Bisimulation Minimisation for Weighted Tree Automata*

Johanna Högberg[1], Andreas Maletti[2], and Jonathan May[3]

[1] Department of Computing Science, Umeå University
S–90187 Umeå, Sweden johanna@cs.umu.se

[2] Faculty of Computer Science, Technische Universität Dresden
D–01062 Dresden, Germany maletti@tcs.inf.tu-dresden.de

[3] Information Sciences Institute, University of Southern California
Marina Del Rey, CA 90292 jonmay@isi.edu

Abstract. We generalise existing forward and backward bisimulation minimisation algorithms for tree automata to weighted tree automata. The obtained algorithms work for all semirings and retain the time complexity of their unweighted variants for all additively cancellative semirings. On all other semirings the time complexity is slightly higher (linear instead of logarithmic in the number of states). We discuss implementations of these algorithms on a typical task in natural language processing.

1 Introduction

By the Myhill-Nerode theorem there exists, for every regular string language L, a unique (up to isomorphism) minimal deterministic finite automaton (dfa) that recognises L. It was a breakthrough when Hopcroft [1] presented an $O(n \log n)$ minimisation algorithm for dfa where n is the number of states. This still up-to-date bound was obtained by partitioning the state space through a "process the smaller half" strategy. However, in general there exists no unique minimal non-deterministic finite automaton (nfa) recognising a given regular language. Meyer and Stockmeyer [2] proved that minimisation of nfa is PSPACE-complete. The minimisation problem for nfa with n states cannot even be efficiently approximated within the factor $o(n)$, unless P = PSPACE [3]. This meant that the problem had to be simplified; either by restricting the domain to a smaller class of devices, or by surrendering every hope of a non-trivial approximation bound. Algorithms that minimise with respect to a *bisimulation* are examples of the latter approach. The concept of bisimularity was introduced by Milner [4] as a formal tool to investigate transition systems. Simply put, two transition systems are bisimulation equivalent if their behaviour—in response to a sequence of actions—cannot be distinguished by an outside observer. Although bisimulation equivalence, as interpreted for various devices, implies language equality, the opposite does not hold in general. We consider weighted tree automata (wta) [5], which are a joint generalisation of tree automata [6, 7] and weighted automata [8].

* This work was partially supported by NSF grant IIS-0428020

Classical tree automata can then be seen as wta with weights in the Boolean semiring, i.e. a transition has weight *true* if it is present, and *false* otherwise.

One type of bisimulation, called *forward bisimulation* in [9, 10], restricts bisimilar states to have identical futures. The future of a state q is the tree series of contexts that is recognised by the wta if the computation starts with the state q and weight 1 at the unique position of the special symbol $\square$ in the context. A similar condition is found in the Myhill-Nerode congruence for a tree language [11] or even in the Myhill-Nerode congruence [12] for a tree series. Let us explain it on the latter. Two trees t and u are equal in the Myhill-Nerode congruence for a given tree series $\mathcal{S}$ over the field $(A, +, \cdot, 0, 1)$, if there exist nonzero coefficients $a, b \in A$ such that for all contexts C we observe that $a^{-1} \cdot (\mathcal{S}, C[t]) = b^{-1} \cdot (\mathcal{S}, C[u])$. The coefficients a and b can be understood as the weights of t and u, respectively. In contrast to the Myhill-Nerode congruence, a forward bisimulation requires a local condition on the tree representation. The condition is strong enough to enforce equivalent futures, but not too strong which is shown by the fact that, on a deterministic *all-accepting* [13] wta M over a field [14] or a wta M over the Boolean semiring [10], minimisation via forward bisimulation yields the unique (up to isomorphism) minimal deterministic wta that recognises the same tree series as M.

The other type of bisimulation we will consider is called *backward bisimulation* in [9, 10]. Backward bisimulation also uses a local condition on the tree representation that enforces that the past of any two bisimilar states is equal. The past of a state is the series that is recognised by the wta if that particular state would be the only final state and its final weight would be 1 (i.e., the past of a state q is the series that maps an input tree t to $h_\mu(t)_q$; see Sect. 2).

The idea behind bisimulation minimisation is to discover and collapse states that in some sense exhibit the same behaviour, thus freeing the input automaton of redundancy. This implies a search for the coarsest relation on the state space that meets the local conditions of the bisimulation relation that we are interested in. The $O\big(n^2 \log n\big)$ minimisation algorithm for nfa by Paige & Tarjan [15] could be called a forward bisimulation minimisation. Bisimulation minimisation of tree automata is discussed in [10]. The paper [10] presents two minimisation algorithms that are based on forward and backward bisimulation and run in time $O(rn^{r+1} \log n)$ and $O(r^2 n^{r+1} \log n)$, respectively, where r is the maximal rank of the input symbols and n is the number of states. In this paper, we generalise these results to weighted tree automata and obtain minimisation algorithms that work for arbitrary semirings in $O(rn^{r+2})$ and $O\big(r^2 n^{r+2}\big)$ for the forward and backward approach, respectively. The counting argument used in [15] and later in [10] is no longer applicable: it was devised for the Boolean semiring and does not generalise. However, when cancellative semirings are considered, we can improve the algorithms to run in $O(rn^{r+1} \log n)$ and $O(r^2 n^{r+1} \log n)$ for the forward and backward approach, respectively, by taking advantage of the "process the smaller half" strategy of Hopcroft. When the forward algorithm is given a deterministic wta, it yields an equivalent deterministic wta in time $O(rn^{r+1})$, which can be optimised to $O(rn^r \log n)$ for additively cancellative semirings.

There are advantages that support having two algorithms. First, forward and backward bisimulation minimisation only yield a minimal wta with respect to the corresponding bisimulation. Thus applying forward and backward bisimulation minimisation in an alternating fashion commonly yields a yet smaller wta. Since both minimisation procedures are very efficient, this approach also works in practice. For the problem of *tree language model* minimisation, discussed in Sect. 5, we minimised our candidate wta in an alternating fashion and found that we were able to get equally small wta after two iterations beginning with backward or three iterations beginning with forward. Our implementation typically ran in $\Theta(rn^{r+1}\log n)^{0.36}$ and $\Theta(r^2n^{r+1}\log n)^{0.36}$ for forward and backward, respectively; well below the theoretical upper bound.

Second, in certain domains one type of bisimulation minimisation is more effective. For example, backward bisimulation is ineffective on deterministic wta because no two states have the same past[4]. On the other hand, wta recognising languages of trees that vary greatly in the root but little in the leaves (for example, syntax parses of natural language sentences), will benefit more from backward bisimulation minimisation than forward. When presented with an unknown wta, we know no way to say for certain which method of minimisation is superior, so it is beneficial to have both.

The bisimulation introduced in [16] can be seen as a combination of backward and forward bisimulation. Containing the restrictions of both, it is less efficient than backward bisimulation when applied to the minimisation of nondeterministic automata, but just as expensive to calculate, and unlike forward bisimulation it does not yield the standard algorithm when applied to deterministic automata. The pair of algorithms presented in this paper thus supersedes that of [16].

2 Preliminaries

We write $\mathbb{N}$ to denote the set of natural numbers including zero. The subset $\{k, k+1, \ldots, n\}$ of $\mathbb{N}$ is abbreviated to $[k, n]$, and the cardinality of a set S is denoted by $|S|$. We abbreviate the Cartesian product $S \times \cdots \times S$ with n factors by S^n, and the inclusion $d_i \in D_i$ for all $i \in [1, k]$ as $d_1 \cdots d_k \in D_1 \cdots D_k$.

Let $\mathcal{P}$ and $\mathcal{R}$ be equivalence relations on S. We say that $\mathcal{P}$ is *coarser* than $\mathcal{R}$ (or equivalently: $\mathcal{R}$ is a *refinement* of $\mathcal{P}$), if $\mathcal{R} \subseteq \mathcal{P}$. The *equivalence class* (or *block*) of an element $s \in S$ with respect to $\mathcal{R}$ is the set $[s]_\mathcal{R} = \{s' \mid (s, s') \in \mathcal{R}\}$. Whenever $\mathcal{R}$ is obvious from the context, we simply write $[s]$ instead of $[s]_\mathcal{R}$. It should be clear that $[s]$ and $[s']$ are equal if s and s' are in relation $\mathcal{R}$, and disjoint otherwise, so $\mathcal{R}$ induces a partition $(S/\mathcal{R}) = \{[s] \mid s \in S\}$ of S.

A *semiring* is a tuple $(A, +, \cdot, 0, 1)$ such that $(A, +, 0)$ is a commutative monoid, $(A, \cdot, 1)$ is a monoid, $\cdot$ distributes (both-sided) over $+$, and 0 is an absorbing element with respect to $\cdot$. We generally assume that $\cdot$ binds stronger than $+$, so $a + b \cdot c$ is interpreted as $a + (b \cdot c)$. The semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$ is said to be *cancellative* if $a + b = a + c$ implies that $b = c$ for every $a, b, c \in A$.

[4] The alteration technique is thus useless for deterministic devices.

A *ranked alphabet* is a finite set of symbols $\Sigma = \bigcup_{k\in\mathbb{N}} \Sigma_{(k)}$ which is partitioned into pairwise disjoint subsets $\Sigma_{(k)}$. The set T_Σ of *trees over* Σ is the smallest set of strings over Σ such that $f\,t_1 \cdots t_k$ in T_Σ for every f in $\Sigma_{(k)}$ and all $t_1, \ldots, t_k$ in T_Σ. We write $f[t_1, \ldots, t_k]$ instead of $f\,t_1 \cdots t_k$ unless k is zero.

A *tree series* over the ranked alphabet Σ and semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$ is a mapping from T_Σ to A. The set of all tree series over Σ and $\mathcal{A}$ is denoted by $\mathcal{A}\langle\langle T_\Sigma\rangle\rangle$. Let $\mathcal{S} \in \mathcal{A}\langle\langle T_\Sigma\rangle\rangle$. We write $(\mathcal{S}, t)$ with $t \in T_\Sigma$ for $\mathcal{S}(t)$. A *weighted tree automaton* M (for short: wta) [17] is a tuple $(Q, \Sigma, \mathcal{A}, F, \mu)$, where Q is a finite nonempty set of *states*; Σ is a ranked alphabet (of *input symbols*); $\mathcal{A} = (A, +, \cdot, 0, 1)$ is a semiring; $F \in A^Q$ is a *final weight distribution*; and $\mu = (\mu_k)_{k\in\mathbb{N}}$ with $\mu_k : \Sigma_{(k)} \to A^{Q^k \times Q}$ is a *tree representation.*

We define $h_\mu \colon T_\Sigma \to A^Q$ for every $\sigma \in \Sigma_{(k)}$, $q \in Q$, and $t_1, \ldots, t_k \in T_\Sigma$ by

$$h_\mu(\sigma[t_1, \ldots, t_k])_q = \sum_{q_1,\ldots,q_k\in Q} \mu_k(\sigma)_{q_1\cdots q_k,q} \cdot h_\mu(t_1)_{q_1} \cdot \ldots \cdot h_\mu(t_k)_{q_k} \enspace .$$

Finally, the tree series *recognised by* M is given by $(\|M\|, t) = \sum_{q\in Q} F_q \cdot h_\mu(t)_q$ for every tree $t \in T_\Sigma$ and denoted by $\|M\|$.

3 Forward Bisimulation

Foundation. Let $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ be a wta. Roughly speaking, a *forward bisimulation on* M is an equivalence relation on Q such that equivalent states react equivalently to future inputs. We enforce this behaviour with only a local condition on μ and F. Let $\Box \notin Q$. The set $C^Q_{(k)}$ of *contexts (over* Q*)* is given by $\{w \in (Q \cup \{\Box\})^k \mid w \text{ contains } \Box \text{ exactly once}\}$, and for every context c and state q we write $c[\![q]\!]$ to denote the word that is obtained from c by replacing the special symbol $\Box$ with q. Henceforth, we assume that the special symbol $\Box$ occurs in no set of states of any wta.

Definition 1 (cf. [9, Definition 3.1]). *Let $\mathcal{R} \subseteq Q \times Q$ be an equivalence relation. We say that $\mathcal{R}$ is a* forward bisimulation on M *if for every (p, q) in $\mathcal{R}$ we have (i) $F(p) = F(q)$ and (ii) $\sum_{r\in D} \mu_k(\sigma)_{c[\![p]\!],r} = \sum_{r\in D} \mu_k(\sigma)_{c[\![q]\!],r}$ for every $\sigma \in \Sigma_{(k)}$, block D in $(Q/\mathcal{R})$, and context c of $C^Q_{(k)}$.*

Example 2. Let $\Delta = \Delta_{(0)} \cup \Delta_{(2)}$ be the ranked alphabet where $\Delta_{(0)} = \{\alpha\}$ and $\Delta_{(2)} = \{\sigma\}$. The mapping $\textsc{zigzag}$ from T_Δ to $\mathbb{N}$ is recursively defined for every t_1, t_2, and t_3 in T_Δ by $\textsc{zigzag}(\alpha) = 1$ and $\textsc{zigzag}(\sigma[\alpha, t_2]) = 2$ and $\textsc{zigzag}(\sigma[\sigma[t_1, t_2], t_3]) = 2 + \textsc{zigzag}(t_2)$. Consider the wta $N = (P, \Delta, \mathbb{N}, G, \nu)$ with the semiring $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$ and $P = \{l, r, L, R, \bot\}$, $G(l) = G(L) = 1$ and $G(p) = 0$ for every $p \in \{r, R, \bot\}$, and

$$1 = \nu_0(\alpha)_{\varepsilon,l} = \nu_0(\alpha)_{\varepsilon,R} = \nu_0(\alpha)_{\varepsilon,\bot} = \nu_2(\sigma)_{r\bot,l} = \nu_2(\sigma)_{\bot l,r} = \nu_2(\sigma)_{\bot\bot,l}$$
$$1 = \nu_2(\sigma)_{R\bot,L} = \nu_2(\sigma)_{\bot L,R} = \nu_2(\sigma)_{\bot\bot,R} = \nu_2(\sigma)_{\bot\bot,\bot} \enspace .$$

All remaining entries in ν are 0. A straightforward induction shows that N recognises $\textsc{zigzag}$. Let us consider $\mathcal{P} = \{l, L\}^2 \cup \{r, R\}^2 \cup \{\bot\}^2$. We claim that $\mathcal{P}$

is a forward bisimulation on N. Obviously, $G(l) = G(L)$ and $G(r) = G(R)$. It remains to check Condition (ii) of Definition 1. We only demonstrate the computation on the symbol σ, the context $\bot\Box$ and the block $\{r, R\}$.

$$\sum_{p \in \{r,R\}} \nu_2(\sigma)_{\bot l,p} = 1 = \sum_{p \in \{r,R\}} \nu_2(\sigma)_{\bot L,p} \qquad \blacksquare$$

Let $\mathcal{R}$ be a forward bisimulation on M. We identify bisimilar states in order to reduce the size of the wta. Next we present how to achieve this. In essence, we construct a wta $(M/\mathcal{R})$ that uses only one state per equivalence class of $\mathcal{R}$.

Definition 3 (cf. [9, Definition 3.3]). *The* forward aggregated wta $(M/\mathcal{R})$ *is the wta* $((Q/\mathcal{R}), \Sigma, \mathcal{A}, F', \mu')$ *with* $F'([q]) = F(q)$ *for every state* q *of* Q*, and* $\mu'_k(\sigma)_{[q_1]\cdots[q_k],D} = \sum_{r \in D} \mu_k(\sigma)_{q_1\cdots q_k,r}$ *for every* $\sigma \in \Sigma_{(k)}$*, word* $q_1 \cdots q_k \in Q^k$*, and block* $D \in (Q/\mathcal{R})$*.*

Example 4. Recall the wta N and the forward bisimulation $\mathcal{P}$ of Example 2. Let us compute $(N/\mathcal{P}) = (P', \Delta, \mathbb{N}, G', \nu')$. We obtain $P' = \{[l], [r], [\bot]\}$, the final weights $G'([l]) = 1$ and $G'([r]) = G'([\bot]) = 0$ and the nonzero entries

$$1 = \nu'_2(\sigma)_{[r][\bot],[l]} = \nu'_2(\sigma)_{[\bot][l],[r]} = \nu'_2(\sigma)_{[\bot][\bot],[l]} = \nu'_2(\sigma)_{[\bot][\bot],[r]} = \nu'_2(\sigma)_{[\bot][\bot],[\bot]}$$
$$1 = \nu'_0(\alpha)_{\varepsilon,[l]} = \nu'_0(\alpha)_{\varepsilon,[r]} = \nu'_0(\alpha)_{\varepsilon,[\bot]} \ . \qquad \blacksquare$$

We should verify that the recognised tree series remains the same. The proof of this property is prepared in the next lemma. It essentially states that a collapsed state of $(M/\mathcal{R})$ works like the combination of its constituents in M.

Lemma 5 (cf. [9, Theorem 3.1]). *Let* $(M/\mathcal{R}) = (Q', \Sigma, \mathcal{A}, F', \mu')$*. Then* $h_{\mu'}(t)_D = \sum_{q \in D} h_\mu(t)_q$ *for every tree* $t \in T_\Sigma$ *and block* $D \in (Q/\mathcal{R})$*.*

The final step establishes that $\|(M/\mathcal{R})\| = \|M\|$. Consequently, collapsing a wta with respect to some forward bisimulation preserves the recognised series.

Theorem 6 (cf. [9, Theorem 3.1]). $\|(M/\mathcal{R})\| = \|M\|$.

The coarser the forward bisimulation $\mathcal{R}$ on M, the smaller $(M/\mathcal{R})$. Our aim is thus to find the coarsest forward bisimulation on M. First we show that a unique coarsest forward bisimulation on M exists.

Theorem 7. *There exists a coarsest forward bisimulation* $\mathcal{P}$ *on* M*, and* $(M/\mathcal{P})$ *admits only the identity as forward bisimulation.*

The previous theorem justifies the name *forward bisimulation minimisation*; given the coarsest forward bisimulation $\mathcal{P}$ on M, the wta $(M/\mathcal{P})$ is minimal with respect to forward bisimulation.

Algorithm. We now present a minimisation algorithm for wta that draws on the ideas presented in the previous section. Algorithm 1 searches for the coarsest forward bisimulation $\mathcal{R}$ on the input wta M by producing increasingly refined equivalence relations $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$. The first of these is the coarsest candidate solution that respects F. The relation $\mathcal{R}_{i+1}$ is derived from $\mathcal{R}_i$ by removing pairs

```
[input:           A wta M = (Q, Σ, A, F, μ);
[initially:       P_0     := Q × Q;
                  R_0     := ker(F) \ split(Q);
                  i       := 0;
[while R_i ≠ P_i: choose S_i ∈ (Q/P_i) and B_i ∈ (Q/R_i) such that
                     B_i ⊂ S_i and |B_i| ≤ |S_i| /2;
                  P_{i+1} := P_i \ cut(B_i);
                  R_{i+1} := (R_i \ split(B_i)) \ split(S_i \ B_i);
                  i       := i + 1;
[return:          (M/R_i);
```

Algorithm 1. A forward bisimulation minimisation algorithm for wta.

of states that prevent $\mathcal{R}_i$ from being a forward bisimulation. The algorithm also produces an auxiliary sequence of relations $\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, \ldots$ that are used to find these offending pairs. Termination occurs when $\mathcal{R}_i$ and $\mathcal{P}_i$ coincide. At this point, $\mathcal{R}_i$ is the coarsest forward bisimulation on M.

Before we discuss the algorithm, its correctness, and its time complexity, we extend our notation. For the rest of this section, let $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ be an arbitrary but fixed wta. We use the following shorthands in Alg. 1.

Definition 8. *Let B be a subset of Q. We write*

- *$cut(B)$ for the subset $(Q^2 \setminus B^2) \setminus (Q \setminus B)^2$ of $Q \times Q$, and*
- *$split(B)$ for the set of all pairs (p, q) in $Q \times Q$ such that $\sum_{r \in B} \mu_k(\sigma)_{c[\![p]\!],r}$ and $\sum_{r \in B} \mu_k(\sigma)_{c[\![q]\!],r}$ differ for some $\sigma \in \Sigma_{(k)}$ and $c \in C^Q_{(k)}$.*

Example 9. Let $N = (P, \Delta, \mathbb{N}, G, \nu)$ be the wta of Example 2 that recognises the tree series ZIGZAG. We will show the iterations of the algorithm on this example wta. Let us start with the initialisation: Clearly, $\mathcal{P}_0$ is $P \times P$, and $\mathcal{R}_0$ is the union $\{l, L\}^2 \cup \{r, R\}^2 \cup \{\bot\}^2$. In the first iteration, we select $S_0 = P$ and $B_0 = \{l, L\}$ and thus compute $\mathcal{P}_1$ to be $\{l, L\}^2 \cup \{r, R, \bot\}^2$, and $\mathcal{R}_1$ to be $\mathcal{R}_0$. Obviously, $\mathcal{P}_1$ is still different from $\mathcal{R}_1$, so the algorithm enters a second iteration. We now let $S_1 = \{r, R, \bot\}$ and $B_1 = \{\bot\}$, which yields $\mathcal{R}_2 = \mathcal{P}_2$, so the algorithm terminates and returns the aggregated wta $(N/\mathcal{R}_2)$. ■

We henceforth abbreviate $|Q|$ to n, and denote by r the maximum k such that $\Sigma_{(k)}$ is non-empty. As we will later argue, there exists a $t < n$ such that Alg. 1 terminates when $i = t$. We use the notations introduced in the algorithm when we set out to prove correctness and termination.

Lemma 10. *The relation $\mathcal{R}_i$ is a refinement of $\mathcal{P}_i$ for all $i \in [0, t]$.*

Lemma 10 ensures that $\mathcal{R}_i$ is a proper refinement of $\mathcal{P}_i$, for all $i \in [0, t-1]$. Since $\mathcal{P}_{i+1}$ is in turn, by definition, a proper refinement of $\mathcal{P}_i$, termination is guaranteed in less than n iterations. It follows that, up to the termination point t, we can always find blocks $B_i \in (Q/\mathcal{R}_i)$ and $S_i \in (Q/\mathcal{P}_i)$ such that B_i is contained in S_i, and the size of B_i is at most half of that of S_i.

Theorem 11. *Algorithm 1 returns the minimal wta $(M/\mathcal{P})$ with respect to forward bisimulation. Equivalently; $\mathcal{P}$ is the coarsest forward bisimulation on M.*

We now analyse the running time of Alg. 1. We use

$$m = \sum\nolimits_{k \in [0,r]} |\{(\sigma, q_1 \cdots q_k, q) \in \Sigma_{(k)} \times Q^k \times Q \mid \mu_k(\sigma)_{q_1 \cdots q_k, q} \neq 0\}| \enspace .$$

to denote the size of μ. In this paper, we assume that the tree representation is not sparse, i.e. that it contains some $\Omega\big(\sum_{k \in [0,r]} n^{k+1}\big)$ entries. For a discussion of how sparse representations affect the performance of the algorithm, see [14]. We also assume that semiring addition can be performed in constant time. We denote by μ_B^{f} the part of μ that contains entries of the form $\mu_k(\sigma)_{q_1 \cdots q_k, q}$, where $q \in B$. The overall time complexity of the algorithm is

$$O\Big(\textsc{Init}^{\mathrm{f}} + \sum\nolimits_{i \in [0,t-1]} \big(\textsc{Select}_i + \textsc{Cut}_i + \textsc{Split}_i^{\mathrm{f}}\big) + \textsc{Aggregate}^{\mathrm{f}}\Big) \enspace ,$$

where $\textsc{Init}^{\mathrm{f}}$, $\textsc{Select}_i$, $\textsc{Cut}_i$, $\textsc{Split}_i^{\mathrm{f}}$, and $\textsc{Aggregate}^{\mathrm{f}}$ are the complexity of: (i) the initialisation phase; (ii) the choice of S_i and B_i; (iii) the computation of $\mathcal{P}_{i+1}$; (iv) the computation of $\mathcal{R}_{i+1}$, and (v) the construction of the aggregated automaton $(M/\mathcal{R}_t)$; respectively.

Lemma 12. $\textsc{Init}^{\mathrm{f}}$ *and* $\textsc{Aggregate}^{\mathrm{f}}$ *are both in $O(m+n)$, whereas* $\textsc{Select}_i$ *is in $O(1)$,* $\textsc{Cut}_i$ *is in $O(|B_i|)$, and* $\textsc{Split}_i^{\mathrm{f}}$ *is in $O(r\,|\mu_{S_i}^{\mathrm{f}}|)$.*

In the worst case, $|S_i|$ equals $n - i$, which means that $\big|\mu_{S_i}^{\mathrm{f}}\big|$ is close to m.

Theorem 13. *Algorithm 1 has time complexity $O(rmn)$.*

We now consider a simplification of Alg. 1 for cancellative semirings. In essence, the second split in the computation of $\mathcal{R}_{i+1}$ can be omitted.

Lemma 14. *When the underlying semiring is cancellative, we can replace the computation of $\mathcal{R}_{i+1}$ in Alg. 1 simply by $\mathcal{R}_{i+1} = \mathcal{R}_i \setminus split(B_i)$.*

The optimised algorithm thus only splits against the block B_i, for each $i \in [0, t-1]$. As no state occurs in more than $\log n$ distinct B-blocks, we are able to obtain a lower time complexity:

Theorem 15. *Alg. 1 optimised for cancellative semirings is in $O(rm \log n)$.*

4 Backward Bisimulation

Foundation. Let $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ be a wta. In this section we investigate backward bisimulations [9]. We introduce the following notation. Let Π be a partition of Q. We write $\Pi_{(k)}$ for the set $\{D_1 \times \cdots \times D_k \mid D_1, \ldots, D_k \in \Pi\}$ for every $k \in \mathbb{N}$. Moreover, we write $\Pi_{(\leq k)}$ for the set $\Pi_{(0)} \cup \cdots \cup \Pi_{(k)}$.

Definition 16 (cf. [9, Definition 4.1]). *Let $\mathcal{R}$ be an equivalence relation on Q. If $\sum_{w \in L} \mu_k(\sigma)_{w,p} = \sum_{w \in L} \mu_k(\sigma)_{w,q}$ for every $(p, q) \in \mathcal{R}$, symbol σ in $\Sigma_{(k)}$, and word $L \in (Q/\mathcal{R})_{(k)}$, then we say that $\mathcal{R}$ is a* backward bisimulation *on M.*

Example 17. Let $N = (P, \Delta, \mathbb{N}, G, \nu)$ where $P = \{l, r, L, R, \perp\}$, Δ is as in Example 2, and $G(l) = 1$ and $G(p) = 0$ for every $p \in \{r, L, R, \perp\}$ and

$$\begin{aligned}
1 &= \nu_0(\alpha)_{\varepsilon,l} = \nu_0(\alpha)_{\varepsilon,r} = \nu_0(\alpha)_{\varepsilon,L} = \nu_0(\alpha)_{\varepsilon,R} = \nu_0(\alpha)_{\varepsilon,\perp} \\
1 &= \nu_2(\sigma)_{\perp L,R} = \nu_2(\sigma)_{\perp L,r} = \nu_2(\sigma)_{\perp l,r} \\
1 &= \nu_2(\sigma)_{R\perp,L} = \nu_2(\sigma)_{R\perp,l} = \nu_2(\sigma)_{r\perp,l} = \nu_2(\sigma)_{\perp\perp,\perp} \quad .
\end{aligned}$$

All remaining entries in ν are 0. The wta N also recognises ZIGZAG. We propose $\mathcal{P} = \{l\}^2 \cup \{r\}^2 \cup \{L, R, \perp\}^2$ as backward bisimulation. We note that $\nu_0(\alpha)_{\varepsilon,L}$ and $\nu_0(\alpha)_{\varepsilon,R}$ and $\nu_0(\alpha)_{\varepsilon,\perp}$ are all equal and $\sum_{p_1 p_2 \in [\perp][\perp]} \nu_2(\sigma)_{p_1 p_2, p} = 1$ and $\nu_2(\sigma)_{p_1 p_2, p} = 0$ for every $p \in \{L, R, \perp\}$ and $p_1, p_2 \in P$ such that $(p_1, \perp) \notin \mathcal{P}$ and $(p_2, \perp) \notin \mathcal{P}$. ∎

For the rest of this section, let $\mathcal{R}$ be a backward bisimulation on M. Next we define how to collapse M with respect to $\mathcal{R}$.

Definition 18 (cf. [9, Definition 3.3]). *The* backward aggregated wta $(M/\mathcal{R})$ *is the wta $((Q/\mathcal{R}), \Sigma, \mathcal{A}, F', \mu')$ such that (i) $F'(D) = \sum_{q \in D} F(q)$ for every block D of $(Q/\mathcal{R})$ and (ii) $\mu'_k(\sigma)_{D_1 \cdots D_k, [q]} = \sum_{w \in D_1 \cdots D_k} \mu_k(\sigma)_{w,q}$ for every symbol σ in $\Sigma_{(k)}$, word $D_1 \cdots D_k$ of blocks in $(Q/\mathcal{R})$, and state $q \in Q$.*

Example 19. Recall the wta N and the backward bisimulation $\mathcal{P}$ from Example 17. We obtain $(N/\mathcal{P}) = (P', \Delta, \mathbb{N}, G', \nu')$ with $P' = \{[l], [r], [\perp]\}$ and $G'([l]) = 1$ and $G'([r]) = G([\perp]) = 0$ and the nonzero tree representation entries

$$\begin{aligned}
1 &= \nu'_2(\sigma)_{[\perp][\perp],[r]} = \nu'_2(\sigma)_{[\perp][l],[r]} = \nu'_2(\sigma)_{[\perp][\perp],[l]} = \nu'_2(\sigma)_{[r][\perp],[l]} = \nu'_2(\sigma)_{[\perp][\perp],[\perp]} \\
1 &= \nu'_0(\alpha)_{\varepsilon,[l]} = \nu'_0(\alpha)_{\varepsilon,[r]} = \nu'_0(\alpha)_{\varepsilon,[\perp]} \quad .
\end{aligned}$$ ∎

Next we prepare Theorem 21, which will show that M and $(M/\mathcal{R})$ recognise the same series. First we prove that every state q of M recognises the same series as the state $[q]$ of $(M/\mathcal{R})$.

Lemma 20 (cf. [9, Theorem 4.2] and [18, Lemma 5.2]). *Let $(M/\mathcal{R})$ be $(Q', \Sigma, \mathcal{A}, F', \mu')$. Then $h_{\mu'}(t)_{[q]} = h_\mu(t)_q$ for every state $q \in Q$ and tree $t \in T_\Sigma$.*

The previous lemma establishes a nice property of bisimilar states. Namely, $h_\mu(t)_p = h_\mu(t)_q$ for every pair $(p, q) \in \mathcal{R}$ of bisimilar states and every tree $t \in T_\Sigma$.

Theorem 21 (cf. [9, Theorem 4.2] & [18, Lemma 5.3]). $\|(M/\mathcal{R})\| = \|M\|$.

Among all backward bisimulations on M, the coarsest one yields the smallest aggregated wta, and this wta admits only the trivial backward bisimulation.

Theorem 22. *There exists a coarsest backward bisimulation $\mathcal{P}$ on M, and the wta $(M/\mathcal{P})$ only admits the identity as backward bisimulation.*

$$
\begin{array}{|lll|}
\hline
\lceil\texttt{input:} & \texttt{A wta } M = (Q, \Sigma, \mathcal{A}, F, \mu)\texttt{;} & \\
\lceil\texttt{initially:} & \mathcal{P}_0 \ := \ Q \times Q\texttt{;} & \mathcal{L}_0 \ := \ (Q/\mathcal{P}_0)_{(\leq r)}\texttt{;} \\
\lfloor & \mathcal{R}_0 \ := \ \mathcal{P}_0 \setminus \mathit{split}^{\mathrm{b}}(\mathcal{L}_0)\texttt{;} & i \ := \ 0\texttt{;} \\
\lceil\texttt{while } \mathcal{R}_i \neq \mathcal{P}_i\texttt{:} & \texttt{choose } S_i \in (Q/\mathcal{P}_i) \texttt{ and } B_i \in (Q/\mathcal{R}_i) \texttt{ such that} & \\
 & \quad B_i \subset S_i \texttt{ and } |B_i| \leq |S_i|/2\texttt{;} & \\
 & \mathcal{P}_{i+1} := \ \mathcal{P}_i \setminus \mathit{cut}(B_i)\texttt{;} & \\
 & \mathcal{L}_{i+1} := \ (Q/\mathcal{P}_{i+1})_{(\leq r)}\texttt{;} & \\
 & \mathcal{R}_{i+1} := \ \big(\mathcal{R}_i \setminus \mathit{split}^{\mathrm{b}}(\mathcal{L}_{i+1}(B_i))\big) \setminus \mathit{split}^{\mathrm{b}}(\mathcal{L}_{i+1}(S_i \setminus B_i, \neg B_i))\texttt{;} & \\
\lfloor & i \quad := \ i + 1\texttt{;} & \\
\lceil\texttt{return:} & (M/\mathcal{R}_i)\texttt{;} & \\
\hline
\end{array}
$$

Algorithm 2. A backward bisimulation minimisation algorithm for wta.

Algorithm. We now show how Alg. 1 can be modified so as to minimise with respect to backward bisimulation. For this we recall the wta $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ with $n = |Q|$ states. Intuitively, the sum $\sum_{w \in D_1 \cdots D_k} \mu_k(\sigma)_{w,q}$ captures the extent to which q is reachable from states in $D_1 \cdots D_k$, on input σ, and is thus a local observation of the properties of q (cf. Definition 16). To decide whether states p and q are bisimilar, we compare $\sum_{w \in L} \mu_k(\sigma)_{w,p}$ and $\sum_{w \in L} \mu_k(\sigma)_{w,q}$ on increasing languages L. If we find a pair (σ, L) on which the two sums disagree, then (p, q) can safely be discarded from our maintained set of bisimilar states.

Definition 23. *Let $B, B' \subseteq Q$ and let $\mathcal{L} \subseteq \mathfrak{P}(Q^*)$ be a set of languages.*

- *We write $\mathcal{L}(B)$ to denote $\{L \cap Q^* B Q^* \mid L \in \mathcal{L}\}$.*
- *We write $\mathcal{L}(B, \neg B')$ when we mean $\{L \cap (Q \setminus B')^* \mid L \in \mathcal{L}(B)\}$.*
- *We write $\mathit{split}^{\mathrm{b}}(\mathcal{L})$ for the set of all (p, q) in $Q \times Q$ for which there exist $\sigma \in \Sigma_{(k)}$ and a language $L \in \mathcal{L} \cap \mathfrak{P}(Q^k)$ such that the sums $\sum_{w \in L} \mu_k(\sigma)_{w,p}$ and $\sum_{w \in L} \mu_k(\sigma)_{w,q}$ differ.*

Algorithm 2, as listed above, is obtained from Alg. 1 as follows. The initialisation of $\mathcal{R}_0$ is replaced with the assignment $\mathcal{R}_0 = \mathcal{P}_0 \setminus \mathit{split}^{\mathrm{b}}((Q/\mathcal{P}_0)_{(\leq r)})$, and the computation of $\mathcal{R}_{i+1}$ with

$$\mathcal{R}_{i+1} = \big(\mathcal{R}_i \setminus \mathit{split}^{\mathrm{b}}((Q/\mathcal{P}_{i+1})_{(\leq r)}(B_i))\big) \setminus \mathit{split}^{\mathrm{b}}((Q/\mathcal{P}_{i+1})_{(\leq r)}(S_i \setminus B_i, \neg B_i)) \ .$$

Example 24. Consider the execution of the backward bisimulation minimisation algorithm on the wta $N = (P, \Delta, \mathbb{N}, G, \nu)$ of Example 17. Clearly, $\mathcal{P}_0$ is $P \times P$. In the computation of $\mathcal{P}_0 \setminus \mathit{split}^{\mathrm{b}}(\mathcal{L}_0)$, the state space can be divided into $\{L, R, \bot\}$ and $\{l, r\}$, as $\sum_{w \in PP} \nu_k(\sigma)_{w,p}$ is 1 when p is in the former set, but 2, when in the latter. No additional information can be derived by inspecting $\nu_0(\alpha)_{\varepsilon,p}$ because this value equals 1 for every $p \in \{l, r, L, R, \bot\}$, so $\mathcal{R}_0 = \{l, r\}^2 \cup \{L, R, \bot\}^2$.

In Iteration 1, S_0 is by necessity P, and B_0 is $\{l, r\}$, so $\mathcal{P}_1 = \mathcal{R}_0$. The tree representation entries for the nullary symbol α will have no further effect on $\mathcal{R}_0$. On the other hand, we have that $\sum_{w \in [\bot][l]} \nu_2(\sigma)_{w,p}$ is nonzero only when $p = l$, which splits the block $\{l, r\}$. Seeing that ν is such that the block $\{L, R, \bot\}$ is only affected by itself, we know that $\mathcal{R}_1 = \{l\}^2 \cup \{r\}^2 \cup \{L, R, \bot\}^2$, is the sought bisimulation. This means that termination happens in Iteration 3, when $\mathcal{P}_3$ has been refined to the level of $\mathcal{R}_1$. ∎

Theorem 25. *Algorithm 2 returns the minimal wta $(M/\mathcal{P})$ with respect to backward bisimulation. Equivalently; $\mathcal{P}$ is the coarsest backward bisimulation on M.*

We now compute the time complexity of Alg. 2, using the same assumptions and notations as in Sect. 3. In addition, we denote by μ^{b}_L, where $L \subseteq \mathfrak{P}(Q^*)$, the part of the tree representation μ that contains entries of the form $\mu_k(\sigma)_{q_1 \cdots q_k, q}$, where $q_1 \cdots q_k$ is in L. The overall complexity of the Alg. 2 can be written as for Alg. 1, with $\textsc{Init}^{\mathrm{f}}$, $\textsc{Split}^{\mathrm{f}}_i$, and $\textsc{Aggregate}^{\mathrm{f}}$, replaced by $\textsc{Init}^{\mathrm{b}}$, $\textsc{Split}^{\mathrm{b}}_i$, and $\textsc{Aggregate}^{\mathrm{b}}$, respectively. By Lemma 26 we thus obtain Theorem 27.

Lemma 26. $\textsc{Init}^{\mathrm{b}}$ *is in* $O(rm + n)$*, whereas* $\textsc{Aggregate}^{\mathrm{b}}$ *is in* $O(m + n)$ *and* $\textsc{Split}^{\mathrm{b}}_i$ *is in* $O\big(r\,|\mu^{\mathrm{b}}_{\mathcal{L}_i(S_i)}|\big)$.

Theorem 27. *Algorithm 2 is in $O(rmn)$.*

As in the forward case, we present an optimisation of Alg. 2 for cancellative semirings that reduces the time complexity.

Lemma 28. *When the underlying semiring is cancellative, we can compute the relation $\mathcal{R}_{i+1}$ as $\mathcal{R}_i \setminus \mathit{split}^{\mathrm{b}}(\mathcal{L}_{i+1}(B_i))$ without effect on the overall algorithm.*

Theorem 29. *The optimisation of Alg. 2 is in $O\big(r^2 m \log n\big)$.*

5 Implementation

In this section we present experimental results obtained by applying an implementation (written in Perl) of Alg. 1 and Alg. 2 to the problem of *language modelling* in the natural language processing domain [19]. A language model is a formalism for determining whether a given sentence is in a particular language. Language models are particularly useful in applications of natural language and speech processing such as translation, transliteration, speech recognition, character recognition, etc., where transformation system output must be verified to be an appropriate sentence in the domain language. Typically they are formed by collecting subsequences of sentences over a large corpus of text and assigning probabilities to the subsequences based on their occurrence counts in the data. To obtain the probability of a sentence one multiplies the probability of subsequences together. It is thus useful to have a data structure for efficiently looking up many subsequences. As effective language models typically have many millions of unique subsequences, but there is considerable similarity between the subsequences, a compressed dictionary of subsequences seems to be a natural choice for such a data structure. A minimisation algorithm is particularly suited for building a compressed dictionary from uncompressed sequence input.

Recent research in natural language processing has focused on using tree-based models to capture syntactic dependencies in applications such as machine

Table 1. Reduction of states and rules by the bisimulation minimisation algorithms.

TREES	ORIGINAL		FORWARD		BACKWARD		CONVERGENCE	
	states	rules	states	rules	states	rules	states	rules
25	162	162	141	161	136	136	115	135
45	295	295	248	290	209	209	161	203
85	526	526	436	516	365	365	271	351
165	1087	1087	899	1054	672	672	468	623
305	1996	1996	1630	1924	1143	1143	735	1029

translation [20, 21]. We thus require a language model of trees, and the subsequences we will represent are subtrees. We prepared a data set by collecting 3-subtrees, i.e. all subtrees of height 3, from sentences taken from the Penn Treebank corpus of syntactically bracketed English news text [22], and collected observation statistics on these subtrees, which we stored as probabilities. In our experiments, we selected at random a subset of these subtrees and constructed an initial wta over the semiring $(\mathbb{R}_+, +, \cdot, 0, 1)$ by representing each 3-subtree in a single path, with an exit weight at the final state equal to the observed probability of the subtree. The sizes of the initial wta are noted in columns 2 and 3 of Table 1. We then performed a single iteration of the forward and backward variants, the results of which are noted in columns 4–7 of Table 1. On average the wta size, taken as $m + n$, is reduced by 10% of original by the forward algorithm and 34% by the backward algorithm. Reduction as a percentage of size by the backward algorithm grew with the size of the wta on this data set, e.g., the largest wta presented in Table 1 was reduced by 42.7%. In contrast, forward minimisation tended to reduce the size of the input by 10% for all wta in our test set. This performance is likely due to the nature of the experimental data used and may differ highly on, e.g., wta with a more densely packed μ, wta representing infinite languages, etc.

As noted in Sect. 1, further minimisation may be obtained by applying the two algorithms in an alternating manner. We found that for the wta in this experiment, two iterations beginning with backward or three iterations beginning with forward resulted in the smallest obtainable wta, the sizes of which are noted in the last two columns of Table 1. On average, the maximal minimisation reduced the size of the input wta by 45% and, as with backward minimisation, the reduction percentage grows with the size of the initial wta, to 55.8% for the largest wta in the sample set.

Acknowledgements The authors gratefully acknowledge the support of Kevin Knight. We appreciate the sample automata provided by Lisa Kaati and would like to thank Frank Drewes for proof-reading the manuscript. Finally, we would like to extend our thanks to the referees for their insightful comments.

References

1. Hopcroft, J.E.: An $n \log n$ algorithm for minimizing states in a finite automaton. In Kohavi, Z., ed.: Theory of Machines and Computations. Academic Press (1971)
2. Meyer, A.R., Stockmeyer, L.J.: The equivalence problem for regular expressions with squaring requires exponential space. In: Proc. 13th Annual Symp. Foundations of Computer Science, IEEE Computer Society (1972) 125–129
3. Gramlich, G., Schnitger, G.: Minimizing nfas and regular expressions. In: Proc. 22nd Int. Symp. Theoretical Aspects of Computer Science. Volume 3404 of LNCS., Springer Verlag (2005) 399–411
4. Milner, R.: A Calculus of Communicating Systems. Springer Verlag (1982)
5. Berstel, J., Reutenauer, C.: Recognizable formal power series on trees. Theoretical Computer Science **18**(2) (1982) 115–148
6. Gécseg, F., Steinby, M.: Tree Automata. Akadémiai Kiadó (1984)
7. Gécseg, F., Steinby, M.: Tree languages. In Rozenberg, G., Salomaa, A., eds.: Handbook of Formal Languages. Volume 3. Springer Verlag (1997) 1–68
8. Eilenberg, S.: Automata, Languages, and Machines—Volume A. Volume 59 of Pure and Applied Mathematics. Academic Press (1974)
9. Buchholz, P.: Bisimulation relations for weighted automata. unpublished (2007)
10. Högberg, J., Maletti, A., May, J.: Backward and forward bisimulation minimisation of tree automata. Technical Report ISI-TR-633, U. So. California (2007)
11. Kozen, D.: On the Myhill-Nerode theorem for trees. Bulletin of the EATCS **47** (1992) 170–173
12. Borchardt, B.: The Myhill-Nerode theorem for recognizable tree series. In: Proc. 7th Int. Conf. Developments in Language Theory. Volume 2710 of LNCS., Springer Verlag (2003) 146–158
13. Drewes, F., Vogler, H.: Learning deterministically recognizable tree series. J. Automata, Languages and Combinatorics (2007) to appear.
14. Högberg, J., Maletti, A., May, J.: Bisimulation minimisation of weighted tree automata. Technical Report ISI-TR-634, U. So. California (2007)
15. Paige, R., Tarjan, R.: Three partition refinement algorithms. SIAM Journal on Computing **16**(6) (1987) 973–989
16. Abdulla, P.A., Kaati, L., Högberg, J.: Bisimulation minimization of tree automata. In: Proc. 11th Int. Conf. Implementation and Application of Automata. Volume 4094 of LNCS., Springer Verlag (2006) 173–185
17. Borchardt, B.: The Theory of Recognizable Tree Series. Akademische Abhandlungen zur Informatik. Verlag für Wissenschaft und Forschung (2005)
18. Abdulla, P.A., Kaati, L., Högberg, J.: Bisimulation minimization of tree automata. Technical Report UMINF 06.25, Umeå University (2006)
19. Jelinek, F.: Continuous speech recognition by statistical methods. Proc. IEEE **64**(4) (1976) 532–557
20. Galley, M., Hopkins, M., Knight, K., Marcu, D.: What's in a translation rule? In: Proc. HLT-NAACL. (2004) 273–280
21. Yamada, K., Knight, K.: A syntax-based statistical translation model. In: Proc. ACL, Morgan Kaufmann (2001) 523–530
22. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of english: The Penn treebank. Comp. Linguistics **19**(2) (1993) 313–330