A Uniform (Bi-)Simulation-Based Framework for Reducing Tree Automata

FIT BUT Technical Report Series

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

FIT

Technical Report No. FIT-TR-2008-005
Faculty of Information Technology, Brno University of Technology

Last modified: May 13, 2008

A Uniform (Bi-)Simulation-Based Framework for Reducing Tree Automata

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

[1] University of Uppsala, Sweden, email: {parosh,lisa.kaati}@it.uu.se
[2] FIT, Brno University of Technology, Czech Rep., email: {holik,vojnar}@fit.vutbr.cz

Abstract. In this paper, we address the problem of reducing the size of nondeterministic (bottom-up) tree automata. We propose a uniform framework that allows for combining various upward and downward bisimulation and simulation relations in order to obtain a language-preserving combined relation suitable for reducing tree automata without a need to determinise them. The framework generalises and improves several previous works and provides a broad spectrum of different relations yielding a possibility of a fine choice between the amount of reduction and the computational demands. We analyse properties of the considered relations both theoretically as well as through a series of experiments.

1 Introduction

Finite tree automata are a natural generalisation of word automata. Since trees (or terms) appear in many areas of computer science and engineering, tree automata are quite broadly applicable—including, for instance, applications in XML manipulation, natural language processing, or formal verification. In most of these applications, dealing with as small automata as possible is highly desirable. In order to reduce the size of a given tree automaton, one can always try to determinise and minimise it. However, the determinisation may lead to an exponential blow-up in the size, and even the minimal deterministic automaton might still be bigger than the original nondeterministic automaton. Moreover, even if the minimal deterministic automaton is really small, it might be impossible to compute it due to the very expensive determinisation step.

An alternative way to reducing a given (nondeterministic) tree automaton is identifying a suitable, language-preserving equivalence relation over its states and collapsing those states that are equal according to this relation. As in the case of word automata, good candidates for such relations are various bisimulations and simulation equivalences. In particular, the so-called forward and backward bisimulations and simulation equivalences are well known to be useful when reducing the size of word automata. In this paper, we deal with their tree automata extensions—the so-called *downward* and *upward bisimulations* and *simulation equivalences*.

The *downward (bi-)simulations*, which straightforwardly generalise the appropriate backward (bi-)simulations from word automata to (bottom-up) tree automata, are compatible with the language inclusion preorder. This is, if a state r downward (bi-)simulates a state q, then the language accepted by q is a subset of the language accepted by r. Therefore, these relations are a natural choice for reducing the size of tree automata.

The *upward (bi-)simulations* are not compatible with the language inclusion preorder. Instead, they are compatible with the inclusion of the so-called *context languages*, where a context of a state q arises from a tree accepted at q by replacing some of its leaves by a "hole". It can, however, be shown that when we restrict ourselves to upward (bi-)simulations compatible with the set of final states of automata, the downward and upward (bi-)simulations can be *combined* in such a way that they yield a language-compatible equivalence. In the worst case, the combined relation is as coarse as the appropriate downward (bi-)simulation equivalence, but according to our practical experiments, it usually leads to significantly better reductions of the automata.

Tree bisimulations can be computed efficiently in time $O(\hat{r}^2\ m\ log\ n)$ 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 the given tree automaton [3, 9, 1]. However, the reduction obtained by using bisimulations is often limited. Tree simulations are weaker than bisimulations and hence offer a better reduction. On the other hand, despite the recent advances in algorithms devised for computing them [2], they are significantly more expensive to compute. The time complexity of computing simulation preorders is roughly in $O(mn)$.

In this paper, we propose a *uniform framework* which allows for fine-tuning the balance between easy to compute bisimulations and coarser, and thus better reducing simulation equivalences. The framework allows one to combine any downward, language-compatible preorder (such as a downward bisimulation or simulation) with an upward bisimulation or simulation (in all possible combinations). A motivation behind the framework is to give the user a possibility to mix in various ways features of different simulations and bisimulations and thus optimise the obtained combined relation for the concrete application at hand. The proposed framework unifies and generalises various previous results [9, 2, 1].

We carefully analyse mutual relationships of the various considered relations. We establish a certain partial ordering between their reduction capabilities, but we also show that many of them have an incomparable reduction power.

Compared to the previous work, we also propose an improved way of computing the combined (bi)simulation relations. Before the composition was accomplished by randomly looking for some combined relation satisfying the needed requirements. In this paper, we prove that there always exists a *unique maximal combined relation* and we show that it can be computed by a simple algorithm in time $O(n^3)$ (or, in fact, even slightly better). The use of the maximal combined relations turns out to itself give much better results in our experiments than the previously known algorithms. Let us also note that the notion of combined (bi-)simulation relations that we propose is applicable even for word automata as for a special case of tree automata.

In order to experimentally examine the broad spectrum of relations offered by our framework, we implemented a prototype tool in which we have performed thorough experiments with tree automata from the domain of formal verification of infinite-state systems based on the so-called regular tree model checking and abstract regular tree model checking. Our experimental results confirm that we have obtained a broad range of algorithms for reducing tree automata, differing in their computation complexity and reduction capabilities.

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 [10] was presented. In [9], two different types of bisimulations—

a backward and forward bisimulation—were presented. These bisimulations turn out to be special cases of the relations arising in our framework.

Efficient algorithms for computing simulation equivalences over tree automata have then been discussed in [2] together with a proposal of combined simulation relations. In [1], the ideas from [2] were extended to work for bisimulations. In this paper, we try to combine advantages of the simulation and bisimulation approaches from [1, 2] and allow them to be mixed in a degree suitable for a given scenario. The way we use for computing the upward and downward relations that we are dealing with is inspired by the approach of [2] and [1]. We, however, provide a significantly improved way of combining these relations.

2 Preliminaries

In this section, we introduce some preliminaries on relations, trees, and tree automata.

Trees. A *ranked alphabet* Σ is a set of symbols together with a function $\# : \Sigma \to \mathbb{N}$. For $f \in \Sigma$, the value $\#(f)$ 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 a ranked 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 $\#(t(p)) = n \geq 0$, 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 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 $\#(f) = n$. 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 where $n = 0$, we speak about the so-called *leaf rules*, which we sometimes 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 $(q_1, \ldots, q_n)$ where $(q_1, \ldots, q_n) \xrightarrow{f} q$ for some f and q. We will drop the reference to A if no confusion may arise. Finally, we denote by $\hat{r}(A)$ 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 $q_i \in Q$, $1 \leq i \leq m$. We omit the reference to A if no confusion may arise.

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 $q = \pi(p)$, 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 $\pi(\varepsilon) = q$. 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 $L(q) = \{t \mid t \Longrightarrow q\}$, while the *language* of A is defined by $L(A) = \bigcup_{q \in F} L(q)$.

An *environment* is a tuple of the form $((q_1, \ldots, q_{i-1}, \square, q_{i+1}, \ldots, q_n), f, q)$ 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 $\square \notin Q$

(called a *hole* below). Like for transition rules, we write $(q_1, \ldots, \square, \ldots, q_n) \xrightarrow{f} q$ provided $((q_1, \ldots, q_{i-1}, q_i, q_{i+1}, \ldots, q_n), f, q) \in \Delta$ for some $q_i \in Q$. Sometimes, we also write the environment as $(q_1, \ldots, \square_i, \ldots, q_n) \xrightarrow{f} q$ to emphasise that the hole is at position i. We denote the set of all environments of A by $Env(A)$ and we will drop the reference to A if no confusion may arise.

Relations. 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 $\equiv$. For a preorder P, we will denote $\equiv_P$ the maximal equivalence included in P. Given a TA $A = (Q, \Sigma, \Delta, F)$, the *language inclusion preorder on Q* is the relation $LP_A = \{(q, r) \in Q \times Q \mid L(q) \subseteq L(r)\}$. We again drop the subscript A if no confusion can arise.

Abstract Tree Automata. The idea of reducing the size of an automaton is to identify suitable equivalence relations on its states, and then collapse the sets of states which form equivalence classes. Consider a TA $A = (Q, \Sigma, \Delta, F)$ 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.
- $(\mathsf{B}_1, \ldots, \mathsf{B}_n) \xrightarrow{f} \mathsf{B}$ iff $(q_1, \ldots, q_n) \xrightarrow{f} q$ for some $q_1 \in \mathsf{B}_1, \ldots, q_n \in \mathsf{B}_n, q \in \mathsf{B}$. This is, there is a transition in the abstract automaton iff there is a transition between states in the corresponding blocks in the original TA.
- $F_\equiv$ contains a block B iff $\mathsf{B} \cap F \neq \emptyset$. Intuitively, a block is accepting if it contains a state which is accepting.

3 A Parametric Notion of (Bi-)Simulations on Tree Automata

We now propose a notion of upward (bi-)simulations *parameterised* by an "*inducing relation*" that is required to be a subset of the language inclusion preorder (which is the case, e.g., for the identity or downward bisimulations or simulations). In the next section, we show that any of the resulting upward (bi-)simulations can be used to construct an equivalence relation suitable for reducing tree automata. For this, the upward (bi-)simulations parameterised by some inducing relation will further be combined with the inducing relation in a certain way—note that the inducing relation is thus used in two different ways: as a parameter of the upward (bi-)simulations and as a constituent of the combined relations. By considering various inducing relations, we obtain a wide spectrum of combined relations differing in their computational complexity and coarseness (which is usually better and never worse than that of the inducing relation).

3.1 Parametric Upward Simulation

Given a tree automaton $A = (Q, \Sigma, \Delta, F)$ and a preorder R included in LP, an *upward simulation U* induced by R is a binary relation on Q such that if qUr, then

(i) if $(q_1, \ldots, q_n) \xrightarrow{f} q'$ with $q_i = q$, $1 \leq i \leq n$, then $(r_1, \ldots, r_n) \xrightarrow{f} r'$ with $r_i = r$, $q'Ur'$, and $q_j R r_j$ for each $j : 1 \leq j \neq i \leq n$;
(ii) $q \in F \implies r \in F$.

The following lemma subsumes basic properties of upward simulations. Note that it also implies that for any $R \subseteq LP$, there is always a unique maximal upward simulation wrt. R which is a preorder.

Lemma 1. *Given a tree automaton A and a preorder $R \subseteq LP$, the set of all upward simulations wrt. R is closed under reflexive and transitive closure and under union.*

Proof. We fix a tree automaton $A = (Q,\Sigma,\Delta,F)$.

Union: Given a preorder $R \subseteq LP$ and two upward simulations U_1 and U_2 wrt. R, we want to prove that $U = U_1 \cup U_2$ is also an upward simulation wrt. R. 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 simulation, whenever $(q_1,\ldots,q_n) \xrightarrow{f} q'$ with $q_i = q$, then there is a rule $(r_1,\ldots,r_n) \xrightarrow{f} r'$ with $q'U_1r'$, $q' \in F \implies r' \in F$, and q_jRr_j for all $j : 1 \leq j \neq i \leq n$. As $U_1 \subseteq U$ gives $q'Ur'$, U fulfils the definition of upward simulation wrt. R.

Reflexive closure: It can be seen from the definition of upward simulations that the identity is an upward simulation wrt. any preorder $R \subseteq LP$. Therefore, from the closure under union, the union of identity and any upward simulation wrt. R is an upward simulation wrt. R.

Transitive closure: Let U be an upward simulation wrt. a preorder $R \subseteq LP$ and let U_T be its transitive closure. Let $q^1U_Tq^m$ and $(q^1_1,\ldots,q^1_n) \xrightarrow{f} r^1$ with $q^1 = q^1_i$. From $q^1U_Tq^m$, we have that there are states $q^1,\ldots,q^m$ such that $q^1Uq^2U\ldots Uq^m$. Therefore, there are also rules $(q^1_1,\ldots,q^1_n) \xrightarrow{f} r^1,\ldots,(q^m_1,\ldots,q^m_n) \xrightarrow{f} r^m$ with $q^1_i = q^1_i,\ldots,q^m_i = q^m$, $r^1U\ldots Ur^m$, $r^1 \in F \implies \ldots \implies r^m \in F$, and $q^1_jR\ldots Rq^m_j$ for all $j : 1 \leq j \neq i \leq n$. Thus, from the definition of U_T, we have $r^1U_Tr^m$, from the transitivity of $\implies$, we have $r^1 \in F \implies r^m \in F$, and from the transitivity of R, we have $q^1_jRq^m_j$ for all $j : 1 \leq j \neq i \leq n$. We have proven that U_T fulfils the definition of an upward simulation wrt. R.

□

3.2 Parametric Upward Bisimulation

Let $A = (Q,\Sigma,\Delta,F)$ be a tree automaton and let R be a preorder included in LP. An *upward bisimulation U on Q wrt. R* is a binary relation on Q such that if qUr, then

(i) $(q_1,\ldots,q_n) \xrightarrow{f} q'$ with $q_i = q, 1 \leq i \leq n$, if and only if $(r_1,\ldots,r_n) \xrightarrow{f} r'$ with $r_i = r$, $q'Ur'$, and $q_j \equiv_R r_j$ for each $j : 1 \leq j \neq i \leq n$;
(ii) $q \in F \iff r \in F$.

As for upward simulations, it is not hard to prove the basic properties of upward bisimulations. Note that the following lemma implies that for any $R \in LP$, there is a unique maximal upward bisimulation wrt. R that is an equivalence. It is also clear that any upward bisimulation wrt. a preorder R is also an upward simulation wrt. R. This will allow us to prove the main results just for upward simulations and maintain bisimulations as a special case.

Lemma 2. *Given a tree automaton A and a preorder $R \subseteq LP$, the set of all upward bisimulations wrt. R is closed under symmetric, reflexive, and transitive closure and under union, and is also a subset of the set of all upward simulations on A wrt. R.*

Proof (Lemma 2). The closure under union, reflexivity, and transitivity can be proven analogically as in the case of upward simulations. What remains is the closure under symmetry. Let U be an upward bisimulation wrt. a preorder $R \subseteq LP$ and let $U_S = U \cup U^{-1}$ be its symmetrical closure. It is sufficient to prove that U^{-1} is an upward bisimulation wrt. R because then, from the closure under union, U_S is an upward bisimulation wrt. R too.

Let $qU^{-1}r$. Then, from rUq, we have that $(r_1,\dots,r_n) \xrightarrow{f} r'$ with $r = r_i$ if and only if $(q_1,\dots,q_n) \xrightarrow{f} q'$ with $q = q_i$, $r'Uq'$, and $r_j \equiv_R q_j$ for all $j : 1 \leq j \neq i \leq n$. As $\equiv_R$ is an equivalence and as $r'Uq'$ is equivalent to $q'U^{-1}r'$, we can rewrite this as $(q_1,\dots,q_n) \xrightarrow{f} q'$ with $q = q_i$ iff $(r_1,\dots,r_n) \xrightarrow{f} r'$ with $r = r_i$, $q'U^{-1}r'$ and $q_j \equiv_R r_j$ for all $j : 1 \leq j \neq i \leq n$. We directly see that U^{-1} matches the definition of an upward bisimulation wrt. R.

□

Let us note that the notion of an upward bisimulation parameterized by the identity corresponds to the notion of a forward bisimulation from [9].

3.3 Possible Inducing Relations: Downward (Bi-)Simulations

We now define downward simulations and bisimulations as two suitable candidates for inducing relations and we discuss their basic properties.

For a tree automaton $A = (Q, \Sigma, \Delta, F)$, a *downward simulation* D is a binary relation on Q such that if qDr and $(q_1,\dots,q_n) \xrightarrow{f} q$, then $(r_1,\dots,r_n) \xrightarrow{f} r$ with q_iDr_i for each $i : 1 \leq i \leq n$. A *downward bisimulation* D is a binary relation on Q such that if qDr, then $(q_1,\dots,q_n) \xrightarrow{f} q$ if and only if $(r_1,\dots,r_n) \xrightarrow{f} r$ with q_iDr_i for each $i : 1 \leq i \leq n$.

The following two lemmas state analogical properties of the downward (bi-)simulations as what holds for the upward (bi-)simulations.

Lemma 3. *Given a tree automaton A, the set of all downward simulations on A is closed under reflexive, and transitive closure and under union.*

Lemma 4. *Given a tree automaton A, the set of all downward bisimulations on A is closed under symmetric, reflexive, and transitive closure and under union, and is also a subset of the set of all downward simulations on A.*

Proofs of Lemmas 3 and Lemma 4 are analogical to that of Lemmas 1 and 2 (cf. [5]). The lemmas imply that for a given tree automaton, there is a unique maximal downward bisimulation which is an equivalence and a unique maximal downward simulation which is a preorder. In [2], we prove that any downward simulation (and therefore any downward bisimulation) is included in LP. We note that the notion of downward bisimulations corresponds to that of backward bisimulations from [9].

4 Combining Relations for Size Reduction

Upward simulation equivalence and upward bisimulation alone cannot be used for reducing tree automata as they do not preserve their language. To circumvent this problem, we have to take into account the inducing relation and *combine* it with the induced upward (bi)simulation—as we have already mentioned in the previous section, the induced relation is thus used in two different ways.

In [2, 1], we have shown how to combine a downward simulation with an upward simulation and how to combine a downward bisimulation with an upward bisimulation. Here, we propose a way how to combine any preorder that is a subset of the language inclusion preorder (including both downward simulations and bisimulations) with either an upward simulation or bisimulation (in any combination).

Moreover, in the previous works, we were randomly computing one relation out of the set of possible language-preserving combined relations. Here, we first prove that there is always a *unique maximal combined preorder* for a given upward (bi-)simulation and its inducing preorder. In Section 6, we then provide a simple algorithm for computing this maximal preorder. From a practical point of view, using the maximal preorder instead of a random one has in some cases a great impact on the size of the reduced automaton as witnessed by our practical experiments.

To show the existence of the unique maximal combined preorder, we first prove that given two preorders over a set Q, say H and S, there is always a unique maximal preorder among the preorders included in $H \circ S$ that include H.

We will make use of the following notion of a set of *H-extensible edges*. Intuitively, an edge (i.e., a couple) $e \in H \circ S$ is H-extensible iff after adding it to the preorder H, there is still a possibility to supplement the obtained relation $H \cup \{e\}$ by edges from $H \circ S$ to ensure transitivity and get this way a preorder again.

H-extensible edges. Let H and S be preorders over a set Q and let $P = H \circ S$. An edge xPy is *H-extensible* if for any yHz resp. zHx there is also the edge xPz resp. zPy. We will denote by $H \oplus S$ the set of all H-extensible edges.

Lemma 5. *The set $H \oplus S$ of all H-extensible edges is a preorder and it is the maximal preorder included in $H \circ S$ that includes H.*

Proof. Let $E = H \oplus S$. Keep in mind that $E, H, S \subseteq P$ and that H and S are reflexive and transitive. We first prove some auxiliary facts for any $x, y, z \in Q$ allowing us to derive the existence of certain edges in the relations that we are dealing with:

I. $xPy \implies xHwSy$ for some $w \in Q$, which follows directly from the definition of P.
II. $xHyPz \implies xPz$. From yPz and (I), we have $yHwSz$ for some $w \in Q$. From $xHyHw$, we have xHw. From $xHwSz$ and from the definition of P, we have xPz.
III. $xEyHz \implies xPz$, which follows directly from the definition of E.
IV. $xPySz \implies xPz$. From xPy and (I), we have $xHwSy$ for some $w \in Q$. From $wSySz$, we have wSz. From $xHwSz$ and (II), we have xPz.
V. $xEyPz \implies xPz$. From yPz and (I), we have $yHwSz$ for some $w \in Q$. From $xEyHw$ and (III), we have xPw, which together with (I) gives $xHvSw$ for some $v \in Q$. From $vSwSz$, we have vSz which together with xHv and (II) gives xPz.

Now we can prove by contradiction the claim of the lemma. Suppose that there exists x,y,z such that $xEyEz$ but not xEz. Recall that $E \subseteq P$. From (I), we have $xHwSyHvSz$ for some $v,w \in Q$. From $xEyHv$ and (III), we have xPv. From $xPvSz$ and (IV), we have xPz. If the edge xPz is H-extensible, then we are done.

Next, suppose that xPz is not H-extensible. Then there is a $q \in Q$ such that either (i) $qHxPz$ but not qPz, or (ii) $xPzHq$ but not xPq. In the case (i), from $qHxPv$ and (II), we get qPv. $qPvSz$ and (IV) gives qPz, which is a contradiction. In the case (ii), from $yEzHq$ and (III), we get yPq. Then $xEyPq$ and (V) gives xPq, which is again a contradiction.

We have proven that the set E of all H-extensible edges is a preorder. Moreover, it can be easily seen that any edge $e \in P \setminus E$ cannot be contained in any preorder R such that $H \subseteq R \subseteq H \circ S$ as no such relation R with $e \in R$ can be transitive. This implies that E is also the maximal preorder included in P that includes H. □

Lemma 5 allows us to define the combined preorder and equivalence as follows.

Combined relations. Consider a tree automaton $A = (Q,\Sigma,\Delta,F)$, a preorder R included in LP, and an upward simulation U wrt R. The relation $C = R \oplus (U^{-1})$ is a *combined preorder* and $\equiv_C$ is then a *combined equivalence*. Correctness of using the combined equivalence for reducing the size of tree automata is stated in the following theorem.

Theorem 1. *$L(A_{\equiv_C}) = L(A)$ for any tree automaton A and each combined preorder C.*

4.1 Proof of Theorem 1

We fix a tree automaton A, a preorder $R \subseteq LP$ and an upward simulation U wrt. R. Let $C = R \oplus (U^{-1})$.

Lemma 6. *If $c[q_1,q_2,\ldots,q_n] \Longrightarrow q$ and q_iUr_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_jRr_j for each j such that $1 \leq j \neq i \leq n$, qUr, 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_1Ur_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 $p' \in dom(c') \setminus \{p,p_1,\ldots,p_j\}$, and $c'(p) = \bigcirc$. Observe that $c'[q',q_{j+1},\ldots,q_n] \Longrightarrow q$ and that $(q_1,\ldots,q_j) \xrightarrow{f} q'$ for some f. By definition of the upward simulation and the premise q_1Ur_1, it follows that there are $r_2,\ldots,r_n,r'$ such that $q_2Rr_2 \in,\ldots,q_jRr_j,q'Ur'$, and $(r_1,\ldots,r_j) \xrightarrow{f} r'$. 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 $q_{j+1}Rr_{j+1},\ldots,q_nRr_n,qUr$, and $c'[r',r_{j+1},\ldots,r_n] \Longrightarrow r$. The claim follows immediately. □

For a state $r \in Q$ and a set $\mathsf{B} \subseteq Q$ of states, we write $(\mathsf{B},r) \in R$ to denote that there is a $q \in \mathsf{B}$ with $(q,r) \in R$. We define $(\mathsf{B},r) \in U$ analogously.

Lemma 7. *For blocks* $\mathsf{B}_1,\ldots,\mathsf{B}_n,\mathsf{B} \in Q_{\equiv_C}$ *and a context* c*, if* $c[\mathsf{B}_1,\ldots,\mathsf{B}_n] \Longrightarrow \mathsf{B}$*, then there exist states* $r_1,\ldots,r_n,r \in Q$ *with* $\mathsf{B}_1Rr_1,\ldots,\mathsf{B}_nRr_n,\mathsf{B}Ur$*, and* $c[r_1,\ldots,r_n] \Longrightarrow r$*.*

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 $q \in B\cap F$ provided that $B\cap F \neq \emptyset$, and any $q \in B$ otherwise. The claim holds obviously by reflexivity of R and U.

For the induction step, we assume that c is not only a single hole. Suppose that $c[\mathsf{B}_1,\ldots,\mathsf{B}_n] \overset{\pi}{\Longrightarrow} \mathsf{B}$ 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 $\mathsf{B}_1 = \pi(p_1),\ldots,\mathsf{B}_j = \pi(p_j)$. Let $\mathsf{B}' = \pi(p)$ 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 $p' \in dom(c')\setminus\{p,p_1,\ldots,p_j\}$, and $c'(p) = \bigcirc$. Observe that $c'[\mathsf{B}',\mathsf{B}_{j+1},\ldots,\mathsf{B}_n] \Longrightarrow \mathsf{B}$. 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 $\mathsf{B}'Rv,\mathsf{B}_{j+1}Rq'_{j+1},\ldots,\mathsf{B}_nRq'_n,\mathsf{B}Uq'$, $c'[v,q'_{j+1},\ldots,q'_n] \Longrightarrow q'$. It follows that there are $u \in \mathsf{B}'$, $q_{j+1} \in \mathsf{B}_{j+1},\ldots,q_n \in \mathsf{B}_n, q \in \mathsf{B}$ such that uRv, qUq' and $q_{j+1}Rq'_{j+1},\ldots,q_nRq'_n$. By definition of $A_{\equiv_C}$, there are states $q_1 \in \mathsf{B}_1,\ldots,q_j \in \mathsf{B}_j$, and $z \in \mathsf{B}'$ such that $(q_1,\ldots,q_j) \xrightarrow{f} z$ for some f. Since $R \subseteq C$ and uRv, we get uCv. Since $u,z \in \mathsf{B}'$, it follows that $u \equiv_C z$ and hence zCu. From transitivity of C, we get zCv. From the definition of C, there is a state w such that zRw and vUw. By the definition of language inclusion preorder and premises zRw and $(q_1,\ldots,q_j) \xrightarrow{f} z$, there are states $r_1,\ldots,r_j$ with $q_1Rr_1,\ldots,q_jRr_j$, and $(r_1,\ldots,r_j) \xrightarrow{f} w$. By Lemma 6 and premises vUw and $c'[v,q'_{j+1},\ldots,q'_n] \Longrightarrow q'$, there are states $r_{j+1},\ldots,r_n$, and r with $q'_{j+1}Rr_{j+1},\ldots,q'_nRr_n,q'Ur$, and $c'[w,r_{j+1},\ldots,r_n] \Longrightarrow r$. Finally, by transitivity of R and U, we get $q_{j+1}Rr_{j+1},\ldots,q_nRr_n$, qUr. The claim thus holds. □

Lemma 8. *If* $t \Longrightarrow \mathsf{B}$*, then* $t \Longrightarrow w$ *for some* w *with* $\mathsf{B}Uw$*. Moreover, if* $\mathsf{B} \in F_{\equiv_C}$*, then also* $w \in F$*.*

Proof. Suppose that $t \overset{\pi}{\Longrightarrow} \mathsf{B}$ for some π. Let $p_1,\ldots,p_n$ be the leafs of t, and let $\pi(p_i) = \mathsf{B}_i$ for each $i : 1 \le i \le n$. Let c be the context we get from t by deleting the leaves $p_1,\ldots,p_n$. Observe that $c[\mathsf{B}_1,\ldots,\mathsf{B}_n] \overset{\pi}{\Longrightarrow} \mathsf{B}$. It follows from Lemma 7 that there exist states $r_1,\ldots,r_n,r \in Q$ and $q_1 \in \mathsf{B}_1,\ldots,q_n \in \mathsf{B}_n,q \in \mathsf{B}$ such that $q_1Rr_1,\ldots,q_nRr_n,qUr$, $c[r_1,\ldots,r_n] \Longrightarrow r$, and if $B\cap F \neq \emptyset$, then $r \in F$. By definition of $A_{\equiv_C}$, it follows that there are $q'_1 \in \mathsf{B}_1,\ldots,q'_n \in \mathsf{B}_n$ and $f_1,\ldots,f_n$ such that $\xrightarrow{f_i} q'_i$ for each i such that $1 \le i \le n$. We show by induction on i that for each i such that $1 \le i \le n$ there are states $u^i_1,\ldots,u^i_i,v^i_{i+1},\ldots,v^i_n,w^i$ such that $q'_1Ru^i_1,\ldots,q'_iRu^i_i,q_{i+1}Rv^i_{i+1},\ldots,q_nRv^i_n,rUw^i$, 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 $R \subseteq C$ and $q_{i+1}Rv_{i+1}$, we get $q_{i+1}Cv_{i+1}$. Since $q_{i+1},q'_{i+1} \in \mathsf{B}_{i+1}$, we have that $q'_{i+1} \equiv_C q_{i+1}$ and hence $q'_{i+1}Cq_{i+1}$. By transitivity of C, it follows that $q'_{i+1}Cv_{i+1}$. By the definition of C, there is z_{i+1} such that $q'_{i+1}Rz_{i+1}$ and $v_{i+1}Uz_{i+1}$. By Lemma 6, there are $z_1,\ldots,z_i,z_{i+2},\ldots,z_n,z$ with $u^i_1Rz_1,\ldots,u^i_iRz_i$, $v^i_{i+2}Rz_{i+2},\ldots,v^i_nRz_n$, w^iUz, and $c[z_1,\ldots,z_n] \Longrightarrow z$. By transitivity of R and the premises $q'_jRu^i_j$ and $u^i_jRz_j$, we have q'_jRz_j for each $j : 1 \le j \le i$. By transitivity of R and the premises $q_jRv^i_j$ and $v^i_jRz_j$, we have q_jRz_j for each $j : i+2 \le j \le n$. Define $u^{i+1}_j = z_j$ for $j : 1 \le j \le i+1$; $v^{i+1}_j = z_j$ for $j : i+2 \le j \le n$; and $w^{i+1} = z$.

The induction proof above implies that $c[u_1^n, \dots, u_n^n] \Longrightarrow w^n$. From the definition of language inclusion preorder and the premises $\xrightarrow{f_i} q_i'$ and $q_i' R u_i^n$, it follows that $\xrightarrow{f_i} u_i^n$ for each $i : 1 \leq i \leq n$. It follows that $t = c[f_1, \dots, f_n] \Longrightarrow w^n$. By definition of U and the fact that $r \in F$ if $B \cap F \neq \emptyset$, it follows that for all $i : 1 \leq i \leq n$, $w^i \in F$ provided that $\mathsf{B} \in F_{\equiv_C}$. Thus, in the claim of the lemma, it suffices to take $w = w^n$. $\square$

Proof (Theorem 1). The inclusion $L(A_{\equiv_C}) \supseteq L(A)$ is trivial. Let $t \in L(A_{\equiv_C})$, i.e., $t \Longrightarrow \mathsf{B}$ for some block B where $\mathsf{B} \cap F \neq \emptyset$. Lemma 8 implies that $t \Longrightarrow w$ such that $w \in F$. $\square$

Note that the Theorem 1 also covers the case of reducing automata using equivalences included in LP. Indeed, given any preorder $R \subseteq LP$, the identity is always an upward simulation wrt. R. Then, the combined preorder $R \oplus id^{-1}$ equals R, which means that we can reduce the automaton using $\equiv_R$. In particular, this elegantly covers as special cases the proofs of correctness of reducing automata using downward bisimulations and simulation equivalences stated in [2].

Corollary 1. $L(A_{\equiv_R}) = L(A)$ *for any tree automaton* A *and each preorder* $R \subseteq LP$.

5 Variants of Combined Relations

Theorem 1 and Lemma 2 allow us to consider quite a large spectrum of relations suitable for tree automata reduction. We now examine properties of the relations from this spectrum that arise when we consider the identity, the maximal downward bisimulation, and the maximal downward simulation as the inducing relation $R \subseteq LP$ for both the maximal upward bisimulation and upward simulation.

Our notation for the various types of combined equivalences that we consider consists of two parts: a relation symbol and an additional symbol above the relation symbol. The relation symbol denotes the type of the inducing downward relation. Namely, $=$ denotes the identity, $\simeq$ denotes the maximal downward bisimulation, and $\sim$ the maximal downward simulation. The additional symbol then denotes the type of the upward relation. We use $\bullet$ for the maximal upward bisimulation and $\circ$ for the maximal upward simulation. No additional symbol corresponds to the maximum equivalence embedded in the downward relation itself—the downward (bi)simulations can be viewed as compositions where the role of the upward relation is played by the identity. For example,

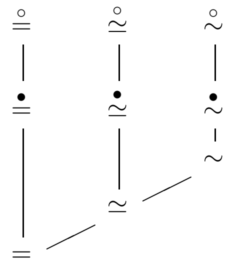
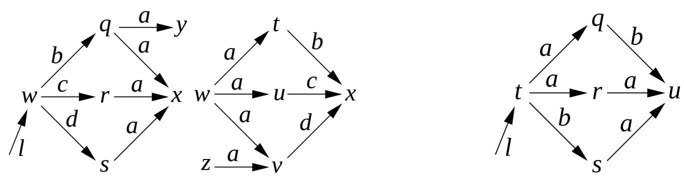


Fig. 1. Coarseness of various types of combined equivalences

$\overset{\circ}{\simeq}$ denotes the relation $\equiv_{R \oplus U^{-1}}$ where R is the maximal downward bisimulation and U is the maximal upward simulation induced by R. In what follows, we will implicitly consider all the downward and upward (bi-)simulations that we will be dealing with to be the maximal ones.

From the definition of a combined preorder, it clearly follows that, for a fixed inducing relation R, if we are choosing the type of the upward relations U from the strongest one to the coarsest one, i.e., starting from the identity and going through the upward bisimulation wrt. R to the upward simulation wrt. R, we obtain coarser and coarser combined preorders $R \oplus U^{-1}$.



(a) Δ_1 (b) Δ_2

Fig. 2. Transition relations of automata proving the non-inclusion relationships from Figure 1

On the other hand, if the inducing preorder R is growing, from the definition of the upward (bi)simulation, we can see that the maximal upward (bi)simulation U wrt. R and thus also the relation $R \circ (U^{-1})$ are growing too. But, when having $R \circ (U^{-1})$ computed and then computing the preorder $R \oplus (U^{-1})$ from it (via computing the set of R-extensible edges), the relation R acts as a restriction. A bigger relation R can cause that more edges are not R-extensible. In general, having two preorders $R_1 \subseteq R_2 \subseteq LP$, we are guaranteed that the maximal upward (bi)simulation U_1 wrt. R_1 is included in the maximal upward (bi)simulation U_2 wrt. R_2. Therefore, we know that $R_1 \circ U_1^{-1} \subseteq R_2 \circ U_2^{-1}$, but the combined preorders $R_1 \oplus U_1^{-1}$ and $R_2 \oplus U_2^{-1}$ can be incomparable.[3]

Based on these observations, we obtain the partial ordering of all the considered types of combined equivalences according to inclusion which is depicted in Figure 1. For an automaton A, we denote by $\equiv(A)$ the combined equivalence of type $\equiv$ on A. In the figure, the line from $\equiv_1$ up to $\equiv_2$ means that for any automaton A, $\equiv_1(A) \subseteq \equiv_2(A)$. It is not hard to find an automaton A showing that all these relationships are strict, i.e., such that for each of the edges in the figure, $\equiv_1(A) \subsetneq \equiv_2(A)$. We construct such an automaton in Example 1.

Example 1. Let $Q = \{q,r,s,t,u,v,w,x,y,z\}$ be a set of states and let Σ be a ranked alphabet such that $\Sigma_0 = \{l\}$ and $\Sigma_1 = \{a,b,c\}$. The TA $A = (Q,\Sigma,\Delta_1,\{x\})$, with Δ_1 depicted in Figure 2(a), proves strictness of the relations in Figure 1. For each two types or relations from Figure 1 such that $\equiv_2$ is above $\equiv_1$, $\equiv_1(A) \subsetneq \equiv_2(A)$ holds. In the table below, for each type of combination, we list nontrivial equivalence classes of the resulting combined equivalence:

$\overset{\circ}{=}$: $\{q,r,s\}$	$\overset{\circ}{\simeq}$: $\{t,u\},\{q,r,s\}$	$\overset{\circ}{\sim}$: $\{t,u,v\},\{q,r,s\},\{x,z\}$
$\overset{\bullet}{=}$: $\{r,s\}$	$\overset{\bullet}{\simeq}$: $\{t,u\},\{r,s\}$	$\overset{\bullet}{\sim}$: $\{t,u,v\},\{r,s\}$
$=$:	$\simeq$: $\{t,u\}$	$\sim$: $\{t,u,v\}$

It is now easy to check that all the inclusions from Figure 1 are strict for the automaton A.

To complete the picture, we need to show that the types of combined relations that are not connected in Figure 1 are really incomparable. In other words, that for each such a pair $\equiv_1, \equiv_2$ of types of combined equivalences that are not connected in Figure 1 there exists an automaton A such that neither $\equiv_1(A) \not\subseteq \equiv_2(A)$ nor $\equiv_1(A) \not\supseteq \equiv_2(A)$. We construct such automata within Example 2.

Example 2. Let $Q = \{q,r,s,t,u,v\}$ be a set of states and let Σ be a ranked alphabet such that $\Sigma_0 = \{l\}$ and $\Sigma_1 = \{a,b,c\}$. All the incomparability results show up taking automata $A_1 = (Q \setminus \{v\},\Sigma,\Delta_2,\{u\})$ and $A_2 = (Q,\Sigma,\Delta_2 \cup \{v \xrightarrow{a} q\},\{u\})$ where the

[3] Although, in our experiments, the former one usually *is* included in the latter one.

transition relation Δ_2 is depicted in Figure 2(b). One can easily check that $\overset{\bullet}{=}(A_1)$ and $\overset{\circ}{=}(A_1)$ define just one nontrivial equivalence class $\{r,s\}$ and thus they are incomparable with $\overset{\circ}{\simeq}(A_1), \overset{\bullet}{\sim}(A_1), \overset{\circ}{\sim}(A_1)$ that define only one nontrivial equivalence class $\{q,r\}$. In the case of the automaton A_2, the added transition $v \xrightarrow{a} q$ distinguishes the downward simulation from the downward bisimulation. Analogically as for A_1, we have that $\overset{\bullet}{\simeq}(A_2)$ and $\overset{\circ}{\simeq}(A_2)$ define just one nontrivial equivalence class $\{r,s\}$ and thus they are incomparable with $\overset{\bullet}{\sim}(A_2)$ and $\overset{\circ}{\sim}(A_2)$ that define only one nontrivial equivalence class $\{q,r\}$. This gives all the incomparability relationships.

According to our experiments presented in Section 7, the reduction capabilities are rising when we move in Figure 1 not only in the bottom-up direction (according to the edges), but also in the left-right direction (as though within a full diamond). As a trade-off, the computational complexity of constructing the relations is rising in the same way from the bottom to the top and from the left to the right.

We note that all these results carry over to word automata. The inclusion properties from Figure 1 hold there too as word automata can be seen as a special case of tree automata. Moreover, our automata examples proving strictness of the relationships and incomparability relationships are built using just leaf and unary rules, and so, in fact, they are valid for word automata too.

6 Computing the Proposed Relations

Below, we first briefly discuss methods for computing downward and upward (bi-)simulations proposed in earlier works. Then, we propose an algorithm for computing the combined relations and analyse its complexity. For the rest of the section, let us fix a tree automaton $A = (Q, \Sigma, \Delta, F)$ and let $n = |Q|$, $m = |\Delta|$, $\ell = |\Sigma|$.

6.1 Computing Downward and Upward (Bi-)Simulations

The problem of computing (bi)simulations over tree automata is addressed in [2, 1, 9]. In [2, 1], a quite general method for computing tree (bi-)simulations via transforming this problem to special instances of the classical problem of computing (bi-)simulations over labelled transition systems (LTS) is proposed. Classical (bi)simulation algorithms like [10, 11] are then applied to the LTS obtained from the translation.

Using the above approach, one obtains algorithms for computing the maximal downward and upward bisimulations in time $O(\hat{r}^3\, m \log n)$ and $O(m \log(n+\ell) + T(R))$, respectively, where $T(R)$ denotes the complexity of computing the inducing relation R.

For the tree simulations, we obtain algorithms with the following complexities when using the translation to LTS: Let D be the maximal downward simulation on A and let $|Lhs(A)/{\equiv_D}|$ be the size of the partitioning of the left-hand sides of the transition rules according to D. D can be computed in time $O((\ell + \hat{r}) \cdot |Lhs| \cdot |Lhs/{\equiv_D}| + m \cdot |Lhs/{\equiv_D}|)$, which can be roughly approximated by $O((\hat{r} + \ell)\, m^2)$. Let U be the maximal upward simulation on A wrt. an inducing preorder R, we denote the set of environments partitioned with respect to U as $Env/{\equiv_U}$. Assuming that $T(R)$ is the time of computing the inducing relation R, U can be computed in time $O((\ell\, |Env| + \hat{r}\, m)\, |Env/{\equiv_U}| + \hat{r}^2\, m \log |Env| + T(R))$, which can be roughly approximated by $O(\ell\, \hat{r}^2\, m^2 + T(R))$.

Let us add that specialised algorithms for computing the downward bisimulation and upward bisimulation induced by the identity were proposed in [9]. These algorithms run in time $O(\hat{r}^2\, m \log n)$ and $O(\hat{r}\, m \log n)$, respectively.

6.2 Computing the Combined Relations

Given a preorder $R \subseteq LP$ and an upward simulation U wrt. R, Lemma 5 offers us an easy way how to compute the combined preorder $C = R \oplus (U^{-1})$—it is sufficient to start by computing the relation $R \circ (U^{-1})$ and then just erase all edges that are not R-extensible. Using suitable data structures, this computation starting from the relations U and R can be implemented in time $O(\min\{|R| \cdot |Q|, |U| \cdot |Q|\})$ as follows.

We encode a relation ρ on Q as an array indexed by elements of Q of lists of elements of Q. If q is present in a list with index r, then $(r,q) \in \rho$. Note that given a boolean matrix representation of the relation, the "array of lists"-representation can be derived in time $O(|Q|^2)$. Note also that as U and R are reflexive, we have that $|U|, |R| \geq |Q|$ and thus $|Q|^2 \leq \min\{|R| \cdot |Q|, |U| \cdot |Q|\}$. Let arrays of lists $\mathsf{R}, \mathsf{U}^{-1}$ encode relations R, U^{-1}.

The relation $R \circ (U^{-1})$ represented by a boolean matrix M can be computed in the following way: (1) Initialise all entries of M to *false*. (2) For each $q \in Q$, pass through all elements of the list $\mathsf{R}[q]$, and for each $r \in \mathsf{R}[q]$, pass through all elements s of $\mathsf{U}^{-1}[r]$, and set $\mathsf{M}[q,s]$ to *true*. This procedure takes time $O(|\{(q,r,s) \mid (q,r) \in R \wedge (r,s) \in U^{-1}\}|) \subseteq O(\min\{|R| \cdot |Q|, |U| \cdot |Q|\})$.

Then we compute a boolean matrix representation E of the set of R-extensible edges as follows: (3) We initialise E as a copy of the matrix M (representing $R \circ U^{-1}$), and in the subsequent steps (4) and (5), we erase all non-R-extensible edges from E. In step (4), we proceed in the following way: For all $q \in Q$, for all $r \in \mathsf{R}[q]$, for all $s \in \mathsf{U}^{-1}[r]$, if not $\mathsf{M}[q,s]$ (i.e., $(q,s) \notin R \circ U^{-1}$), then $\mathsf{E}[q,s] = \mathit{false}$. Step (5) is symmetrical to step (4), but we swap the roles of R and U^{-1}. This gives us the set of R-extensible edges $R \oplus U^{-1}$ represented by the matrix E. The complexity of the steps (3), (4), (5) is in $O(|\{(q,r,s) \mid (q,r) \in R \wedge (r,s) \in U^{-1}\}| + |\{(q,r,s) \mid (q,r) \in U^{-1} \wedge (r,s) \in R\}|)$, which is again in $O(\min\{|R| \cdot |Q|, |U| \cdot |Q|\})$.

7 Experiments

We have implemented our algorithms in a prototype tool written in Java. We have used the tool on a number of tree automata from the frameworks of *regular tree model checking* (RTMC) and *abstract regular tree model checking* (ARTMC) [8, 4, 6, 7].

RTMC is the name of a family of techniques for analysing infinite-state systems such as parameterised networks of processes, systems with queues, stacks, unbounded integers, and/or dynamic linked data structures like lists or trees. In RTMC, states are represented by trees, sets of states by tree automata, and transitions by tree transducers (or, sometimes, also by some specialised operations on tree automata). ARTMC is a combination of RTMC and the abstract-check-refine paradigm which usually greatly improves the efficiency of the technique. Most of the algorithms in the frameworks of both RTMC and ARTMC rely crucially on efficient automata reduction methods since the size of the generated automata often explodes, making computations infeasible without a reduction.

The tree automata that we have considered in our experiments arose within various computations within the frameworks of RTMC and ARTMC. Our experimental evaluation was carried out on an AMD Athlon 64 X2 2.19GHz PC with 2.0 GB RAM. We have compared the size of tree automata after reducing them with all the different reduction techniques considered in this paper.

Table 1. The obtained reduction in percent and the computation time in seconds for the various considered relations applied for reducing TA obtained from RTMC and ARTMC case studies. The size of the TA is the number of their states plus the number of their transition rules.

TA		$\sim$		$\overset{\circ}{=}$		$\overset{\circ}{\simeq}$		$\overset{\circ}{\sim}$	
origin	size	reduction	time	reduction	time	reduction	time	reduction	time
ARTMC	195	18%	0.5 s	2%	0.5 s	23%	0.5 s	61%	1.0 s
RTMC	613	27%	3.5 s	19%	2.0 s	19 %	2.5 s	88%	5.1 s
RTMC	909	52%	3.6 s	72%	3.1 s	82%	3.4 s	89%	35.1 s
ARTMC	2029	10%	27.0 s	37%	26.0 s	33%	29.0 s	93%	39.0 s
RTMC	2403	26%	31.0 s	0%	25.0 s	0%	34.0 s	82%	37.1 s

TA		$\simeq$		$\overset{\bullet}{=}$		$\overset{\bullet}{\simeq}$		$\overset{\bullet}{\sim}$	
origin	size	reduction	time	reduction	time	reduction	time	reduction	time
ARTMC	195	18%	0.1 s	2%	0.5 s	23%	0.2 s	23%	0.6 s
RTMC	613	0%	0.3 s	0%	0.4 s	0%	0.8 s	27%	3.7 s
RTMC	909	14%	0.6 s	72%	0.4 s	82%	0.8 s	83%	4.1 s
ARTMC	2029	10%	1.7 s	14%	1.4 s	19%	3.1 s	44%	29.0 s
RTMC	2403	0%	0.3 s	0%	0.6 s	0%	0.7 s	27%	31.0 s

Table 1 shows the computation time and the reduction (in percent) for the different relations within the considered framework and illustrates that we have really obtained a wide spectrum of relations differing in their reduction capabilities and computational complexity. As can be seen from the results, $\overset{\circ}{\sim}$ gives the best reduction in all experiments, but it also suffers from a high computation time. Combining simulations and bisimulations does not give the same amount of reduction as the combined simulation, but the computation time is lower and the reduction is better than $\overset{\bullet}{\simeq}$. Note that no attempt to optimise the implementation of any of the relations was done, and therefore the computation times could probably be much lower with an optimised implementation for all of them.

8 Conclusions and Future Work

We have presented a uniform framework for deriving equivalence relations suitable for reducing tree automata based on combining upward and downward simulation and bisimulation relations. The framework generalises and extends our previous work [2, 1], it employes a significantly improved and more efficient technique of combining upward and downward (bi-)simulation relations, and offers a large spectrum of relations suitable for reducing tree automata. We establish a partial ordering of the obtained combined relations according to their reduction capabilities. We have also performed a number of experiments with automata from the area of (abstract) regular tree model checking that show practical applicability of the obtained relations. We conclude that the considered relations really offer a fine choice of balance in the trade-off between reduction capabilities and computational requirements.

The framework is built on quite general principles and we believe that it can be extended to more advanced types of automata such as guided tree automata, nested word automata, or hedge automata that find their use in many applications in formal verification, decision procedures of various logics, structured document processing, or natural language processing. From the practical point of view, it is also interesting to investigate more efficient techniques of computing the (bi-)simulation relations, e.g., by computing them in a symbolic way (for symbolically encoded automata).

Furthermore, it can be interesting to explore more deeply the principles of the proposed combination of downward and upward (bi-)simulation relations. One can, for instance, think of defining still weaker types of relations preserving the language of tree automata by using the combined relations repeatedly as inducing relations.

Acknowledgement. We would like to thank Ahmed Bouajjani for fruitful discussions on the subject of the paper.

References

1. P. Abdulla, A. Bouajjani, L. Holík, L. Kaati, and T. Vojnar. Composed Bisimulation for Tree Automata., 2008. Accepted at CIAA'08.
2. P. Abdulla, A. Bouajjani, L. Holík, L. Kaati, and T. Vojnar. Computing Simulations over Tree Automata: Efficient Techniques for Reducing Tree Automata. In *Proc. of TACAS'08*, LNCS. Springer, 2008. An extended version appeared as the technical report FIT-TR-2007-001 of FIT, Brno University of Technology.
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, B. Jonsson, P. Mahata, and J. d'Orso. Regular Tree Model Checking. In *Proc. of CAV'02*, volume 2404 of *LNCS*. Springer, 2002.
5. P. Abdulla, A. Legay, J. d'Orso, and A. Rezine. Simulation-Based Iteration of Tree Transducers. In *Proc. of TACAS'05*, volume 3440 of *LNCS*. Springer, 2005.
6. 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.
7. A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract Regular Tree Model Checking. *ENTCS*, 149:37–48, 2006.
8. A. Bouajjani and T. Touili. Extrapolating Tree Transformations. In *Proc. of CAV'02*, volume 2404 of *LNCS*. Springer, 2002.
9. 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. Springer, 2007.
10. R. Paige and R. Tarjan. Three Partition Refinement Algorithms. *SIAM Journal on Computing*, 16:973–989, 1987.
11. F. Ranzato and F. Tapparo. A New Efficient Simulation Equivalence Algorithm. In *Proc. of LICS'07*. IEEE CS, 2007.