Journal of Computer and System Sciences 79 (2013) 23–38

Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

The covering and boundedness problems for branching vector addition systems ☆

Stéphane Demri [a], Marcin Jurdziński [b], Oded Lachish [b,1], Ranko Lazić [b,*]

[a] LSV, ENS de Cachan & CNRS & INRIA, 61, avenue du Président Wilson, 94235 Cachan Cedex, France
[b] DIMAP, Department of Computer Science, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK

ARTICLE INFO

Article history:
Received 5 April 2011
Received in revised form 15 March 2012
Accepted 4 April 2012
Available online 6 April 2012

Keywords:
Petri nets
Branching vector addition systems
Covering
Boundedness
Computational complexity

ABSTRACT

The covering and boundedness problems for branching vector addition systems are shown complete for doubly-exponential time.

1. Introduction

Vector addition systems (shortly, VAS), or equivalently Petri nets (e.g., [1]), are a fundamental model of computation, which is more expressive than finite-state machines and less than Turing-powerful. Decidability and complexity of a variety of problems have been extensively studied ([2] is a comprehensive survey).

A k-dimensional VAS consists of an initial vector of non-negative integers, and a finite set of vectors of integers, all of dimension k. Let us call the initial vector *axiom*, and the other vectors *rules*. A computation can then be thought of as a *derivation*: it starts with the axiom, and at each step, the next vector is derived from the current one by adding a rule. The vectors of interest are the ones derived *admissibly*, i.e. at the end of a derivation which is such that none of the vectors derived during it contains a negative entry.

Covering and boundedness are two central decision problems for VAS. The former asks whether a vector that is pointwise greater than or equal to a given vector can be admissibly derived, and the latter asks whether the set of all admissibly derived vectors is finite. In a landmark article [3], Rackoff showed that covering and boundedness for VAS are in ExpSpace, matching Lipton's lower bound of ExpSpace-hardness [4].[2] Considering the expressively equivalent VAS with states (shortly, VASS), Rosier and Yen refined the proofs of Lipton and Rackoff to obtain almost matching lower and upper bounds in terms of three parameters: the dimension, the binary size of the maximum absolute value of an entry in a rule, and the number of states [5]. Lipton's result was also extended by Mayr and Meyer to reversible Petri nets, which are equivalent to commutative

☆ A preliminary and shorter version of this work was published in the proceedings of FSTTCS 2009.

* Corresponding author. Fax: +44 24 7657 3024.
E-mail addresses: demri@lsv.ens-cachan.fr (S. Demri), mju@dcs.warwick.ac.uk (M. Jurdziński), oded@dcs.bbk.ac.uk (O. Lachish), lazic@dcs.warwick.ac.uk (R. Lazić).
[1] Present address: Computer Science and Information Systems, Birkbeck (University of London), Malet Street, London WC1E 7HX, UK.

[2] We recommend http://rjlipton.wordpress.com/2009/04/08/an-expspace-lower-bound/.

0022-0000/$ – see front matter
http://dx.doi.org/10.1016/j.jcss.2012.04.002

semigroups [6]. Building on Rosier and Yen's work, Habermehl showed that space exponential in the size of the system and polynomial in the size of the formula suffices for model checking the propositional linear-time μ-calculus on VASS, and he obtained a matching lower bound already for LTL on BPP [7]. Further related developments include the identification by Atig and Habermehl of another path logic for Petri nets whose model checking problem is ExpSpace-complete [8], and Demri's proof of ExpSpace-membership of a generalised boundedness problem which subsumes reversal boundedness, place boundedness, regularity and several other interesting problems for VASS [9].

The following is a natural extension of VAS: instead of linearly, computation proceeds from the leaves to the root of a tree. For each node which is not a leaf, its vector is derived by summing the vectors derived at its children and adding a rule vector.[3] The same condition of admissibility applies, i.e. no derived vector may contain a negative entry. This model of computation is branching VAS (shortly, BVAS).

In recent years, it has turned out that BVAS have interesting connections to a number of formalisms:

- BVAS correspond to a class of linear index grammars in computational linguistics [11,12];
- reachability (i.e. admissible derivability) for BVAS is decidable iff provability in multiplicative exponential linear logic is decidable [13];
- Verma and Goubault-Larrecq have extended the computation of Karp and Miller trees [14] to BVAS, and used it to draw conclusions about a class of equational tree automata which are useful for analysing cryptographic protocols [15];
- if first-order logic with 2 variables on finite data trees (which has applications to the XPath query language for XML) is decidable, then so is reachability for BVAS [16].

Covering and boundedness for BVAS are decidable easily using the branching extension of Karp and Miller's procedure [15]. However, the resulting algorithms do not operate in primitive recursive time or space, even in the linear case [17].

The main results we report are that, by switching from VAS to BVAS, covering and boundedness move two notches up the complexity hierarchy, to 2ExpTime-complete.

For the 2ExpTime-memberships, consider the following simple-minded idea for transferring knowledge about VAS derivations to the branching case:

Every simple path from a leaf to the root in a BVAS derivation is a VAS derivation.

We show that the idea can give us mileage, but only after the following new insight, which is needed because the subderivations that grow off the simple path and hence contribute summands to it make the resulting VAS contain rules with unbounded positive entries.

For VAS, we can obtain similar upper bounds to Rackoff's, but which depend only on the dimension and the minimum negative entry in a rule, i.e. not on the maximum positive entry in a rule.

The insight is at the centre of our proofs. In the case of covering, we show it essentially by inspecting carefully a proof of Rackoff, but in the case of boundedness, it relies on proving a new result on small solutions of integer programming problems, which extends a classical theorem of Borosh and Treybig and may also be a contribution of wider interest. To complete the proofs of the 2ExpTime-memberships, we provide arguments for reducing the heights of appropriate BVAS derivations to at most doubly-exponential, and for why resulting small witnesses can be guessed and verified by alternating Turing machines in exponential space.

To obtain 2ExpTime-hardness for covering and boundedness for BVAS, we extend the proof of Lipton to show that computations of alternating machines of size N with counters bounded by 2^{2^N} can be simulated in reverse by BVAS of size $O(N^2)$. Although universal branchings of alternating counter machines copy counter valuations whereas BVAS sum vectors derived at children nodes, the inner workings of Lipton's construction enable us to add a bit of machinery by which the BVAS can simulate the copying. We remark that, as is the case with Lipton's result, the lower bound is shown already for BVAS whose rules contain only entries -1, 0 or 1.

After fixing notations and making some preliminary observations in the next section, that covering and boundedness are in 2ExpTime is shown in Sections 3 and 4, respectively. We then argue in Section 5 that both problems are 2ExpTime-hard.

2. Preliminaries

Numbers, vectors and matrices We write $\mathbb{N}_+$, $\mathbb{N}$ and $\mathbb{Z}$ for the sets of all positive, non-negative and arbitrary integers, respectively. Since we shall only work with integers, let the open interval (a, b) denote $(a, b) \cap \mathbb{Z}$, and analogously for half-open and closed intervals.

Given a dimension $k \in \mathbb{N}$, let $\mathbf{0}$ denote the zero vector and, for each $i \in [1, k]$, $\mathbf{e}_i$ denote the ith unit vector. For $\mathbf{v}, \mathbf{w} \in \mathbb{Z}^k$ and $B \in \mathbb{Z}$, we write:

[3] A different branching extension of VAS was used by Urquhart [10].