Antichain-Based Universality and Inclusion Testing over Nondeterministic Finite Tree Automata*

Ahmed Bouajjani¹, Peter Habermehl¹,², Lukáš Holík³, Tayssir Touili¹, and Tomáš Vojnar³

¹ LIAFA, CNRS and University Paris Diderot, France
{abou,haberm,touili}@liafa.jussieu.fr
² LSV, ENS Cachan, CNRS, INRIA
³ FIT, Brno University of Technology, Czech republic
{holik,vojnar}@fit.vutbr.cz

Abstract. We propose new antichain-based algorithms for checking universality and inclusion of nondeterministic tree automata (NTA). We have implemented these algorithms in a prototype tool and our experiments show that they provide a significant improvement over the traditional determinisation-based approaches. We use our antichain-based inclusion checking algorithm to build an abstract regular tree model checking framework based entirely on NTA. We show the significantly improved efficiency of this framework through a series of experiments with verifying various programs over dynamic linked tree-shaped data structures.

1 Introduction

Tree automata are useful in numerous different areas, including, e.g., the implementation of decision procedures for various logics, XML manipulation, linguistics or formal verification of systems, such as parameterised networks of processes, cryptographic protocols, or programs with dynamic linked data structures. A classical implementation of many of the operations, such as minimisation or inclusion checking, used for dealing with tree automata in the different application areas often assumes that the automata are deterministic. However, as our own practical experience discussed later in the paper shows, the determinisation step may yield automata being too large to be handled although the original nondeterministic automata are quite small. It may even be the case that the corresponding minimal deterministic automata are small, but they cannot be computed as the intermediary automata resulting from determinisation are too big.

As the situation is similar for other kinds of automata, recently, a lot of research has been done to implement efficiently operations like minimisation (or at least reduction) and universality or inclusion checking on nondeterministic word, Büchi, or tree automata. We follow this line of work and propose and experimentally evaluate new efficient algorithms for universality and inclusion checking on nondeterministic (bottom-up) tree automata. Instead of the classical subset construction, we use antichains of sets

* This work was supported by the French projects ANR-06-SETI-001 AVERISS and RNTL AVERILES, the Czech Grant Agency (projects 102/07/0322, 102/05/H050), the Barrande project 17356TD, and the Czech Ministry of Education by the project MSM 0021630528 Security-Oriented Research in Information Technology.
of states of the considered automata and extend some of the antichain-based algorithms recently proposed for universality and inclusion checking over finite word automata [12] to tree automata (while also showing that the others are not practical for them).

To evaluate the proposed algorithms, we have implemented them in a prototype tool over the Timbuk tree automata library [9] and tested them in a series of experiments showing that they provide a significant advantage over the traditional determinisation-based approaches. The experiments were done on randomly generated automata with different densities of transitions and final states like in [12] as well as within an important complex application of tree automata. Indeed, our antichain-based inclusion checking algorithm for tree automata fills an important hole in the tree automata technology enabling us to implement an abstract regular tree model checking (ARTMC) framework based entirely on nondeterministic tree automata. ARTMC is a generic technique for automated formal verification of various kinds of infinite-state and parameterised systems. In particular, we consider its use for verification of programs manipulating dynamic tree-shaped data structures, and we show that the use of nondeterministic instead of deterministic tree automata improves significantly the efficiency of the technique.

Related Work. In [12], antichains were used for dual forward and backward algorithms for universality and inclusion testing over finite word automata. In [8], antichains were applied for Büchi automata. Here, we show how the forward algorithms from [12] can be extended to finite (bottom-up) tree automata (using algorithms computing upwards). We also show that the backward computation from word automata is not practical for tree automata (where it corresponds to a downward computation). The regular tree model checking framework was studied in, e.g., [11,6,2], and its abstract version in [45]—in all cases using deterministic tree automata. When implementing a framework for abstract regular tree model checking based on nondeterministic tree automata, we exploit the recent results [1] on simulation-based reduction of tree automata.

2 Preliminaries

An alphabet $\Sigma$ is ranked if it is endowed with a mapping $\text{rank} : \Sigma \to \mathbb{N}$. For $k \geq 0$, $\Sigma_k = \{ f \in \Sigma \mid \text{rank}(f) = k \}$ is the set of symbols of rank $k$. The set $T_\Sigma$ of terms over $\Sigma$ is defined inductively: if $k \geq 0$, $f \in \Sigma_k$, and $t_1, \ldots, t_k \in T_\Sigma$, then $f(t_1, \ldots, t_k)$ is in $T_\Sigma$. We abbreviate the so-called leaf terms of the form $a()$, $a \in \Sigma_0$, by simply $a$.

A (nondeterministic, bottom-up) tree automaton (NTA) is a tuple $A = (Q, \Sigma, F, \delta)$ where $Q$ is a finite set of states, $\Sigma$ is a ranked alphabet, $F \subseteq Q$ is a set of final states, and $\delta$ is a set of rules of the form $f(q_1, \ldots, q_n) \to q$ where $n \geq 0$, $f \in \Sigma_n$, and $q_1, \ldots, q_n, q \in Q$. We abbreviate the leaf rules of the form $a() \to q$, $a \in \Sigma_0$, as $a \to q$. Let $t$ be a term over $\Sigma$. A bottom-up run of $A$ on $t$ is obtained as follows: first, we assign a state to each leaf according to the leaf rules in $\delta$, then for each internal node, we collect the states assigned to all its children and associate a state to the node itself according to the non-leaf $\delta$ rules. Formally, if during the state assignment process subterms $t_1, \ldots, t_n$ are labelled with states $q_1, \ldots, q_n$, and if a rule $f(q_1, \ldots, q_n) \to q$ is in $\delta$, which we will denote by $f(q_1, \ldots, q_n) \to_\delta q$, then the term $f(t_1, \ldots, t_n)$ can be labelled with $q$. A term $t$ is accepted if $A$ reaches its root in a final state. The language