Composed Bisimulation for Tree Automata

FIT BUT Technical Report Series

Parosh A. Abdulla, Ahmed Bouajjani, Lukáš Holík, Lisa Kaati, and Tomáš Vojnar

Technical Report No. FIT-TR-2008-004
Faculty of Information Technology, Brno University of Technology

Last modified: October 14, 2008

Composed Bisimulation for Tree Automata*

Parosh A. Abdulla[1], Ahmed Bouajjani[2], Lukáš Holík[3],
Lisa Kaati[1], and Tomáš Vojnar[3]

[1] University of Uppsala, Sweden, email: {parosh,lisa.kaati}@it.uu.se
[2] LIAFA, University Paris 7, France, email: abou@liafa.jussieu.fr
[3] FIT, Brno University of Technology, Czech Rep., email:
{holik,vojnar}@fit.vutbr.cz

Abstract. We address the problem of reducing the size of (nondeterministic, bottom-up) tree automata (TA) using suitable, language-preserving equivalences on the states of the automata. In particular, we propose the so-called *composed bisimulation equivalence* as a new language preserving equivalence. A composed bisimulation equivalence is defined in terms of two different relations, namely the upward and downward bisimulation equivalence. We provide simple and efficient algorithms for computing these relations. The notion of composed bisimulation equivalence is motivated by an attempt to obtain an equivalence that can provide better reductions than what currently known bisimulation-based approaches can offer, but which is not significantly more difficult to compute (and hence stays below the computational requirements of simulation-based reductions). The experimental results we present in the paper show that our composed bisimulation equivalence meets such requirements, and hence provides users of TA with a finer way to resolve the trade-off between the available degree of reduction and its cost.

1 Introduction

Tree automata (TA) are widely used in many areas of computer science such as XML manipulation, natural language processing, or formal verification. For instance, in formal verification, TA are—among other uses—at the heart of the so-called regular tree model checking (RTMC) framework developed for a fully automated verification of infinite-state or parameterised systems such as parameterised networks of processes with a tree-like topology or programs with dynamic linked data-structures [9, 5, 7, 8]. In RTMC, TA are, in particular, used to finitely represent and manipulate infinite sets of reachable configurations.

* 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*

In many applications of TA, such as in the above mentioned RTMC framework, it is highly desirable to deal with automata which are as small as possible, in order to save memory as well as time. In theory, one can always determinise and minimise any given (bottom-up) tree automaton. However, the determinisation step may lead to an exponential blow-up in the size of the TA. Therefore, even if the minimal deterministic TA is small, it might not be feasible to compute it in practice because of the expensive determinisation step. Moreover, the minimal deterministic TA may still be bigger than the original non-deterministic TA.

To avoid determinisation, a TA can be reduced by identifying and collapsing states that are equal wrt a suitable equivalence relation that preserves the language of the automaton. One such equivalence is *downward bisimulation* (also called backward bisimulation) *equivalence* considered in [11]. For a given TA A, the downward bisimulation equivalence can be computed efficiently in time $\mathcal{O}(\hat{r}^2\, m \log n)$ where $\hat{r}$ is the maximal rank of the input symbols, m the size of the transition table, and n the number of states of A. Unfortunately, the reduction obtained by using downward bisimulation equivalence might be limited.

To get a better reduction, some simulation-based equivalence (as, e.g., *downward simulation equivalence* or *composed simulation equivalence* [1]) can be used. Simulation-based relations are commonly weaker than those based on bisimulations and hence they can offer a better reduction. On the other hand, they are considerably harder to compute—in particular, the time complexity of computing bisimulations on TA is in $\mathcal{O}(mn)$. Hence, despite the recent advances in efficient heuristics for computing simulation relations on TA [1], the choice between bisimulations and simulations is a trade-off between the time consumption of the reduction and the achieved degree of reduction.

In this paper, we propose a new notion of *composed bisimulation equivalence*, which is a composition of *downward bisimulation equivalence* and its dual *upward bisimulation equivalence* (also proposed in the paper).[4] The proposal is motivated by an attempt to obtain a relation which is still easy to compute and, on the other hand, can give a better reduction than downward bisimulation equivalence, and hence give users of TA a finer choice in the above mentioned trade-off.

We then also discuss how upward bisimulation equivalence (which is a basis for computing composed bisimulation equivalence) can be com-

[4] Note that a composed bisimulation equivalence is not itself a bisimulation, but rather a relation built from bisimulations.

puted in an efficient way.[5] Inspired by the approach of [1], we show how the computation of upward bisimulation equivalence can be reduced to computing (word) bisimulation equivalence on suitable *transition systems* derived from the automata at hand. This transformation allows us to reuse the results proposed for an efficient computation of (word) bisimulation equivalence on transition systems (or, equivalently, Kripke structures or finite word automata).

We have implemented a prototype tool in which we have performed practical experiments with using the proposed composed bisimulation framework for reducing TA. Our experimental results show that composed bisimulation equivalence indeed reduces the size of TA much more than downward bisimulation equivalence and more than downward simulation equivalence, but, as expected, less than composed simulation equivalence. Computationally, composed bisimulation equivalence is, of course, more difficult to compute than downward bisimulation equivalence, but it is still much easier to compute than all simulation-based relations.

Related work. Several algorithms for reducing the size of non-deterministic tree automata while preserving their language have been proposed in the literature. The first attempt was done in [3] where an algorithm inspired by the partition refinement algorithm by Paige and Tarjan [13] was presented.

In [11], two different types of bisimulations—namely, backward and forward bisimulations—are presented. The concept of backward bisimulations corresponds to downward bisimulations used here. Forward bisimulations are even cheaper to compute than backward bisimulations and turn out to be especially well-suited for reducing deterministic TA. Moreover, [11] also shows that by using backward bisimulations followed by forward bisimulations (or vice versa), one can get a better reduction than using any of the methods alone. The reduction power of the sequential applications of these relations is in general incomparable with the use of our composed bisimulation equivalence. Moreover, all of these approaches differ in various practical aspects of their use as we discuss at the end of the paper.

Efficient algorithms for computing simulation equivalences over TA have been discussed in [1]. Our method for computing upward bisimulation equivalences (and possibly also downward bisimulation equivalences)

[5] Downward bisimulation equivalence can be computed by an algorithm proposed in [11], or a similar approach as in the case of upward bisimulation equivalence can be used too.

is inspired by the approach of [1], which we here extend to cope with bisimulation relations.

Plan of the paper. In the next section, we give some preliminaries on tree automata and transition systems. In Section 3, we present the upward and downward bisimulation equivalences. In Section 4, we discuss how these relations can be computed. Next, in Section 5, composed bisimulation equivalence is proposed. In Section 6, we present our experimental results and a further comparison of the various existing as well as newly proposed relations suitable for reducing TA. Finally, in Section 7, we give some concluding remarks and directions for future work.

2 Preliminaries

In this section, we introduce some preliminaries on trees, tree automata, and transition systems (TS).

For an equivalence relation $\equiv$ defined on a set Q, we call each equivalence class of $\equiv$ a *block*, and use $Q/\equiv$ to denote the set of blocks in $\equiv$.

Trees. A *ranked alphabet* Σ is a set of symbols together with a function $Rank : \Sigma \to \mathbb{N}$. For $f \in \Sigma$, the value $Rank(f)$ is called the *rank* of f. For any $n \geq 0$, we denote by Σ_n the set of all symbols of rank n from Σ. Let ϵ denote the empty sequence. A *tree* t over an alphabet Σ is a partial mapping $t : \mathbb{N}^* \to \Sigma$ that satisfies the following conditions:

- $dom(t)$ is a finite, prefix-closed subset of $\mathbb{N}^*$, and
- for each $p \in dom(t)$, if $Rank(t(p)) = n \geq 0$, then $\{i \mid pi \in dom(t)\} = \{1, \ldots, n\}$.

Each sequence $p \in dom(t)$ is called a *node* of t. For a node p, we define the i^{th} *child* of p to be the node pi, and we define the i^{th} *subtree* of p to be the tree t' such that $t'(p') = t(pip')$ for all $p' \in \mathbb{N}^*$. A *leaf* of t is a node p which does not have any children, i.e., there is no $i \in \mathbb{N}$ with $pi \in dom(t)$. We denote by $T(\Sigma)$ the set of all trees over the alphabet Σ.

Tree Automata. A (finite, non-deterministic, bottom-up) *tree automaton* (TA) is a 4-tuple $A = (Q, \Sigma, \Delta, F)$ where Q is a finite set of states, $F \subseteq Q$ is a set of final states, Σ is a ranked alphabet, and Δ is a set of transition rules. Each transition rule is a triple of the form $((q_1, \ldots, q_n), f, q)$ where $q_1, \ldots, q_n, q \in Q$, $f \in \Sigma$, and $Rank(f) = n$. We use $(q_1, \ldots, q_n) \xrightarrow{f} q$ to denote that $((q_1, \ldots, q_n), f, q) \in \Delta$. In the special case of $n = 0$, we

speak about the so-called *leaf rules*, which we may abbreviate as $\xrightarrow{f} q$. We use $Lhs(A)$ to denote the set of *left-hand sides* of rules, i.e., the set of tuples of the form $((q_1, \ldots, q_n), f)$ where $(q_1, \ldots, q_n) \xrightarrow{f} q$ for some q. Finally, we denote by $Rank(A)$ the smallest $n \in \mathbb{N}$ such that $n \geq m$ for each $m \in \mathbb{N}$ where $(q_1, \ldots, q_m) \in Lhs(A)$ for some $q_i \in Q$, $1 \leq i \leq m$.

A *run* of A over a tree $t \in T(\Sigma)$ is a mapping $\pi : dom(t) \to Q$ such that for each node $p \in dom(t)$ where $q = \pi(p)$, we have that if $q_i = \pi(pi)$ for $1 \leq i \leq n$, then Δ has a rule $(q_1, \ldots, q_n) \xrightarrow{t(p)} q$. We write $t \overset{\pi}{\Longrightarrow} q$ to denote that π is a run of A over t such that $\pi(\epsilon) = q$. We use $t \Longrightarrow q$ to denote that $t \overset{\pi}{\Longrightarrow} q$ for some run π. The *language* of a state q is defined by $L(q) = \{t | \, t \Longrightarrow q\}$, while the *language* of A is defined by $L(A) = \bigcup_{q \in F} L(q)$.

An *environment* is a tuple of the form $((q_1, \ldots, q_{i-1}, \Box, q_{i+1}, \ldots, q_n), f, q)$ obtained by removing a state q_i, $1 \leq i \leq n$, from the i^{th} position of the left hand side of a rule $((q_1, \ldots, q_{i-1}, q_i, q_{i+1}, \ldots, q_n), f, q)$, and by replacing it by a special symbol $\Box \notin Q$ (called a *hole* below). Like for transition rules, we write $(q_1, \ldots, \Box, \ldots, q_n) \xrightarrow{f} q$ provided that $((q_1, \ldots, q_{i-1}, q_i, q_{i+1}, \ldots, q_n), f, q) \in \Delta$ for some $q_i \in Q$. Sometimes, we also write the environment as $(q_1, \ldots, \Box_i, \ldots, q_n) \xrightarrow{f} q$ to emphasise that the hole is at position i. We denote the set of all environments of A by $Env(A)$.

Transition Systems. A (finite) *transition system (TS)* is a pair $T = (Q, \Delta)$ where Δ is a finite set of states and $\to \subseteq Q \times Q$ is a transition relation. Given a TS $T = (Q, \Delta)$ and two states $q, r \in Q$, we denote by $q \longrightarrow r$ the fact that $(q, r) \in \Delta$.

A (finite) *labeled transition system (LTS)* is a quadruple $T = (Q, \Delta, \Sigma, \mathcal{L})$ where (Q, Δ) is a transition system, Σ is a finite set of labels and $\mathcal{L} : \Delta \to 2^{\Sigma}$ is a labeling function that assigns a set of labels to each transition of Δ. Given an LTS $T = (Q, \Delta, \Sigma, \mathcal{L})$ and two states $q, r \in Q$, we denote by $q \xrightarrow{a} r$ the fact that $(q, r) \in \Delta$ and $a \in \mathcal{L}((q, r))$.

3 Downward and Upward Bisimulation

In this section, we present two different relations on states of tree automata—namely, *downward bisimulations* and *upward bisimulations*—that we will subsequently use as a basis for defining composed bisimulations, which we will then exploit for reducing tree automata.

Downward Bisimulation. For a tree automaton $A = (Q, \Sigma, \Delta, F)$, a *downward bisimulation* D is a binary relation on Q such that if $(q, r) \in D$, then

(i) whenever there are $q_1, \ldots, q_n \in Q, f \in \Sigma$ such that $(q_1, \ldots, q_n) \xrightarrow{f} q$, then there are $r_1, \ldots, r_n \in Q$ such that $(r_1, \ldots, r_n) \xrightarrow{f} r$ and $(q_i, r_i) \in D$ for each $i : 1 \leq i \leq n$; and, symmetrically,

(ii) whenever there are $r_1, \ldots, r_n \in Q, f \in \Sigma$ such that $(r_1, \ldots, r_n) \xrightarrow{f} r$, then there are $q_1, \ldots, q_n \in Q$ such that $(q_1, \ldots, q_n) \xrightarrow{f} q$ and $(q_i, r_i) \in D$ for each $i : 1 \leq i \leq n$.

Lemma 1. *For each tree automaton there exists a unique maximal downward bisimulation which is an equivalence.*

The proof of Lemma 1 is straightforward and similar to the (slightly more complicated) proof of Lemma 2 presented below—and so we leave it to the reader. The maximal downward bisimulation, denoted as the *downward bisimulation equivalence* or $\simeq$ below, corresponds to the maximal backward bisimulation from [11].

Upward Bisimulation. Given a tree automaton $A = (Q, \Sigma, \Delta, F)$ and a downward bisimulation D, an *upward bisimulation* U wrt D is a binary relation on Q such that if $(q, r) \in U$, then

(i) whenever there are $q_1, \ldots, q_n, q' \in Q, f \in \Sigma$ such that $q_i = q$ and $(q_1, \ldots, q_n) \xrightarrow{f} q'$, then there are $r_1, \ldots, r_n, r' \in Q$ s.t. $(r_1, \ldots, r_n) \xrightarrow{f} r'$, $r_i = r$, $(q', r') \in R$, and $q_j \simeq r_j$ for each $j : 1 \leq j \neq i \leq n$; and symmetrically,

(ii) whenever there are $r_1, \ldots, r_n, r' \in Q, f \in \Sigma$ such that $r_i = r$ and $(r_1, , \ldots, r_n) \xrightarrow{f} r'$, then there are $q_1, \ldots, q_n, q' \in Q$ s.t. $(q_1, \ldots, q_n) \xrightarrow{f} q'$, $q_i = q$, $(q', r') \in R$ and $q_j \simeq r_j$ for each $j : 1 \leq j \neq i \leq n$; and, finally,

(iii) $q \in F$ iff $r \in F$.

Lemma 2. *For each tree automaton A and a downward bisimulation D which is an equivalence there exists a unique maximal upward bisimulation wrt D which is also an equivalence.*

Proof. Let $A = (Q, \Sigma, \Delta, F)$. We will prove that the set of all upward bisimulations wrt D is closed under reflexive and transitive closure and under union, from which the lemma directly follows.

(Closure under Union) Given two upward bisimulations U_1 and U_2 wrt D, we want to prove that $U = U_1 \cup U_2$ is also an upward bisimulation wrt D. Let qUr for some $q, r \in Q$, then either qU_1r or qU_2r. Assume without loss of generality that qU_1r. Then, from the definition of upward bisimulations, $q \in F \iff r \in F$, and there are $q_1, \ldots, q_2, q' \in Q$ with $q_i = q$ such that $(q_1, \ldots, q_n) \xrightarrow{f} q'$ if and only if there are $r_1, \ldots, r_2, r' \in Q$ such that $r_i = r$, $q'U_1r'$, q_jDr_j for all $j : 1 \leq j \neq i \leq n$, and $(r_1, \ldots, r_n) \xrightarrow{f} r'$. As $U_1 \subseteq U$ gives $q'Ur'$, U fulfils the definition of upward bisimulations wrt D.

(Reflexive Closure) It can be seen from the definition that the identity is an upward bisimulation wrt D for any downward bisimulation D. Therefore, from the closure under union, the union of the identity and any upward bisimulation wrt D is an upward bisimulation wrt D.

(Transitive Closure) Let U be an upward bisimulation wrt D and let U_T be its transitive closure. We will prove that U_T is also an upward bisimulation wrt D. Let qU_Tr for some $q, r \in Q$. Suppose that there exist $s_1^1, \ldots, s_n^1, t^1 \in Q$ such that $q = s_i^1, 1 \leq i \leq n$, and $(s_1^1, \ldots, s_n^1) \xrightarrow{f} t^1$. From $s_i^1U_Tr$, we have that there are states $s_i^1, \ldots, s_i^m \in Q$ with $s_i^m = r$ such that $s_i^1Us_i^2U \ldots Us_i^m$. From the definition of upward bisimulation, this implies that there are states $s_1^2, \ldots, s_n^2, t^2, \ldots, s_1^m, \ldots, s_n^m, t^m \in Q$ such that $(s_1^2, \ldots, s_n^2) \xrightarrow{f} t^2, \ldots, (s_1^m, \ldots, s_n^m) \xrightarrow{f} t^m$, $t^1U \ldots Ut^m$, $s_i^1 \in F \iff \ldots \iff s_i^m \in F$, and $s_j^1D \ldots Ds_j^m$ for all $j : 1 \leq j \neq i \leq n$. From transitivity of U_T, we have $t^1U_Tt^m$; from transitivity of $\iff$, we have $s_i^1 \in F \iff s_i^m \in F$; and from transitivity of D, we have $s_j^1Ds_j^m$ for all $j : 1 \leq j \neq i \leq n$. We have thus proven that U_T fulfils Conditions (i) and (iii) from the definition of upward bisimulation wrt D, Condition (ii) can be proven in a symmetrical way to Condition (i), hence U_T is an upward bisimulation wrt D. □

In the sequel, we will use $\overset{\bullet}{\simeq}$ to denote the (unique) maximal upward bisimulation wrt $\simeq$, which we call the *upward bisimulation equivalence*.

4 Computing Downward and Upward Bisimulation Equivalences

In this section, we describe how the bisimulation equivalences described in the previous section can be computed.

4.1 Computing Downward Bisimulation Equivalences

For computing the downward bisimulation equivalence, one can use the specialised algorithm proposed for computing the equal maximal backward bisimulation in [11]. This algorithm runs in time $\mathcal{O}(\hat{r}^2\, m \log n)$ where m is the number of transitions, n is the number of states, and $\hat{r}$ is the maximal rank of the alphabet.

An alternative approach to computing the downward bisimulation equivalence is to use an analogy of the method presented in [1], where an approach for computing maximal downward simulations on tree automata via their translation to certain specialised labeled transition systems (LTSs) is proposed. Downward simulations are then computed on the generated LTSs using standard simulation algorithms such as [10, 14]. Since downward bisimulation equivalence is a bisimulation counterpart of downward simulation equivalence, the LTSs generated for computing the latter can also be exploited for computing the former using standard algorithms for computing (word) bisimulations such as [13]. This method gives us an algorithm which is easy to implement and runs in time $\mathcal{O}(\hat{r}^3\, m \log n)$.

We now give the details of the translation from the downward bisimulation problem on a tree automaton $A = (Q, \Sigma, \Delta, F)$ to the bisimulation problem on a derived LTS.

Consider a LTS $(Q, \Delta, \Sigma, \mathcal{L})$. A *bisimulation* is an equivalence relation on Q such that if $(q, r) \in R$, then $q \xrightarrow{a} q'$ for some q' if and only if $r \xrightarrow{a} r'$ for some r' such that $(q', r') \in R$. For an equivalence relation I, we use $\cong_I$ to denote the largest bisimulation which is included in I.

We derive a LTS $A^{\odot} = (Q^{\odot}, \Delta^{\odot}, \Sigma^{\odot}, \mathcal{L})$ and an so called *initial equivalence* I on the set $Q^{\odot}$ as follows:

- The set $Q^{\odot}$ contains a state $q^{\odot}$ for each state $q \in Q$, and it also contains a state $(q_1, \ldots, q_n)^{\odot}$ for each $(q_1, \ldots, q_n) \in Lhs(A)$.
- The set $\Sigma^{\odot}$ contains a symbol ϵ and each index $i \in \{1, 2, \ldots, n\}$ where n is the maximal rank of any symbol in Σ.
- For each transition rule $(q_1, \ldots, q_n) \xrightarrow{f} q$ of A, $A^{\odot}$ contains both the transition $q^{\odot} \xrightarrow{\epsilon} ((q_1, \ldots, q_n), f)^{\odot}$ and $((q_1, \ldots, q_n), f)^{\odot} \xrightarrow{i} q_i^{\odot}$ for each $i : 1 \leq i \leq n$.
- The sets $Q^{\odot}$, $\Sigma^{\odot}$, $\Delta^{\odot}$ and $\mathcal{L}$ do not contain any other elements.

Furthermore, we define the initial equivalence I to be the smallest relation containing the following elements:

- $(q_1^{\odot}, q_2^{\odot}) \in I$ for all states $q_1, q_2 \in Q$.

– for left-hand sides l_1 and l_2, $(l_1^{\odot}, l_2^{\odot}) \in I$ if they are of the form $l_1 = ((q_1, \ldots, q_n)), f)$ and $l_2 = ((r_1, \ldots, r_n)), f)$. In other words, the two left-hand sides has the same arity and share the same symbol.

The following theorem shows correctness of the translation.

Theorem 1. *For all $q, r \in Q$, we have $q^{\odot} \cong_I r^{\odot}$ iff qDr.*

Proof. (if) Suppose that $q^{\odot} \cong_I r^{\odot}$. This means that there is a bisimulation $R^{\odot}$ on $Q^{\odot}$ such that $(q^{\odot}, r^{\odot}) \in R^{\odot}$. We define D to be the smallest binary relation on Q such that $(q', r') \in D$ if $(q'^{\odot}, r'^{\odot}) \in R^{\odot}$. Obviously, $(q, r) \in D$. We show that D is a downward bisimulation on Q which immediately implies the result.

Suppose that $(q', r') \in D$ and $(q_1, \ldots, q_n) \xrightarrow{f} q'$. Since $(q', r') \in D$ we know that $(q'^{\odot}, r'^{\odot}) \in R^{\odot}$; and since $(q_1, \ldots, q_n) \xrightarrow{f} q'$ we know by definition of $A^{\odot}$ that $q'^{\odot} \xrightarrow{\epsilon} ((q_1, \ldots, q_n), f)^{\odot}$. Since $R^{\odot}$ is a bisimulation and $R^{\odot} \subseteq I$, there are $r_1, \ldots, r_n \in Q, g \in \Sigma$ with $r'^{\odot} \xrightarrow{\epsilon} ((r_1, \ldots, r_n), g)^{\odot}$ and $(((q_1, \ldots, q_n), f)^{\odot}, ((r_1, \ldots, r_n), f)^{\odot}) \in R^{\odot}$. Since $r'^{\odot} \xrightarrow{\epsilon} ((r_1, \ldots, r_n), f)^{\odot}$ we have $(r_1, \ldots, r_n) \xrightarrow{f} r'$. Also, by definition of $A^{\odot}$ we know that $((q_1, \ldots, q_n), f)^{\odot} \xrightarrow{i} q_i^{\odot}$ for each $i : 1 \leq i \leq n$. We observe that r_i is the only state such that $((r_1, \ldots, r_n), f)^{\odot} \xrightarrow{i} r_i^{\odot}$, and hence it must be the case that $(q_i^{\odot}, r_i^{\odot}) \in R^{\odot}$. This means that $(q_i, r_i) \in D$ for each $i : 1 \leq i \leq n$.

(only if) Suppose that qDr. This means that there is a bisimulation D on Q such that $(q, r) \in D$. We define $R^{\odot}$ to be the smallest binary relation on $Q^{\odot}$ such that $(q'^{\odot}, r'^{\odot}) \in R^{\odot}$ if $(q', r') \in D$, and $(((q_1, \ldots, q_n)^{\odot}, f), ((r_1, \ldots, r_n)^{\odot}, f)) \in R^{\odot}$ if $(q_i, r_i) \in D$ for each $i : 1 \leq i \leq n$. Obviously, $(q, r) \in R^{\odot}$. We show that $R^{\odot}$ is a simulation on $Q^{\odot}$ which immediately implies the result. In the proof, we consider two sorts of states in $A^{\odot}$; namely those corresponding to states and those corresponding to left hand sides in A.

Suppose that $(q'^{\odot}, r'^{\odot}) \in R^{\odot}$ and $q'^{\odot} \xrightarrow{\epsilon} ((q_1, \ldots, q_n), f)^{\odot}$. Since $(q'^{\odot}, r'^{\odot}) \in R^{\odot}$, we know that $(q', r') \in D$, and as $q'^{\odot} \xrightarrow{\epsilon} ((q_1, \ldots, q_n), f)^{\odot}$, we know by definition of $A^{\odot}$ that $(q_1, \ldots, q_n) \xrightarrow{f} q'$. Since D is a downward simulation, there are $r_1, \ldots, r_n \in Q$ with $(r_1, \ldots, r_n) \xrightarrow{f} r'$ and $(q_i, r_i) \in D$ for each $i : 1 \leq i \leq n$. Since $(r_1, \ldots, r_n) \xrightarrow{f} r'$, we have $r'^{\odot} \xrightarrow{\epsilon} ((r_1, \ldots, r_n), f)^{\odot}$. By definition of $R^{\odot}$, it follows that $(((q_1, \ldots, q_n), f)^{\odot}, ((r_1, \ldots, r_n), f)^{\odot}) \in R^{\odot}$.

Now, suppose that $(((q_1, \ldots, q_n), f)^{\odot}, ((r_1, \ldots, r_n), f)^{\odot}) \in R^{\odot}$ and that

$(q_1, \ldots, q_n), f)^{\odot} \xrightarrow{i} q_i^{\odot}$. We know that $((r_1, \ldots, r_n), f)^{\odot} \xrightarrow{i} r_i^{\odot}$ from definition of $A^{\odot}$. Since $(((q_1, \ldots, q_n)), f)^{\odot}, ((r_1, \ldots, r_n), f)^{\odot}) \in R^{\odot}$, it follows by definition of $R^{\odot}$ that $(q_i, r_i) \in D$ and hence also that $(q_i^{\odot}, r_i^{\odot}) \in R^{\odot}$. □

4.2 Complexity of Computing Downward Bisimulation via LTS Translation

We analyse the complexity of computing downward bisimulation using the translation scheme presented above. Let $m = |\Delta|$, $n = |Q|$, $\hat{r} = Rank(A)$, and $p = |\Sigma|$.

The initial equivalence I can be computed in time $\mathcal{O}(\hat{r}m + p)$. Furthermore, we observe that $|\Sigma^{\odot}| = \mathcal{O}(r)$, $|Q^{\odot}| = \mathcal{O}(n + m) = \mathcal{O}(m)$, and $|\Delta^{\odot}| = \mathcal{O}(\hat{r}m)$. From the Paige-Tarjan algorithm [13] (more precisely, from its generalization allowing multiple labels as in [2]), we know that we can compute $\cong_I$ in time $\mathcal{O}(|\Sigma^{\odot}|\,|\Delta^{\odot}| \log |Q^{\odot}|)$. Therefore, the time complexity of using our method for computing upward bisimulation amounts to $\mathcal{O}\left(\hat{r}^2 m \log m\right) \leq \mathcal{O}\left(\hat{r}^2 m \log\left(n^{\hat{r}} p\right)\right) = \mathcal{O}\left(\hat{r}^3 m \log(np)\right)$.

4.3 Computing Upward Bisimulation Equivalences

Similarly to the second above mentioned approach to computing downward bisimulation equivalences, we can translate the problem of computing upward bisimulation equivalences on TA to computing (word) bisimulation equivalences on suitable TSs. This translation is analogical to the translation used for computing the maximal upward simulations in [1].[6] Note that no specialised algorithm for computing the upward bisimulation equivalence like the one for computing the downward bisimulation equivalence from [11] is available.

Consider a transition system (Q, Δ). A *(word) bisimulation* is a binary relation R on Q such that if $(q, r) \in R$, then whenever $q \xrightarrow{a} q'$ for some $q' \in Q, a \in \Sigma$, then $r \xrightarrow{a} r'$ for some $r' \in Q$ such that $(q', r') \in R$, and symmetrically, whenever $r \xrightarrow{a} r'$ for some $r' \in Q, a \in \Sigma$, then $q \xrightarrow{a} q'$ for some $q' \in Q$ such that $(r', q') \in R$. For the so called initial equivalence relation I over Q, one can easily show that there is a unique largest bisimulation included in I which is an equivalence and which we denote by $\cong_I$ in what follows.

[6] In [1], the translation is done towards LTSs in order to have a uniform approach with the downward simulations. However, the labels are not really needed for computing upward simulations nor bisimulations and so we optimise the algorithm a bit here.

We translate the computation of the upward bisimulation equivalence on TA into computing the (word) bisimulation equivalence on TSs. Consider a TA $A = (Q, \Sigma, \Delta, F)$ and the downward bisimulation equivalence $\simeq$. We derive a TS $A^\bullet = (Q^\bullet, \Delta^\bullet)$ and an initial equivalence I on the set $Q^\bullet$ as follows:

- The set $Q^\bullet$ contains a state $q^\bullet$ for each state $q \in Q$ and a state $e^\bullet$ for each environment $e \in Env(A)$.
- The set $\Delta^\bullet$ is the smallest set such that if $(q_1, \ldots, q_n) \xrightarrow{f} q$, where $1 \leq i \leq n$, then the set $\Delta^\bullet$ contains both $q_i^\bullet \longrightarrow e_i^\bullet$ and $e_i^\bullet \longrightarrow q^\bullet$, where e_i is of the form $(q_1, \ldots, \Box_i, \ldots, q_n) \xrightarrow{f} q$.

Furthermore, we define the initial equivalence I to be the smallest relation containing the following elements:

- $(q_1^\bullet, q_2^\bullet)$ for all states $q_1, q_2 \in Q$ such that $q_1 \in F \iff q_2 \in F$.
- $(e_1^\bullet, e_2^\bullet)$ if the environments e_1, e_2 are of the forms $(q_1, \ldots, \Box_i, \ldots, q_n) \xrightarrow{f} q$ and $(r_1, \ldots, \Box_i, \ldots, r_n) \xrightarrow{f} r$, respectively, $q_j \simeq r_j$ for each $j : 1 \leq j \neq i \leq n$, and $q \in F$ iff $r \in F$. In other words, the two environments share the same label, and, moreover, the respective states in the left hand sides are equivalent wrt $\simeq$ at all positions except position i. Furthermore, the states in the right hand sides agree on their membership in F.

The following theorem shows correctness of the translation, i.e., it allows us to compute $\overset{\bullet}{\simeq}$ by (i) computing $A^\bullet$; (ii) computing I; and (iii) computing $\cong_I$ by running a word bisimulation algorithm on $A^\bullet$ and I.

Theorem 2. *For all $q, r \in Q$, we have $q \overset{\bullet}{\simeq} r$ iff $q^\bullet \cong_I r^\bullet$.*

Proof. (if) Suppose that $q^\bullet \cong_I r^\bullet$. We show that $q \overset{\bullet}{\simeq} r$. We define a binary relation R on Q such that $R = \{(s_1, s_2) \mid s_1^\bullet \cong_I s_2^\bullet\}$. Obviously, $(q, r) \in R$, and hence if we show that R is an upward bisimulation wrt $\simeq$, we have $q \overset{\bullet}{\simeq} r$.

It remains to show that R is indeed an upward bisimulation wrt $\simeq$. Assume $(s_1, s_2) \in R$ and assume that there is a transition rule of the form $(q_1, \ldots, q_n) \xrightarrow{f} q_{n+1}$, where $q_i = s_1$. Since $(s_1, s_2) \in R$, we know that $s_1^\bullet \cong_I s_2^\bullet$. By definition, we have $s_1^\bullet \longrightarrow e_1^\bullet$ where e_1 is of the form $(q_1, \ldots, \Box_i, \ldots, q_n) \xrightarrow{f} q_{n+1}$. Since $s_1^\bullet \cong_I s_2^\bullet$, we know that there is an $e_2^\bullet$ such that $s_2^\bullet \longrightarrow e_2^\bullet$ and $e_2^\bullet \cong_I e_1^\bullet$. Since $\cong_I \subseteq I$, we get that $(e_1^\bullet, e_2^\bullet) \in I$, and hence e_2 is of the form $(r_1, \ldots, \Box_i, \ldots, r_n) \xrightarrow{f} r_{n+1}$, where $q_j \simeq r_j$

for each $j : 1 \leq j \neq i \leq n$. By definition, the only transition $e_1^\bullet$ can make is $e_1^\bullet \longrightarrow q_{n+1}^\bullet$; and the only transition $e_2^\bullet$ can make is $e_2^\bullet \longrightarrow r_{n+1}^\bullet$. From this, it follows that $q_{n+1}^\bullet \cong_I r_{n+1}^\bullet$ and therefore $(q_{n+1}, r_{n+1}) \in R$. Finally, consider states $s_1, s_2 \in Q$ such that $(s_1, s_2) \in R$. By definition, it follows that $s_1^\bullet \cong_I s_2^\bullet$. Since $\cong_I \subseteq I$, it follows that $(s_1^\bullet, s_2^\bullet) \in I$ and, according to the definition of I, we get that $s_1 \in F$ iff $s_2 \in F$.

(only if) Suppose that $q \stackrel{\bullet}{\simeq} r$. We show that $q^\bullet \cong_I r^\bullet$. Let R be the smallest binary relation on $Q^\bullet$ containing

- $(s_1^\bullet, s_2^\bullet)$ if $s_1 \stackrel{\bullet}{\simeq} s_2$,
- $(e_1^\bullet, e_2^\bullet)$ where e_1 is of the form $(q_1, \ldots, \Box_i, \ldots, q_n) \xrightarrow{f} q_{n+1}$, e_2 is of the form $(r_1, \ldots, \Box_i, \ldots, r_n) \xrightarrow{f} r_{n+1}$, $q_j \simeq r_j$ for each $j : 1 \leq j \neq i \leq n$, and $q_{n+1} \stackrel{\bullet}{\simeq} r_{n+1}$.

Obviously, $(q^\bullet, r^\bullet) \in R$, and hence if we show that R is a (word) bisimulation included in I, we have $q^\bullet \cong_I r^\bullet$.

We show that R is a bisimulation included in I. Assume that $(s_1^\bullet, s_2^\bullet) \in R$ and $s_1^\bullet \longrightarrow e_1^\bullet$, where e_1 is of the form $(q_1, \ldots, \Box_i, \ldots, q_n) \xrightarrow{f} q_{n+1}$. This means that $(q_1, \ldots, q_n) \xrightarrow{f} q_{n+1}$ where $q_i = s_1$. Since $(s_1^\bullet, s_2^\bullet) \in R$, it follows that $s_1 \stackrel{\bullet}{\simeq} s_2$ and therefore there are $r_1, \ldots, r_{n+1}$ where $(r_1, \ldots, r_n) \xrightarrow{f} r_{n+1}$, $r_i = s_2$, $r_j \simeq q_j$ for each $j : 1 \leq j \neq i \leq n$, and $r_{n+1} \stackrel{\bullet}{\simeq} q_{n+1}$. Define e_2 to be $(r_1, \ldots, \Box_i, \ldots, r_n) \xrightarrow{f} r_{n+1}$. Moreover, by definition, we also know that $s_2^\bullet \longrightarrow e_2^\bullet$ and that $(e_1^\bullet, e_2^\bullet) \in R$.

Assume that $(e_1^\bullet, e_2^\bullet) \in R$ and that $e_1^\bullet \longrightarrow s_1^\bullet$. Let e_1 be an environment of the form $(q_1, \ldots, \Box_i, \ldots, q_n) \xrightarrow{f} q_{n+1}$. Then we know by definition that $q_{n+1} = s_1$. Since $(e_1^\bullet, e_2^\bullet) \in R$, we know that e_2 is of the form $(r_1, \ldots, \Box_i, \ldots, r_n) \xrightarrow{f} r_{n+1}$ where $q_{n+1} \stackrel{\bullet}{\simeq} r_{n+1}$ (and therefore $(q_{n+1}^\bullet, r_{n+1}^\bullet) \in R$). By definition, we have that $e_2^\bullet \longrightarrow r_{n+1}^\bullet$.

Now we show that the $R \subseteq I$. If $(s_1^\bullet, s_2^\bullet) \in R$, then, by definition, we know that $s_1 \stackrel{\bullet}{\simeq} s_2$. This means that $s_1 \in F$ iff $s_2 \in F$ and hence $(s_1^\bullet, s_2^\bullet) \in I$.

Assume that $(e_1, e_2) \in R$. Let e_1 be of the form $(q_1, \ldots, \Box_i, \ldots, q_n) \xrightarrow{f} q_{n+1}$. It follows that e_2 is of the form $(r_1, \ldots, \Box_i, \ldots, r_n) \xrightarrow{f} r_{n+1}$ where $q_j \simeq r_j$ for each $j : 1 \leq j \neq i \leq n$ and $q_{n+1} \stackrel{\bullet}{\simeq} r_{n+1}$. According to the definition of $\stackrel{\bullet}{\simeq}$, it is the case that $q_{n+1} \in F$ iff $r_{n+1} \in F$, and hence we get that $(e_1^\bullet, e_2^\bullet) \in I$. $\square$

4.4 Complexity of Computing Upward Bisimulation Equivalences

We analyse the complexity of computing the upward bisimulation equivalence using the translation scheme presented above. Let $m = |\Delta|$, $n = |Q|$, $\hat{r} = \mathit{Rank}(A)$, and $p = |\Sigma|$. In order to be able to express complexity of our algorithm in these terms, we approximate $|Env(A)|$ by $\hat{r}m$.

Given the relation $\simeq$, we can compute the transition system $A^\bullet$ in time $\mathcal{O}(\hat{r}m \log(\hat{r}m))$. Firstly, we create the subset $\{q^\bullet \mid q \in Q\}$ of TS states. To create the subset $\{e^\bullet \mid e \in |Env(A)|\}$, for each rule, we create all possible environments that can arise from it by replacing a left-hand side state by $\Box$. To avoid duplicities, each of the newly created environments has to be checked whether it is created for the first time (this introduces the logarithmic factor into the complexity of the procedure). The transitions of the TS can be created while creating the set of environments (a constant time for each transition, the number of transition equals $2|Env(A)|$). Thus, the entire $A^\bullet$ can be constructed in time $\mathcal{O}(\hat{r}m \log(\hat{r}m))$.

Moreover, the initial equivalence I can be built in time $\mathcal{O}(\hat{r}^2 m)$. To compute I on environments, we first group them according to the position of the hole (time $\mathcal{O}(\hat{r}m)$). Each of these $\hat{r}$ groups will then be iteratively split into equivalence classes of I wrt non-hole left-hand side states (only those environments with left-hand side states downward bisimulation equivalent at the corresponding positions can be in the same class), wrt the symbol of the environment, and wrt membership of the right-hand side in the set of final states. This can be done in time $\mathcal{O}(\hat{r}^2 m)$.

Furthermore, we observe that $|Q^\bullet| \in \mathcal{O}(n + \hat{r}m) = \mathcal{O}(\hat{r}m)$ and $|\Delta^\bullet| \in \mathcal{O}(\hat{r}m)$. From the Paige-Tarjan algorithm [13], we know that we can compute $\cong_I$ in time $\mathcal{O}(|\Delta^\bullet| \log |Q^\bullet|)$. Therefore, the time complexity of computing $\cong_I$ amounts to $\mathcal{O}(\hat{r}m \log (\hat{r}m)) \subseteq \mathcal{O}(\hat{r}m \log \left(\hat{r}n^{\hat{r}}p\right)) = \mathcal{O}(\hat{r}^2\, m \log n + \hat{r}m \log p)$, which is also the final complexity as it covers the complexity of computing $A^\bullet$ and I. This means that, for a given Σ, we have time complexity $\mathcal{O}(m \log n)$.

5 Composed Bisimulation Equivalence

Consider a tree automaton $A = (Q, \Sigma, \Delta, F)$. We will reduce A with respect to an equivalence relation $\overset{\circ}{\simeq}$, which we propose below and which we call a *composed bisimulation equivalence*. Like downward bisimulation equivalence, composed bisimulation equivalence preserves the language of tree automata, but it may be much coarser than downward bisimulation

equivalence (note that upward bisimulation equivalence does not preserve the language of tree automata).

For a state $r \in Q$ and a set $B \subseteq Q$ of states, we write $r \simeq B$ to denote that there is a state $q \in B$ such that $q \simeq r$. We define $r \overset{\bullet}{\simeq} B$ and $r \overset{\circ}{\simeq} B$ analogously. We define $\overset{\circ}{\simeq}$ to be an equivalence relation such that $\simeq \subseteq \overset{\circ}{\simeq} \subseteq \left(\simeq * \overset{\bullet}{\simeq}\right)$. Here, $*$ denotes the composition of the two relations.

To compute $\overset{\circ}{\simeq}$, the two relations $\overset{\bullet}{\simeq}$ and $\simeq$ are composed, and all links in the relation violating transitivity are removed, while all elements from $\simeq$ are maintained. In such a manner, we obtain a new relation R. We define $\overset{\circ}{\simeq}$ to be $R \cap R^{-1}$. Notice that depending on how the transitive fragment is computed, there may be several relations satisfying the condition of $\overset{\circ}{\simeq}$.

Language Preservation. Consider a tree automaton $A = (Q, \Sigma, \Delta, F)$ and an equivalence relation $\equiv$ on Q. The *abstract tree automaton* derived from A and $\equiv$ is $A/\equiv = (Q/\equiv, \Sigma, \Delta/\equiv, F/\equiv)$ where:

- $Q/\equiv$ is the set of blocks in $\equiv$. In other words, we collapse all states which belong to the same block into one abstract state.
- $(B_1, \ldots, B_n) \xrightarrow{f} B$ iff $(q_1, \ldots, q_n) \xrightarrow{f} q$ for some $q_1 \in B_1, \ldots, q_n \in B_n, q \in B$. This is, there is a transition in the abstract automaton iff there is a transition between states in the corresponding blocks.
- $F/\equiv$ contains a block B iff $B \cap F \neq \emptyset$. Intuitively, a block is accepting if it contains a state which is accepting.

We will now consider the abstract automaton $A/\overset{\circ}{\simeq}$ where the states of A are collapsed according to $\overset{\circ}{\simeq}$. We will relate the languages of A and $A/\overset{\circ}{\simeq}$ by upcoming Theorem 3. To do that, we will first prove a series of lemmas where we utilize the notion of a *context*.

Intuitively, a context is a tree with "holes" instead of leaves. Formally, we consider a special symbol $\bigcirc \notin \Sigma$ with rank 0. A *context* over Σ is a tree c over $\Sigma \cup \{\bigcirc\}$ such that for all leaves $p \in c$, we have $c(p) = \bigcirc$. For a context c with leaves $p_1, \ldots, p_n$ and trees $t_1, \ldots, t_n$, we define $c[t_1, \ldots, t_n]$ to be the tree t, where

- $dom(t) = dom(c) \bigcup \{p_1 \cdot p' \,|\, p' \in dom(t_i)\} \bigcup \cdots \bigcup \{p_n \cdot p' \,|\, p' \in dom(t_n)\}$,
- for each $p = p_i \cdot p'$, we have $t(p) = t_i(p')$, and
- for each $p \in dom(c) \setminus \{p_1, \ldots, p_n\}$, we have $t(p) = c(p)$.

In other words, $c[t_1, \ldots, t_n]$ is the result of appending the trees $t_1, \ldots, t_k$ to the holes of c. We extend the notion of runs to contexts. Let c be

a context with leaves $p_1, \ldots, p_n$. A *run* π of A on c from $(q_1, \ldots, q_n)$ is defined in a similar manner to a run on a tree except that for a leaf p_i, we have $\pi(p_i) = q_i$, $1 \leq i \leq n$. In other words, each leaf labelled with $\bigcirc$ is annotated by one q_i. We use $c\,[q_1, \ldots, q_n] \overset{\pi}{\Longrightarrow} q$ to denote that π is a run of A on c from $(q_1, \ldots, q_n)$ such that $\pi(\epsilon) = q$. The notation $c\,[q_1, \ldots, q_n] \Longrightarrow q$ is explained in a similar manner to runs on trees.

The first of the series of lemmas leading to Theorem 3 associates runs on contexts with the upward bisimulation equivalence $\overset{\bullet}{\simeq}$.

Lemma 3. *If $c[q_1, q_2, \ldots, q_n] \Longrightarrow q$ and $q_i \overset{\bullet}{\simeq} r_i$ for some $1 \leq i \leq n$, then there are states $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_n, r$ such that $q_j \simeq r_j$ for each $j : 1 \leq j \neq i \leq n$, $q \overset{\bullet}{\simeq} r$, and $c[r_1, \ldots, r_n] \Longrightarrow r$.*

Proof. To simplify the notation, we assume (without loss of generality) that $i = 1$. We use induction on the structure of c. The base case is trivial since the context c consists of a single hole. For the induction step, we assume that c is not only a single hole. Suppose that $c[q_1, q_2, \ldots, q_n] \overset{\pi}{\Longrightarrow} q$ for some run π and that $q_1 \overset{\bullet}{\simeq} r_1$. Let $p_1, \ldots, p_j$ be the left-most leaves of c with a common parent. Let p be the parent of $p_1, \ldots, p_j$. Notice that $q_1 = \pi(p_1), \ldots, q_j = \pi(p_j)$. Let $q' = \pi(p)$ and let c' be the context c with the leaves $p_1, \ldots, p_j$ deleted.

In other words, $dom(c') = dom(c) \setminus \{p_1, \ldots, p_j\}$, $c'(p') = c(p')$ if $p' \in dom(c') \setminus \{p\}$, and $c'(p) = \bigcirc$. Observe that $c'[q', q_{j+1}, \ldots, q_n] \Longrightarrow q$ and also that $f(q_1, q_2, \ldots, q_j) \longrightarrow q'$ for some f. By definition of the upward bisimulation equivalence and the premise $q_1 \overset{\bullet}{\simeq} r_1$, it follows that there are $r_2, \ldots, r_j, r'$ such that $q_2 \simeq r_2, \ldots, q_j \simeq r_j, q' \overset{\bullet}{\simeq} r'$, and $f(r_1, r_2, \ldots, r_j) \longrightarrow r'$. Since c' is smaller than c, we can apply the induction hypothesis and conclude that there are $r_{j+1}, \ldots, r_n, r$ such that $q_{j+1} \simeq r_{j+1}, \ldots, q_n \simeq r_n, q \overset{\bullet}{\simeq} r$, and $c'[r', r_{j+1}, \ldots, r_n] \Longrightarrow r$. The claim follows immediately. □

With Lemma 3 in hand, we can relate runs of the abstract automaton with runs of the original automaton. Each run in $A/\overset{\circ}{\simeq}$ can be simulated by a run in A which starts from states that are equivalent wrt the downward bisimulation equivalence and ends up in a state that is equivalent wrt the upward bisimulation equivalence.

Lemma 4. *For blocks $B_1, \ldots, B_n, B \in Q/\overset{\circ}{\simeq}$ and a context c, wheneaver $c[B_1, \ldots, B_n] \Longrightarrow B$, then there exist states $r_1, \ldots, r_n, r \in Q$ with $r_1 \simeq B_1, \ldots, r_n \simeq B_n, r \overset{\bullet}{\simeq} B$, and $c[r_1, \ldots, r_n] \Longrightarrow r$. Moreover, if $B \in F/\overset{\circ}{\simeq}$, then also $r \in F$.*

Proof. The claim is shown by induction on the structure of c. In the base case, the context c consists of a single hole. We choose any $q \in B \cap F$ if $B \cap F \neq \emptyset$, and any $q \in B$ otherwise. The claim holds obviously by reflexivity of $\simeq$ and $\overset{\bullet}{\simeq}$.

For the induction step, we assume that c is not only a single hole. Suppose that $c[B_1, \ldots, B_n] \overset{\pi}{\Longrightarrow} B$ for some run π. Let $p_1, \ldots, p_j$ be the left-most leaves of c with a common parent. Let p be the parent of $p_1, \ldots, p_j$. Notice that $B_1 = \pi(p_1), \ldots, B_j = \pi(p_j)$. Let $B' = \pi(p)$ and let c' be the context c with the leaves $p_1, \ldots, p_j$ deleted. In other words, $dom(c') = dom(c) \setminus \{p_1, \ldots, p_j\}$, $c'(p') = c(p')$ provided $p' \in dom(c') \setminus \{p\}$, and $c'(p) = \bigcirc$.

Observe that $c'[B', B_{j+1}, \ldots, B_n] \Longrightarrow B$. Since c' is smaller than c, we can apply the induction hypothesis and conclude that there are $v, q'_{j+1}, \ldots, q'_n, q'$ such that $v \simeq B', q'_{j+1} \simeq B_{j+1}, \ldots, q'_n \simeq B_n, q' \overset{\bullet}{\simeq} B$, $c'[v, q'_{j+1}, \ldots, q'_n] \Longrightarrow q'$, and if $B \cap F \neq \emptyset$, then $q' \in F$. It follows that there are $u \in B', q_{j+1} \in B_{j+1}, \ldots, q_n \in B_n, q \in B$ with $u \simeq v, q_{j+1} \simeq q'_{j+1}, \ldots, q_n \simeq q'_n$, and $q \overset{\bullet}{\simeq} q'$. By definition of $A/\overset{\circ}{\simeq}$, there are states $q_1 \in B_1, \ldots, q_j \in B_j$, and $z \in B'$ such that $(q_1, \ldots, q_j) \xrightarrow{f} z$ for some f.

Since $\simeq \subseteq \overset{\circ}{\simeq}$ and $u \simeq v$, we get $u \overset{\circ}{\simeq} v$. Since $u, z \in B'$, it follows that $z \overset{\circ}{\simeq} u$. From transitivity of $\overset{\circ}{\simeq}$, we get $z \overset{\circ}{\simeq} v$. From the definition of $\overset{\circ}{\simeq}$, there is a state w such that $z \simeq w$ and $w \overset{\bullet}{\simeq} v$. By the definition of the downward bisimulation equivalence and premises $z \simeq w$ and $(q_1, \ldots, q_j) \xrightarrow{f} z$, there are states $r_1, \ldots, r_j$ with $q_1 \simeq r_1, \ldots, q_j \simeq r_j$, and $(r_1, \ldots, r_j) \xrightarrow{f} w$. By Lemma 3 and premises $v \overset{\bullet}{\simeq} w$ and $c'[v, q'_{j+1}, \ldots, q'_n] \Longrightarrow q'$, there are states $r_{j+1}, \ldots, r_n$, and r with $q'_{j+1} \simeq r_{j+1}, \ldots, q'_n, \simeq r_n, q' \overset{\bullet}{\simeq} r$, and $c'[w, r_{j+1}, \ldots, r_n] \Longrightarrow r$. Finally, by transitivity of $\simeq$ and $\overset{\bullet}{\simeq}$, $q_{j+1} \simeq r_{j+1}, \ldots, q_n \simeq r_n$, $q \overset{\bullet}{\simeq} r$. Moreover, by definition of $\overset{\bullet}{\simeq}$ and the fact that $q' \in F$ if $B \cap F \neq \emptyset$, we get that $r \in F$ if $B \in F/\overset{\circ}{\simeq}$. The claim thus holds. □

The next lemma already states that the language of the abstract automaton is a subset of the language of the original automaton.

Lemma 5. *If $t \Longrightarrow B$, then $t \Longrightarrow w$ for some w with $B \overset{\bullet}{\simeq} w$. Moreover, if $B \in F/\overset{\circ}{\simeq}$, then also $w \in F$.*

Proof. Suppose that $t \overset{\pi}{\Longrightarrow} B$ for some π. Let $p_1, \ldots, p_n$ be the leafs of t, and let $\pi(p_i) = B_i$ for each $i : 1 \leq i \leq n$. Let c be the context we get from

t by deleting the leaves $p_1, \ldots, p_n$. Observe that $c[B_1, \ldots, B_n] \stackrel{\pi}{\Longrightarrow} B$. It follows from Lemma 4 that there exist states $r_1, \ldots, r_n, r \in Q$ and $q_1 \in B_1, \ldots, q_n \in B_n, q \in B$ such that $q_1 \simeq r_1, \ldots, q_n \simeq r_n, q \stackrel{\bullet}{\simeq} r$, $c[r_1, \ldots, r_n] \Longrightarrow r$, and if $B \cap F \neq \emptyset$, then $r \in F$. By definition of $A/\stackrel{\circ}{\simeq}$, it follows that there are $q'_1 \in B_1, \ldots, q'_n \in B_n$ and $f_1, \ldots, f_n$ such that $\stackrel{f_i}{\longrightarrow} q'_i$ for each i such that $1 \leq i \leq n$.

We show by induction on i that for each i such that $1 \leq i \leq n$ there are states $u^i_1, \ldots, u^i_i, v^i_{i+1}, \ldots, v^i_n, w^i$ such that $q'_1 \simeq u^i_1, \ldots, q'_i \simeq u^i_i, r_{i+1} \simeq v^i_{i+1}, \ldots, r_n \simeq v^i_n, r \simeq w^i$, and $c[u^i_1, \ldots, u^i_i, v^i_{i+1}, \ldots, v^i_n] \Longrightarrow w^i$. The base case where $i = 0$ is trivial. We consider the induction step. Since $\simeq \subseteq \stackrel{\circ}{\simeq}$ and $v^i_{i+1} \simeq r_{i+1}$, we get $v^i_{i+1} \stackrel{\circ}{\simeq} r_{i+1}$. Since $r_{i+1} \simeq q_{i+1}$, we get $r_{i+1} \stackrel{\circ}{\simeq} q_{i+1}$. Since $q_{i+1}, q'_{i+1} \in B_{i+1}$, we have that $q_{i+1} \stackrel{\circ}{\simeq} q'_{i+1}$. By transitivity of $\stackrel{\circ}{\simeq}$, it follows that $v^i_{i+1} \stackrel{\circ}{\simeq} q'_{i+1}$. By the definition of $\stackrel{\circ}{\simeq}$, there is z_{i+1} such that $q'_{i+1} \simeq z_{i+1}$ and $v^i_{i+1} \stackrel{\bullet}{\simeq} z_{i+1}$. By Lemma 3, there are $z_1, \ldots, z_i, z_{i+2}, \ldots, z_n, z$ such that $u^i_1 \simeq z_1, \ldots, u^i_i \simeq z_i, v^i_{i+2} \simeq z_{i+2}, \ldots, v^i_n \simeq z_n, w^i \stackrel{\bullet}{\simeq} z$, and $c[z_1, \ldots, z_n] \Longrightarrow z$. By transitivity of $\simeq$ and the premises $q'_j \simeq u^i_j$ and $u^i_j \simeq z_j$, we have $q'_j \simeq z_j$ for each $j : 1 \leq j \leq i$. By transitivity of $\simeq$ and the premises $q_j \simeq v^i_j$ and $v^i_j \simeq z_j$, we have $q_j \simeq z_j$ for each $j : i+2 \leq j \leq n$. Define $u^{i+1}_j = z_j$ for $j : 1 \leq j \leq i+1$; $v^{i+1}_j = z_j$ for $j : i+2 \leq j \leq n$; and $w^{i+1} = z$.

The induction proof above implies that $c[u^n_1, \ldots, u^n_n] \Longrightarrow w^n$. From the definition of downward bisimulation and the premises that $\stackrel{f_i}{\longrightarrow} q'_i$ and $q'_i \simeq u^n_i$, it follows that f_i can move to u^n_i for each $i : 1 \leq i \leq n$. It follows that $t \Longrightarrow w^n$ By definition of $\stackrel{\bullet}{\simeq}$ and the fact that $r \in F$ if $B \cap F \neq \emptyset$, it follows that $\forall 1 \leq i \leq n.\ w^i \in F$ provided that $B \in F/\stackrel{\circ}{\simeq}$. Thus, in the claim of the lemma, it suffices to take $w = w^n$. □

In other words, Lemma 5 says that each tree t which leads to a block B in $A/\stackrel{\circ}{\simeq}$ will also lead to a state in A which is in the block B wrt the upward bisimulation equivalence. Moreover, if t can be accepted at B in $A/\stackrel{\circ}{\simeq}$ meaning that B contains a final state of A, i.e., $B \cap F \neq \emptyset$), then it can be accepted at w in A (i.e., $w \in F$) too. This leads to the following theorem.

Theorem 3. *$L(A/\stackrel{\circ}{\simeq}) = L(A)$ for each tree automaton A.*

TA	$\simeq$		$\sim$		$\overset{\circ}{\simeq}$		$\overset{\circ}{\sim}$	
size	reduction	time	reduction	time	reduction	time	reduction	time
202	41%	0.2	41%	3.7	45%	0.3	49%	7.1
354	11%	0.3	11%	4.2	42%	0.4	52%	7.3
909	14%	0.6	52%	3.6	82%	0.8	89%	5.2
1357	9%	3.1	10%	19.8	69%	4.2	82%	36.1
1748	1%	1.4	60%	7.8	75%	3.4	99%	13.7

Table 1. Experimental results on reducing tree automata

6 Experiments and a Comparison of Various Relations on TA

We have implemented our algorithms in a prototype tool written in Java and made experiments with the tool on multiple automata obtained from the framework of *regular tree model checking* (RTMC). RTMC is the name of a family of techniques for analysing infinite-state systems in which configurations of the systems being analysed are represented by trees, sets of the configurations by TA, and transitions of the analysed systems by tree transducers. Most of the algorithms in RTMC rely crucially on efficient reduction methods since the size of the generated automata often explodes, making a further computation with the automata infeasible without a reduction. In particular, the TA that we have considered arose during verification of the *Arbiter* protocol and the *Leader* election protocol [6].

Our experimental evaluation was carried out on an AMD Athlon 64 X2 2.19GHz PC with 2.0 GB RAM. We compare the size of the considered TA after reducing them using the downward bisimulation equivalence $\simeq$, a composed bisimulation equivalence $\overset{\circ}{\simeq}$, the downward simulation equivalence $\sim$, and a composed simulation equivalence $\overset{\circ}{\sim}$. Definitions and algorithms for computing downward simulation and composed simulation equivalences can be found in [1].

It is well known that simulations usually (though not necessarily) give a better reduction but they are harder to compute than bisimulations. Indeed, in [4], we show that $\simeq \subseteq \sim \subseteq \overset{\circ}{\sim}$, but $\overset{\circ}{\simeq}$ and $\overset{\circ}{\sim}$ as well as $\overset{\circ}{\simeq}$ and $\sim$ are incomparable, i.e., for each of the two pairs, there exists a TA for which the relations are incomparable.

In Table 1, we show the computation time (in seconds) and the reduction (in percent) obtained on the chosen automata when using $\overset{\circ}{\simeq}$, $\overset{\circ}{\sim}$, $\simeq$, and $\sim$. As can be seen from the results, composed simulation gives the

best reduction in all cases, but, on the other hand, it has a much higher computation time than all the other relations. Composed bisimulation gives a better reduction than both downward simulation and downward bisimulation. The time for computing composed simulation is lower than all simulation relations.

We have further performed a preliminary comparison of our relations by using the backward and forward bisimulations presented in [11] where it is also argued that using these relations in succession in one of the two possible orders (i.e., reducing a given TA first using the backward bisimulation and then the result by the forward bisimulation or vice versa) gives a better result than using just one of them. We show that using each of these two-step reductions is in theory incomparable with our composed bisimulation equivalences.

For each of the three reduction techniques, we now present an automaton such that reducing it by the particular reduction technique gives better result (i.e., the resulting automaton has less states) than reducing it by the other two techniques. The automaton $A_1 = (Q_1, \Sigma_1, \Delta_1, Q_1)$ can be reduced most when using composed bisimulation, in the case of the atomaton $A_2 = (Q_2, \Sigma_2, \Delta_2, Q_2)$, using forward bisimulation reduction followed by forward (downward) bisimulation reduction is the best, and the automaton $A_3 = (Q_3, \Sigma_3, \Delta_3, Q_3)$ can be reduced most using forward (downward) bisimulation reduction followed by forward bisimulation reuction. The automata, that were obtained with help of a random automata generator, look as follows:

$Q_1 = \{q_x, q_0, q_1, q_2, q_3, q_4, r_x, r_0, r_1, r_2, r_3, r_4\}$, $(\Sigma_1)_1 = \{x, a, b, c, d\}$,

Δ_3 :

$$
\begin{array}{llllll}
q_x \xrightarrow{x} q_x, & q_0 \xrightarrow{d} q_1, & q_2 \xrightarrow{d} q_1, & r_x \xrightarrow{x} r_x, & r_1 \xrightarrow{d} r_0, & r_1 \xrightarrow{d} r_2, \\
q_x \xrightarrow{a} q_4, & q_0 \xrightarrow{d} q_0, & q_2 \xrightarrow{d} q_0, & r_4 \xrightarrow{a} r_x, & r_0 \xrightarrow{d} r_0, & r_0 \xrightarrow{d} r_2, \\
q_x \xrightarrow{a} q_3, & q_1 \xrightarrow{d} q_4, & q_3 \xrightarrow{c} q_4, & r_3 \xrightarrow{a} r_x, & r_4 \xrightarrow{d} r_1, & r_4 \xrightarrow{c} r_3, \\
q_x \xrightarrow{b} q_3, & q_2 \xrightarrow{c} q_3, & q_3 \xrightarrow{c} q_2, & r_3 \xrightarrow{b} r_x, & r_3 \xrightarrow{c} r_2, & r_2 \xrightarrow{c} r_3, \\
q_0 \xrightarrow{c} q_0, & q_2 \xrightarrow{c} q_1, & q_3 \xrightarrow{d} q_2, & r_0 \xrightarrow{c} r_0, & r_1 \xrightarrow{c} r_2, & r_2 \xrightarrow{d} r_3, \\
q_0 \xrightarrow{c} q_3, & q_2 \xrightarrow{c} q_0, & q_3 \xrightarrow{d} q_1, & r_3 \xrightarrow{c} r_0, & r_0 \xrightarrow{c} r_2, & r_1 \xrightarrow{d} r_3, \\
q_0 \xrightarrow{c} q_1, & q_2 \xrightarrow{d} q_3, & q_3 \xrightarrow{d} q_0, & r_1 \xrightarrow{c} r_0, & r_3 \xrightarrow{d} r_2, & r_0 \xrightarrow{d} r_3, \\
q_0 \xrightarrow{d} q_2, & q_2 \xrightarrow{d} q_2, & q_4 \xrightarrow{d} q_4, & r_2 \xrightarrow{d} r_0, & r_2 \xrightarrow{d} r_2, & r_4 \xrightarrow{d} r_4
\end{array}
$$

$Q_2 = Q_3 = \{q_0, q_1, q_2\}$, $(\Sigma_2)_0 = (\Sigma_3)_0 = \{a, b\}$, $(\Sigma_2)_1 = (\Sigma_3)_1 = \{c, d\}$,

Δ_1 :

$$\begin{array}{lll}
\xrightarrow{a} q_2, & q_2 \xrightarrow{c} q_1, & q_2 \xrightarrow{d} q_1, \\
\xrightarrow{a} q_1, & q_1 \xrightarrow{c} q_2, & q_2 \xrightarrow{d} q_2, \\
\xrightarrow{b} q_1, & q_1 \xrightarrow{c} q_1, & q_0 \xrightarrow{d} q_0, \\
\xrightarrow{b} q_2, & q_1 \xrightarrow{c} q_0, & q_0 \xrightarrow{d} q_1 \\
 & q_0 \xrightarrow{c} q_1, &
\end{array}$$

Δ_2 :

$$\begin{array}{lll}
\xrightarrow{a} q_1, & q_1 \xrightarrow{c} q_0, & q_1 \xrightarrow{d} q_2, \\
\xrightarrow{a} q_2, & q_0 \xrightarrow{c} q_0, & q_1 \xrightarrow{d} q_1, \\
\xrightarrow{b} q_1, & q_2 \xrightarrow{c} q_0, & q_1 \xrightarrow{d} q_0, \\
\xrightarrow{b} q_2, & & q_0 \xrightarrow{d} q_0.
\end{array}$$

On the other hand, in all our test cases, backward bisimulation followed by forward bisimulation behaved in a very similar way to composed bisimulation—a more thorough experimental comparison is to be done in the future.

Let us note that applying the forward and backward bisimulations in succession has the advantage that the second relation is applied on a smaller input automaton. On the other hand, our notion of composition has the advantage of being applicable also for composing simulations (while re-using a lot of the needed algorithms and data structures) and even for combining simulations and bisimulations as we discuss in our follow-up work [4], which thus gives the user a wider variety of easily implementable reductions for TA.

7 Conclusions and Future Work

We have presented a new notion of equivalence, called *composed bisimulation equivalence*, for reducing TA while preserving their language. Composed bisimulation equivalence is defined in terms of a composition of downward bisimulation equivalence proposed earlier in the literature and upward bisimulation equivalence proposed in this paper. We have discussed theoretical as well as experimental evidence that composed bisimulation equivalence offers a new compromise between reduction capabilities and computational demands and thus offers designers of tools based on TA a finer choice of the technique to be used for reducing TA. Moreover, the notion of composed bisimulation equivalence and also the associated algorithms share some common kernel with those used for working with various simulation equivalences in [1], which allows for an easy implementation and combinations of all these approaches. Indeed, the possibility of composing not only upward and downward bisimulations or upward and downward simulations, but defining a parametric (bi-)simulation framework allowing one to mix simulations and bisimulations and thus further

tune the desired degree of the trade-off between reductions and their costs, is considered in the recent follow-up work [4].

There are several interesting directions for future work. First, one can consider extending the results to the domain of symbolically encoded tree automata like in the MONA tree automata library [12], allowing one to deal with significantly larger automata. Next, one can investigate the possibility of repeated (nested) compositions of various equivalences. Finally, it can be interesting to extend the algorithms presented in this paper to work for other kinds of tree automata such as guided tree automata, weighted tree automata, or unranked tree automata.

Acknowledgement. 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 MEB 020840, and the Czech Ministry of Education by the project MSM 0021630528.

References

1. P. Abdulla, A. Bouajjani, L. Holik, L. Kaati, and T. Vojnar. Computing Simulations over Tree Automata: Efficient Techniques for Reducing Tree Automata. In *Proc. of TACAS'08*, LNCS. Springer, 2008.
2. P. Abdulla, J. Deneux, L. Kaati, and M. Nilsson. Minimization of Non-deterministic Automata with Large Alphabets. In *Implementation and Application of Automata*, LNCS, pages 31–42. Springer, 2006.
3. P. Abdulla, J. Högberg, and L. Kaati. Bisimulation Minimization of Tree Automata. In *Proc. of CIAA'06*, volume 4094 of *LNCS*, pages 173–185. Springer, 2006.
4. P. Abdulla, L. Holík, L. Kaati, and T. Vojnar. A Uniform (Bi-)Simulation-Based Framework for Reducing Tree Automata. Technical Report FIT-TR-2008-05, FIT, Brno University of Technology, Czech Republic, 2008.
5. P. Abdulla, B. Jonsson, P. Mahata, and J. d'Orso. Regular Tree Model Checking. In *Proc. of CAV'02*, volume 2404 of *LNCS*. Springer, 2002.
6. P. Abdulla, A. Legay, J. d'Orso, and A. Rezine. Tree Regular Model Checking: A Simulation-based Approach. *The Journal of Logic and Algebraic Programming*, 69(1-2):93–121, 2006.
7. A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract Regular Tree Model Checking. In *Proc. of INFINITY'05*. Published in ENTCS 149(1), 2006.
8. A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract Regular Tree Model Checking. *ENTCS*, 149:37–48, 2006.
9. A. Bouajjani and T. Touili. Extrapolating Tree Transformations. In *Proc. of CAV'02*, volume 2404 of *LNCS*. Springer, 2002.
10. M. Henzinger, T. Henzinger, and P. Kopke. Computing Simulations on Finite and Infinite Graphs. In *Proc. of FOCS'95*. IEEE, 1995.
11. J. Högberg, A. Maletti, and J. May. Backward and Forward Bisimulation Minimisation of Tree Automata. In *Proc. of CIAA'07*, volume 4094 of *LNCS*, pages 109–121. Czech Technical University in Prague, Czech Republic, 2007.

12. N. Klarlund and A. Møller. MONA Version 1.4 User Manual, 2001. BRICS, Department of Computer Science, University of Aarhus, Denmark.
13. R. Paige and R. Tarjan. Three Partition Refinement Algorithms. *SIAM Journal on Computing*, 16:973–989, 1987.
14. F. Ranzato and F. Tapparo. A New Efficient Simulation Equivalence Algorithm. In *Proc. of LICS'07*. IEEE CS, 2007.