[3] to VASS reachability. Here we will see only a small sample of the problems interreducible with reachability.

5.1. Formal Languages

The transitions of a VASS can be labelled with symbols from $\Sigma \cup \{\varepsilon\}$, where Σ is a finite alphabet and ε denotes the empty string. Formally, we let in this case $T \subseteq Q \times \mathbb{Z}^d \times (\Sigma \cup \{\varepsilon\}) \times Q$, and denote by π_Σ the projection $T^* \to \Sigma^*$ defined by $\pi_\Sigma(q, \boldsymbol{a}, b, q') \stackrel{\text{def}}{=} b$. This allows to define the (reachability) *language* of a labelled VASS $\mathcal{V}$ between an initial configuration c and a final configuration c' as the union over all runs from c to c' in $\mathcal{S}_\mathcal{V}$ of the concatenations of labels:

$$L_\mathcal{V}(c, c') \stackrel{\text{def}}{=} \{\pi_\Sigma(\sigma) \mid c \xrightarrow{\sigma}_\mathcal{V} c'\} \,. \tag{19}$$

The *non-emptiness problem* for such languages is thus equivalent to the reachability problem. Labelled VASS are also known as *partially blind multicounter automata* [Greibach 1978] and can also be recognised by suitable *valence automata* over polycyclic monoids [Render and Kambites 2009, Proposition 5.1]. The study of this class of languages was already under way for Petri nets before 1978; see e.g. [Hack 1975b].

The equivalence between the reachability problem and language emptiness is rather natural one, but there are some less obvious connections, for instance:

The Szilard language $S(\mathcal{G})$ of a context-free grammar $\mathcal{G}$ is the set of sequences of names of the productions used in grammar derivations. Crespi-Reghizzi and Mandrioli [1977] show that the problem, given a context-free grammar $\mathcal{G}$ and a regular language R, of whether $S(\mathcal{G}) \cap R$ is non-empty, is recursively equivalent to VASS reachability.

The shuffle closure $\mathit{Shuffle}(L)$ of a language L is obtained by *shuffling*, i.e. interspersing, an arbitrary number of words from L. Gischer [1981] showed that the shuffle closure $\mathit{Shuffle}(R)$ of any regular language is always a VASS language. Bojańczyk, David, Muscholl, Schwentick, and Segoufin [2011] strengthened this result and showed that the non-emptiness problem for *data automata*, which is equivalent to checking the non-emptiness of $\mathit{Shuffle}(R_1) \cap R_2$ for R_1 and R_2 two regular languages, is interreducible with VASS reachability.

5.2. Logic

There is a natural connection between logic and VASS reachability through *model-checking*: the model-checking problem for a logic allowing to express reachability in the infinite transition system $\mathcal{S}_\mathcal{V}$ will be at least as hard as reachability. It turns out

[3]Due to the unknown exact complexity of the reachability problem, the class of reductions we could afford when defining this class is unclear; some of the examples here use many-one EXPSPACE reductions.

that many temporal logics able to express reachability have an undecidable model-checking problem on VASS—with a few notable exceptions [Howell et al. 1991; Jančar 1990].

Here are some more surprising examples of connections between VASS and logic:

Data Logics. A *data word* is a sequence of pairs (b, d) where b is a label taken from a finite alphabet and d is a datum from an infinite countable data domain. Data logics allow to reason on such words but can only compare data for equality and disequality. The satisfiability problem for several different such logics on data words is interreducible with VASS reachability [e.g. Bojańczyk et al. 2011; Kara et al. 2010; Demri et al. 2013; Decker et al. 2014; Colcombet and Manuel 2014].

Linear Logic. VASS are convenient to describe the usage of discrete resources, and they have been used as models of linear logic fragments [e.g. Engberg and Winskel 1997]. More to the point of this column, VASS reachability is equivalent to validity in the !-*Horn* fragment of linear logic [Kanovich 1995].

Note that the cases of data logics on *data trees* [Bojańczyk et al. 2009; Dimino et al. 2015] and of larger fragments of propositional linear logic [Lazić and Schmitz 2015] can be approached through branching extensions of VASS; see §6.2.2.

5.3. Concurrent Systems

Vector addition systems and Petri nets are especially suited for the modelling of finite-state processes running concurrently, and the reachability problem naturally pops up when trying to verify their correctness. What is perhaps less obvious is that this idea can be applied beyond the verification of safety properties on finite-state processes.

For instance, German and Sistla [1992] show that concurrent systems consisting of a main control process along with an arbitrary number of user processes can be checked against specifications written in linear temporal logic—including *liveness* conditions—by a reduction to the reachability problem. As another example, Ganty and Majumdar [2012] prove that liveness of a class of *recursive* asynchronous programs is equivalent to VASS reachability.

5.4. Process Calculi

There is a rich literature on the use of Petri net executions as event structures to provide process semantics [e.g. Nielsen et al. 1981; Degano et al. 1989]. Petri nets themselves fit in the hierarchies of process calculi among the rather low-level ones [Mayr 2000].

But even very expressive calculi tend to have practically useful fragments that can be reduced to VASS. For instance, Meyer [2009] describes a fragment of the π-calculus with restricted usage of names, which can be translated into Petri nets. Larger fragments can also be tackled through VASS extensions with data [Rosa-Velardo and Martos-Salgado 2012], see Section 6.3 for related models.

6. A FEW EXTENSIONS

The decidability of the reachability problem for VASS is an intricate result, and undecidability is never very far. This section is an opportunity to see how far the Decidability Theorem can be pushed, but also to advertise for a few open problems.

6.1. Zero Tests

A *zero test* is a special type of transition $t = (q, \mathrm{zero}_i, q')$ where i ranges over $\{1, \ldots, d\}$, allowing a step $(q, \boldsymbol{u}) \xrightarrow{t} (q', \boldsymbol{u})$ if $\boldsymbol{u}(i) = 0$. Allowing unrestricted zero tests yields

a Minsky machine, with an undecidable reachability problem already in dimension two—even coverability is undecidable on such systems.

However, reachability in VASS extended with the ability to test a *single* component for zero—for instance always the first component—remains decidable. It still remains decidable if several components can be tested with a *hierarchical* policy: component i_1 can be freely tested for zero, but i_2 can only be tested for zero in configurations where the i_1th component is zero, and i_3 only in configurations where both the i_1th and i_2th components are zero, etc. [Reinhardt 2008; Bonnet 2013]. The complexity of reachability in these models is widely open; the best known lower bound is still Lipton's EXPSPACE-hardness, and no upper bound is known.

6.2. Recursion and Nesting

Motivated by the need to model distributed systems with some recursive behaviour, there is a variety of VASS extensions that include recursion in some manner, and differing on the (sometimes subtle) way in which they allow interactions between recursion and the integer components. For instance, *nested counter systems* [Lomazova and Schnoebelen 2000; Decker et al. 2014] act on finite multisets of finite multisets of ... of finite multisets of states as configurations, allowing to model hierarchical computations, but have an undecidable reachability problem. On the other hand, the *process rewrite systems* of Mayr [2000] perform prefix rewrites on terms of a process algebra, and generalise in a sense both VASS and pushdown systems, but still enjoy a decidable reachability problem—with unknown complexity.

6.2.1. Pushdown VASS. One natural way to extend VASS to handle recursion is to add new push and pop operations acting on a pushdown stack with a finite stack alphabet. Note that this generalises VASS with a single zero test, since the particular component tested for zero could be implemented as a stack with a distinguished bottom-of-stack symbol. The decidability of reachability is currently open, but here at least we have better lower bounds: Lazić [2013] showed indeed the problem to be TOWER-hard.

6.2.2. Alternating Branching VASS. A different way of adding a pushdown stack to a VASS is to let it store vectors from $\mathbb{N}^d$ on its stack. The system can then add a vector $\boldsymbol{a}$ from $\mathbb{Z}^d$ to the vector $\boldsymbol{u}$ currently on top of the stack. The key question is which semantics to employ when popping $\boldsymbol{u}$ and pushing multiple vectors, say $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, to the top of the stack. For instance, if we allow to duplicate the vector $\boldsymbol{u}$ so that $\boldsymbol{u} = \boldsymbol{u}_1 = \boldsymbol{u}_2$, then we obtain the model of *alternating VASS* [e.g. Courtois and Schmitz 2014], which have an undecidable reachability problem.

A very interesting case occurs when we split $\boldsymbol{u}$ nondeterministically into $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ such that $\boldsymbol{u} = \boldsymbol{u}_1 + \boldsymbol{u}_2$. This model of *branching VASS* was introduced by Rambow [1994] in computational linguistics, and independently rediscovered on several occasions since [see the survey in Schmitz 2010]. The decidability of the reachability problem for branching VASS is a major open problem[4] already mentioned in this column [Bojańczyk 2014]—it is in particular equivalent to provability in MELL, the multiplicative exponential fragment of linear logic [de Groote et al. 2004]. As with pushdown VASS, although we do not know whether reachability is decidable,[4] we have again a TOWER lower bound [Lazić and Schmitz 2015].

6.3. Data

The model of *data nets* of Lazić, Newcomb, Ouaknine, Roscoe, and Worrell [2008] extends Petri nets with the ability to manipulate data from some infinite domain $\mathbb{D}$. Dif-

[4]There is a recent claim that the problem is decidable [Bimbó 2015], but I have been unable to verify its proof.

ferent variants exist, all with an undecidable reachability problem, except for one case: *unordered data Petri nets*, where the system can only manipulate data as pure names through equality and disequality constraints. In more concrete terms, the configurations of such a system no longer carry a single vector from $\mathbb{N}^d$, but a finite multiset of them, padded with infinitely many 0's. Transitions nondeterministically select some (bounded) number of such vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$, and apply some translations $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$ from $\mathbb{Z}^d$ to each one. The decidability of the reachability problem for this model is open, with a TOWER lower bound proven by Lazić et al. [2008]

ACKNOWLEDGMENTS

I thank Mikołaj Bojańczyk, Christoph Haase, and Ranko Lazić for their comments on a draft of this column. I am of course the one to blame for the remaining mistakes and inaccuracies.

REFERENCES

Mohamed Faouzi Atig and Peter Habermehl. 2011. On Yen's path logic for Petri nets. *Int. J. Fund. Comput. Sci.* 22, 4 (2011), 783–799. DOI:10.1142/S0129054111008428

Katalin Bimbó. 2015. The decidability of the intensional fragment of classical linear logic. *Theor. Comput. Sci.* 597 (2015), 1–17. DOI:10.1016/j.tcs.2015.06.019

Michel Blockelet and Sylvain Schmitz. 2011. Model-checking coverability graphs of vector addition systems. In *Proc. MFCS 2011 (Lect. Notes in Comput. Sci.)*, Vol. 6907. Springer, 108–119. DOI:10.1007/978-3-642-22993-0_13

Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. 2015. Reachability in two-dimensional vector addition systems with states is PSPACE-complete. In *Proc. LICS 2015*. IEEE Press, 32–43. DOI:10.1109/LICS.2015.14

Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. 2016. Approaching the coverability problem continuously. In *Proc. TACAS 2016 (Lect. Notes in Comput. Sci.)*. http://arxiv.org/abs/1510.05724 To appear.

Mikołaj Bojańczyk. 2014. Some open problems in automata and logic. *ACM SIGLOG News* 1, 2 (2014), 3–12. DOI:10.1145/2677161.2677163

Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. 2011. Two-variable logic on data words. *ACM Trans. Comput. Logic* 12, 4:27 (2011), 1–26. DOI:10.1145/1970398.1970403

Mikołaj Bojańczyk, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. 2009. Two-variable logic on data trees and XML reasoning. *J. ACM* 56, 3 (2009), 1–48. DOI:10.1145/1516512.1516515

Rémi Bonnet. 2013. *Theory of Well-Structured Transition Systems and Extended Vector-Addition Systems*. Thèse de doctorat. ENS Cachan. http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/bonnet-phd13.pdf

Wilfried Buchholz, E. Adam Cichoń, and Andreas Weiermann. 1994. A uniform approach to fundamental sequences and hierarchies. *Math. Logic Quart.* 40, 2 (1994), 273–286. DOI:10.1002/malq.19940400212

E. Adam Cichoń and Elias Tahhan Bittar. 1998. Ordinal recursive bounds for Higman's Theorem. *Theor. Comput. Sci.* 201, 1–2 (1998), 63–84. DOI:10.1016/S0304-3975(97)00009-1

Thomas Colcombet and Amaldev Manuel. 2014. Generalized data automata and fixpoint logic. In *Proc. FSTTCS 2014 (Leibniz Int. Proc. Inf.)*, Vol. 29. LZI, 267–278. DOI:10.4230/LIPIcs.FSTTCS.2014.267

Jean-Baptiste Courtois and Sylvain Schmitz. 2014. Alternating vector addition systems with states. In *Proc. MFCS 2014 (Lect. Notes in Comput. Sci.)*, Vol. 8634. Springer, 220–231. DOI:10.1007/978-3-662-44522-8_19

Stefano Crespi-Reghizzi and Dino Mandrioli. 1977. Petri nets and Szilard languages. *Inform. and Cont.* 33, 2 (1977), 177–192. DOI:10.1016/S0019-9958(77)90558-7

Philippe de Groote, Bruno Guillaume, and Sylvain Salvati. 2004. Vector addition tree automata. In *Proc. LICS 2004*. IEEE Press, 64–73. DOI:10.1109/LICS.2004.51

Normann Decker, Peter Habermehl, Martin Leucker, and Daniel Thoma. 2014. Ordered navigation on multi-attributed data words. In *Proc. Concur 2014 (Lect. Notes in Comput. Sci.)*, Vol. 8704. Springer, 497–511. DOI:10.1007/978-3-662-44584-6_34

Pierpaolo Degano, José Meseguer, and Ugo Montanari. 1989. Axiomatizing net computations and processes. In *Proc. LICS'89*. 175–185. DOI:10.1109/LICS.1989.39172

Stéphane Demri. 2013. On selective unboundedness of VASS. *J. Comput. Syst. Sci.* 79, 5 (2013), 689–713. DOI:10.1016/j.jcss.2013.01.014

Stéphane Demri, Diego Figueira, and M. Praveen. 2013. Reasoning about data repetitions with counter systems. In *Proc. LICS 2013*. IEEE Press, 33–42. DOI:10.1109/LICS.2013.8

Stéphane Demri and Régis Gascon. 2009. The effects of bounding syntactic resources on Presburger LTL. *J. Logic Comput.* 19, 6 (2009), 1541–1575. DOI:10.1093/logcom/exp037

Nachum Dershowitz and Zohar Manna. 1979. Proving termination with multiset orderings. *Commun. ACM* 22, 8 (1979), 465–476. DOI:10.1145/359138.359142

Jérémie Dimino, Florent Jacquemard, and Luc Segoufin. 2015. $\mathrm{FO}^2(<, +1, \sim)$ on data trees, data tree automata and branching vector addition systems. (2015). http://hal.inria.fr/hal-00769249 Preprint.

Uffe Engberg and Glynn Winskel. 1997. Completeness results for linear logic on Petri nets. *Ann. Pure Appl. Log.* 86, 2 (1997), 101–135. DOI:10.1016/S0168-0072(96)00024-3

Javier Esparza. 1998. Decidability and complexity of Petri net problems — an introduction. In *Lectures on Petri Nets I: Basic Models (Lect. Notes in Comput. Sci.)*, Vol. 1491. Springer, 374–428. DOI:10.1007/3-540-65306-6_20

Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp Meyer, and Filip Niksic. 2014. An SMT-based approach to coverability analysis. In *Proc. CAV 2014 (Lect. Notes in Comput. Sci.)*, Vol. 8559. Springer, 603–619. DOI:10.1007/978-3-319-08867-9_40

Javier Esparza and Mogens Nielsen. 1994. Decidability issues for Petri nets — a survey. *Bull. EATCS* 52 (1994), 244–262.

John Fearnley and Marcin Jurdziński. 2015. Reachability in two-clock timed automata is PSPACE-complete. *Inform. and Comput.* 243 (2015), 26–36. DOI:10.1016/j.ic.2014.12.004

Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. 2011. Ackermannian and primitive-recursive bounds with Dickson's Lemma. In *Proc. LICS 2011*. IEEE Press, 269–278. DOI:10.1109/LICS.2011.39

Pierre Ganty and Rupak Majumdar. 2012. Algorithmic verification of asynchronous programs. *ACM Trans. Progr. Lang. Syst.* 34, 1:6 (2012), 1–48. DOI:10.1145/2160910.2160915

Steven M. German and A. Prasad Sistla. 1992. Reasoning about systems with many processes. *J. ACM* 39, 3 (1992), 675–735. DOI:10.1145/146637.146681

Jay Gischer. 1981. Shuffle languages, Petri nets, and context-sensitive grammars. *Commun. ACM* 24, 9 (1981), 597–605. DOI:10.1145/358746.358767

Sheila A. Greibach. 1978. Remarks on blind and partially blind one-way multicounter machines. *Theor. Comput. Sci.* 7, 3 (1978), 311–324. DOI:10.1016/0304-3975(78)90020-8

Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. 2009. Reachability in succinct and parametric one-counter automata. In *Proc. Concur 2009 (Lect. Notes in Comput. Sci.)*, Vol. 5710. Springer, 369–383. DOI:10.1007/978-3-642-04081-8_25

Michel H. T. Hack. 1975a. *Decidability questions for Petri nets*. Ph.D. Dissertation. MIT. http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-161.pdf

Michel H. T. Hack. 1975b. *Petri net languages*. Computation Structures Group Memo 124. MIT. http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-159.pdf

John E. Hopcroft and Jean-Jacques Pansiot. 1979. On the reachability problem for 5-dimensional vector addition systems. *Theor. Comput. Sci.* 8 (1979), 135–159. DOI:10.1016/0304-3975(79)90041-0

Rodney R. Howell, Louis E. Rosier, Dung T. Huynh, and Hsu-Chun Yen. 1986. Some complexity bounds for problems concerning finite and 2-dimensional vector addition systems with states. *Theor. Comput. Sci.* 46 (1986), 107–140. DOI:10.1016/0304-3975(86)90026-5

Rodney R. Howell, Louis E. Rosier, and Hsu-Chun Yen. 1991. A taxonomy of fairness and temporal logic problems for Petri nets. *Theor. Comput. Sci.* 82, 2 (1991), 341–372. DOI:10.1016/0304-3975(91)90228-T

Petr Jančar. 1990. Decidability of a temporal logic problem for Petri nets. *Theor. Comput. Sci.* 74, 1 (1990), 71–93. DOI:10.1016/0304-3975(90)90006-4

Alexander Kaiser, Daniel Kroening, and Thomas Wahl. 2014. A widening approach to multithreaded program verification. *ACM Trans. Progr. Lang. Syst.* 36, 4, Article 14 (2014), 29 pages. DOI:10.1145/2629608

Max I. Kanovich. 1995. Petri nets, Horn programs, linear logic and vector games. *Ann. Pure Appl. Log.* 75, 1–2 (1995), 107–135. DOI:10.1016/0168-0072(94)00060-G

Ahmet Kara, Thomas Schwentick, and Thomas Zeume. 2010. Temporal logics on words with multiple data values. In *Proc. FSTTCS 2010 (Leibniz Int. Proc. Inf.)*, Vol. 8. LZI, 481–492. DOI:10.4230/LIPIcs.FSTTCS.2010.481

Richard M. Karp and Raymond E. Miller. 1969. Parallel program schemata. *J. Comput. Syst. Sci.* 3, 2 (1969), 147–195. DOI:10.1016/S0022-0000(69)80011-5

S. Rao Kosaraju. 1982. Decidability of reachability in vector addition systems. In *Proc. STOC'82*. ACM, 267–281. DOI:10.1145/800070.802201

Jean-Luc Lambert. 1992. A structure to decide reachability in Petri nets. *Theor. Comput. Sci.* 99, 1 (1992), 79–104. DOI:10.1016/0304-3975(92)90173-D

Ranko Lazić. 2013. The reachability problem for vector addition systems with a stack is not elementary. Preprint. (2013). http://arxiv.org/abs/1310.1767 Presented at RP 2012.

Ranko Lazić, Tom Newcomb, Joël Ouaknine, A.W. Roscoe, and James Worrell. 2008. Nets with tokens which carry data. *Fund. Inform.* 88, 3 (2008), 251–274.

Ranko Lazić and Sylvain Schmitz. 2015. Non-elementary complexities for branching VASS, MELL, and extensions. *ACM Trans. Comput. Logic* 16, 3, Article 20 (2015), 30 pages. DOI:10.1145/2733375

Jérôme Leroux. 2010. The general vector addition system reachability problem by Presburger inductive

invariants. *Logic. Meth. in Comput. Sci.* 6, 3 (2010), 1–25. DOI:10.2168/LMCS-6(3:22)2010
Jérôme Leroux. 2011. Vector addition system reachability problem: a short self-contained proof. In *Proc. POPL 2011*. ACM, 307–316. DOI:10.1145/1926385.1926421
Jérôme Leroux. 2013a. Presburger vector addition systems. In *Proc. LICS 2013*. IEEE Press, 23–32. DOI:10.1109/LICS.2013.7
Jérôme Leroux. 2013b. Vector Addition System Reversible Reachability Problem. *Logic. Meth. in Comput. Sci.* 9, 1, Article 5 (2013). DOI:10.2168/LMCS-9(1:5)2013
Jérôme Leroux, M. Praveen, and Grégoire Sutre. 2013. A relational trace logic for vector addition systems with application to context-freeness. In *Proc. Concur 2013 (Lect. Notes in Comput. Sci.)*, Vol. 8052. Springer, 137–151. DOI:10.1007/978-3-642-40184-8_11
Jérôme Leroux and Sylvain Schmitz. 2015. Demystifying reachability in vector addition systems. In *Proc. LICS 2015*. IEEE Press, 56–67. DOI:10.1109/LICS.2015.16
Jérôme Leroux and Grégoire Sutre. 2004. On flatness for 2-dimensional vector addition systems with states. In *Proc. Concur 2004 (Lect. Notes in Comput. Sci.)*, Vol. 3170. Springer, 402–416. DOI:10.1007/978-3-540-28644-8_26
Richard J. Lipton. 1976. *The reachability problem requires exponential space*. Technical Report 62. Yale University. http://cpsc.yale.edu/sites/default/files/files/tr63.pdf
Martin H. Löb and Stanley S. Wainer. 1970. Hierarchies of number-theoretic functions. I. *Arch. Math. Log.* 13, 1–2 (1970), 39–51. DOI:10.1007/BF01967649
Irina A. Lomazova and Philippe Schnoebelen. 2000. Some decidability results for nested Petri nets. In *Proc. PSI'99 (Lect. Notes in Comput. Sci.)*, Vol. 1755. Springer, 208–220. DOI:10.1007/3-540-46562-6_18
Ernst W. Mayr. 1981. An algorithm for the general Petri net reachability problem. In *Proc. STOC'81*. ACM, 238–246. DOI:10.1145/800076.802477
Richard Mayr. 2000. Process rewrite systems. *Inform. and Comput.* 156, 1–2 (2000), 264–286. DOI:10.1006/inco.1999.2826
Roland Meyer. 2009. A theory of structural stationarity in the *pi*-calculus. *Acta Inf.* 46, 2 (2009), 87–137. DOI:10.1007/s00236-009-0091-x
Horst Müller. 1985. The reachability problem for VAS. In *Advances in Petri Nets 1984*. Lect. Notes in Comput. Sci., Vol. 188. Springer, 376–391. DOI:10.1007/3-540-15204-0_21
Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. 1981. Petri nets, event structures and domains, part I. *Theor. Comput. Sci.* 13, 1 (1981), 85–108. DOI:10.1016/0304-3975(81)90112-2
Carl A. Petri. 1962. *Kommunikation mit Automaten*. Ph.D. Dissertation. Universität Bonn. http://edoc.sub.uni-hamburg.de/informatik/volltexte/2011/160/
Charles Rackoff. 1978. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.* 6, 2 (1978), 223–231. DOI:10.1016/0304-3975(78)90036-1
Owen Rambow. 1994. Multiset-valued linear index grammars: imposing dominance constraints on derivations. In *Proc. ACL'94*. ACL Press, 263–270. DOI:10.3115/981732.981768
Klaus Reinhardt. 2008. Reachability in Petri nets with inhibitor arcs. In *Proc. RP 2008 (Elec. Notes in Theor. Comput. Sci.)*, Vol. 223. 239–264. DOI:10.1016/j.entcs.2008.12.042
Elaine Render and Mark Kambites. 2009. Rational subsets of polycyclic monoids and valence automata. *Inform. and Comput.* 207, 11 (2009), 1329–1339. DOI:10.1016/j.ic.2009.02.012
Christophe Reutenauer. 1990. *The mathematics of Petri nets*. Masson and Prentice.
Fernando Rosa-Velardo and María Martos-Salgado. 2012. Multiset rewriting for the verification of depth-bounded processes with name binding. *Inform. and Comput.* 215 (2012), 68–87. DOI:10.1016/j.ic.2012.03.004
Louis E. Rosier and Hsu-Chun Yen. 1986. A multiparameter analysis of the boundedness problem for vector addition systems. *J. Comput. Syst. Sci.* 32, 1 (1986), 105–135. DOI:10.1016/0022-0000(86)90006-1
George S. Sacerdote and Richard L. Tenney. 1977. The decidability of the reachability problem for vector addition systems. In *Proc. STOC'77*. ACM, 61–76. DOI:10.1145/800105.803396
Sylvain Schmitz. 2010. On the computational complexity of dominance links in grammatical formalisms. In *Proc. ACL 2010*. ACL Press, 514–524.
Sylvain Schmitz. 2014. Complexity bounds for ordinal-based termination. In *Proc. RP 2014 (Lect. Notes in Comput. Sci.)*, Vol. 8762. Springer, 1–19. DOI:10.1007/978-3-319-11439-2_1
Sylvain Schmitz. 2016. Complexity hierarchies beyond ELEMENTARY. *ACM Trans. Comput. Theory* 8, 1, Article 3 (2016), 36 pages. DOI:10.1145/2858784