Theoretical Computer Science 534 (2014) 53–68

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

The complexity of computing the behaviour of lattice automata on infinite trees

Karsten Lehmann [a,b,*], Rafael Peñaloza [c,d,*]

[a] Optimisation Research Group, NICTA, Australia
[b] Artificial Intelligence Group, Australian National University, Australia
[c] Theoretical Computer Science, TU Dresden, Germany
[d] Center for Advancing Electronics Dresden, Germany

ARTICLE INFO

Keywords:
Weighted automata
Behaviour computation
Lattices
Complexity

ABSTRACT

Several logic-based decision problems have been shown to be reducible to the emptiness problem of automata. In a similar way, non-standard reasoning problems can be reduced to the computation of the behaviour of weighted automata. In this paper, we consider a variant of weighted Büchi automata working on (unlabelled) infinite trees, where the weights belong to a lattice. We analyse the complexity of computing the behaviour of this kind of automata if the underlying lattice is not distributive.

We show that the decision version of this problem is in ExpTime and PSpace-hard in general, assuming that the lattice operations are polynomial-time computable. If the lattice is what we call linear-space-computable-encoded, then the upper bound can be reduced to PSpace, but the lower bound also decreases to NP-hard and co-NP-hard. We conjecture that the upper bounds provided are in fact tight.

1. Introduction

Automata have long been recognized as tools for solving logic-based reasoning tasks. Beyond well-known results on the equivalence of recognizable and MSOL-definable languages [10,20], automata working on infinite inputs have been successfully applied to decide satisfiability of linear temporal logic (LTL) [31,22] formulas, and reason with description logic (DL) [2] knowledge bases, to name just two well-known examples. The main idea in both cases is to construct an automaton $\mathcal{A}$ that accepts all the (well-structured) models of the input, and perform an emptiness test on $\mathcal{A}$. The constructed automaton is a generalized Büchi automaton on infinite words in the case of LTL [38], and a looping automaton (that is, a Büchi automaton where all states are accepting) on infinite trees for DL reasoning [3,7,11,28].[1]

In most of these constructions, it is possible to use a simplified alphabet having only one symbol. Additional alphabet symbols can be encoded within the set of states of the automaton, and in this case the relevant models are described by the *accepting runs* of the automaton, rather than by the recognized language.

Automata-based decision procedures have been generalized to weighted automata over lattices as a means to deal with non-standard reasoning problems, such as axiom-pinpointing [6,24], access control [4,25], or context-based reasoning [5], as well as with non-standard semantics like fuzzy [9,34,35] and possibilistic semantics [30,32]. The idea behind these

* Corresponding authors.
E-mail addresses: karsten.lehmann@nicta.com.au (K. Lehmann), penaloza@tcs.inf.tu-dresden.de (R. Peñaloza).

[1] Other automata models have been considered in the literature, e.g. [12]. For the sake of brevity and clarity, in this paper we focus only on those based on Büchi automata.

http://dx.doi.org/10.1016/j.tcs.2014.02.036
0304-3975/

constructions is that every model can be associated to a "weight" corresponding to the non-standard task. For example, in the axiom-pinpointing scenario, where one is interested in finding the causes of an inconsistency, this weight will be the set of axioms violated by the model.[2] We are then interested in finding the supremum of the weights of all these models, which in the case of axiom-pinpointing will be the set of all sets of axioms that prevent the existence of a model.

Suppose that we can associate every transition of the constructed automaton with a weight in such a way that the infimum of the weights of all transitions appearing in a successful run (that is, the weight of this successful run) corresponds exactly to the weight of the model it represents, as described before. Then, this kind of non-standard reasoning reduces to a computation of the behaviour of the weighted automaton. To fully understand the complexity of non-standard reasoning tasks, we need to study how hard it is to compute the behaviour of lattice automata. Thus, we are interested in the complexity of computing the behaviour of Büchi automata on infinite trees, whose weights belong to a lattice.

For distributive lattices, it is known that the behaviour of generalized Büchi automata can be computed in polynomial time [6,16], matching the complexity of deciding emptiness of (unweighted) Büchi automata [33,37]. This result provides tight upper bounds for the complexity of axiom-pinpointing in expressive DLs, and of reasoning in special kinds of fuzzy and possibilistic DLs and LTL. Unfortunately, distributivity of the lattice is not always a valid assumption. For example, in access control the underlying access lattice is often provided by the security manager, or automatically generated from a compact description of the access rights needed [14], and it can take any shape. In this paper we study the complexity of computing the behaviour in case the lattice is not distributive. We notice that without distributivity, the automata are not any more *weighted automata* in the standard sense, as defined in [26,27,17]. Variants of weighted automata on finite [18] and infinite [19] trees, in which the distributivity assumption is dropped have been recently studied; in fact, the underlying algebra has been generalized to more complex valuation monoids [15]. However, those papers focus mostly on the expressivity of the automata, and their relation with weighted logics. To the best of our knowledge, there has been no systematic study of the complexity of computing the behaviour of automata over non-distributive structures.

We show that the behaviour of automata over arbitrary lattices can be computed by a simple "black-box" mechanism that tests emptiness of exponentially many unweighted Büchi automata. This yields an exponential time upper bound for the computation of the behaviour, assuming that lattice operations can be performed in polynomial time. Unfortunately, the best-case running time of this algorithm is also exponential on the number of different weights appearing in the automation. If the lattice can be represented in such a way that its operations do not increase the space requirements (a condition we call *linear-space-computable-encoded*), then this upper bound can be improved to polynomial space. The exponential upper bound for general lattices is not new; in fact, it is a simple consequence of the results from [19], where it was shown that every recognizable tree language over bi-locally finite strong bimonoids can be expressed as a recognizable step-function; i.e., the behaviour of every automaton over such strong bimonoids can be described as a finite weighted sum of languages accepted by (unweighted) automata. The tighter upper bound for the class of linear-space-computable-encoded lattices, on the other hand, was previously unknown.

Regarding lower bounds, we provide a linear-space-computable-encoded lattice L_{sat} and show that computing the behaviour of automata over this lattice is hard for the classes NP and co-NP. We further improve the lower bound for general lattices by providing a lattice L_{qbf} for which computing the behaviour is PSPACE-hard. This second lattice, however, is not linear-space-computable-encoded. The best previously known lower bound for the complexity of computing the behaviour was the polynomial-time lower bound obtained for distributive lattices [6,16]. Our results show that dropping the distributivity does increase the complexity of the problem. To the best of our efforts, we were unable to close the gap between the lower and upper bounds found; however, we conjecture that the upper bounds are tight.

The paper is divided as follows. We first recall basic concepts from lattice and automata theory, and formally define the decision problem we study. Then, in Section 3 we provide upper bounds for the complexity of this problem by means of an algorithm. The lower bounds are provided in Section 4, before concluding the paper.

2. Lattice tree automata

We study a simple class of weighted automata that receive as input infinite unlabelled trees of a fixed arity k, and use elements of a possibly infinite lattice as weights. For a positive integer k, we denote the set $\{1,\dots,k\}$ by $[k]$. We identify the nodes of an infinite tree by words from $[k]^*$ in the usual way: the root node is represented by the empty word ε, and the i-th successor of a node u is represented by ui for $i, 1 \leqslant i \leqslant k$. In the case of labelled trees, we refer to the labelling of the node $u \in [k]^*$ in the tree r by $r(u)$. An infinite tree r with labels from a set Q can be described as a function $r : [k]^* \to Q$.

As previously said, we consider only unlabelled trees as inputs for our automata. Given an arity k, there is exactly one such tree, that we simply call $[k]^*$. We will often refer to paths in this tree. A *path* is a subset $\mathsf{p} \subseteq [k]^*$ that contains the empty word ($\varepsilon \in \mathsf{p}$), is closed under prefixes (i.e., if $ui \in \mathsf{p}$, then $u \in p$ for every $u \in [k]^*, i \in [k]$), and every node has exactly one successor (that is, if $u \in \mathsf{p}$, then there is exactly one $i \in [k]$ with $ui \in \mathsf{p}$).

We call the unary tree, where $k = 1$, an *infinite word*. Notice that an infinite word has exactly one path that is equivalent to the word itself. As usual, we will often represent an infinite word with labels from a set Q as an infinite sequence of elements from Q.

[2] The actual weights used are slightly more complex than described here. For the full details see [6].