Bisimulation Minimisation of Weighted Tree Automata

Johanna Högberg

Department of Computing Science, Umeå University
S–90187 Umeå, Sweden

Andreas Maletti

Faculty of Computer Science, Technische Universität Dresden
D–01062 Dresden, Germany

Jonathan May

Information Sciences Institute, University of Southern California
Marina Del Rey, CA 90292

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 of the input automaton). We demonstrate that weighted forward bisimulation minimisation can be used to minimise deterministic all-accepting weighted tree automata. Finally, we discuss implementations of these algorithms on a typical task in natural language processing.

1 Introduction

1.1 Minimisation algorithms

By the Myhill-Nerode theorem there exists, for every regular string language $\mathcal{L}$, a unique – up to isomorphism – minimal deterministic finite automaton (dfa) that recognises $\mathcal{L}$. In the beginning of the 1970's, several minimisation algorithms for finite automata were known, though all had a computational complexity of $O(n^2)$ or worse, where n is the number of states of the input automaton. It was then something of a breakthrough when John Hopcroft presented an $O(n \log n)$ algorithm for dfa in [18]. This bound, which to date has not been improved, was obtained by partitioning the state space of the input automaton through a "process the smaller half" strategy.

It was well-known at the time that there is, in general, no unique minimal nondeterministic finite automaton (nfa) that recognises a given regular language, but hope of an efficient algorithm capable of deriving *a* minimal automaton still flourished. The search for such an algorithm was cancelled two years later when Alain Meyer and Larry Stockmeyer proved that minimisation of nfa is PSPACE complete [30]. As it turned out, the minimisation problem for nfa with n states cannot even be efficiently approximated within the factor $o(n)$, unless P = PSPACE [16]. 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 their input automata with respect to *bisimularity* are examples of the latter approach. The concept of bisimularity was introduced in 1980 by the British computer scientist Robin Milner as a formal tool for investigating labelled transition systems. Simply put, two transition systems are bisimulation equivalent if their behaviour – in response to a given 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.

1.2 Weighted tree automata

Weighted tree automata [4, 23, 6] generalise both finite tree automata [14, 15] and weighted string automata: classical tree automata can be seen as weighted tree automata with weights in the Boolean semiring, i.e. a transition has weight *true* if it is present, and *false* otherwise. Let us shortly recall the model of [6]. A *weighted tree automaton* (for short: wta) is a tuple $(Q, \Sigma, \mathcal{A}, F, \mu)$ where Q is the finite set of *states*; Σ is the ranked alphabet of *input symbols*; $\mathcal{A} = (A, +, \cdot, 0, 1)$ is the semiring of *coefficients* (or alternatively: *weights*); $F \colon Q \to A$ is the *final weight distribution*; and $\mu = (\mu_k)_{k \geq 0}$ with $\mu_k \colon \Sigma_{(k)} \to A^{Q^k \times Q}$ is the *tree representation.* The tree representation assigns to each k-ary input symbol a $(Q^k \times Q)$-indexed matrix over the coefficients. Intuitively, the weight of a transition $\sigma(q_1, \ldots, q_k) \to q$ is $\mu_k(\sigma)_{q_1 \cdots q_k, q}$. If the weight is 0, then this essentially means that the transition is absent.

The wta processes an input tree of the form $\sigma[t_1, \ldots, t_k]$ by first processing the subtrees $t_1, \ldots, t_k$ and then processing the remaining symbol σ. The such obtained weights are combined with the semiring multiplication and nondeterminism is resolved with the addition; i.e., we sum over all combinations of states that the wta could be in after processing $t_1, \ldots, t_k$. Formally, the mapping $h_\mu \colon \mathrm{T}_\Sigma \to A^Q$ is defined as follows:

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

The semantics of the wta is a (formal) tree series, which is a mapping of type $\mathrm{T}_\Sigma \to A$. The semantics of the wta assigns to an input tree t the F-weighted sum of the components of $h_\mu(t)$. Thus, the coefficient of t is $\sum_{q \in Q} F(q) \cdot h_\mu(t)_q$. A tree series that can be computed by a wta is called *recognisable.*

The minimisation of the representation of a recognisable tree series over fields is already considered in [9, 8], but a formal complexity analysis is missing. In this report, we will consider procedures for arbitrary commutative semirings. Since minimisation is PSPACE complete in certain commutative semirings, we only consider minimisation with respect to a bisimulation relation.

1.3 Forward vs. backward bisimulation

The idea behind bisimulation minimisation is to discover and unify states that in some sense exhibit the same behaviour, thus freeing the input automaton of redundancy. This implies in effect a search for the coarsest relation on the state space that meets the local conditions of the bisimulation relation that we are interested in.

One type of bisimulation, called *forward bisimulation* in [10, 17], 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 [22] for a tree language or even in the Myhill-Nerode congruence [5] 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 $a^{-1} \cdot (\mathcal{S}, C[t]) = b^{-1} \cdot (\mathcal{S}, C[u])$. In this expression, the coefficients a and b can be understood as the weights of t and u, respectively. Both sides of the previous equation can be understood as futures; the futures $\mathcal{S}_t$ and $\mathcal{S}_u$ are given for every context C by $(\mathcal{S}_t, C) = a^{-1} \cdot (\mathcal{S}, C[t])$ and $(\mathcal{S}_u, C) = b^{-1} \cdot (\mathcal{S}, C[u])$. Roughly speaking, in $\mathcal{S}_t$ a context is assigned the weight of $C[t]$ in $\mathcal{S}$ with the weight of t cancelled out. In other words, trees t and u are equal if and only if their futures $\mathcal{S}_t$ and $\mathcal{S}_u$ coincide. In contrast to the Myhill-Nerode congruence, a forward bisimulation requires a local condition on the tree representation. This local condition is strong enough, so that the equivalence of the futures of bisimilar states is enforced. On the other hand, the condition is not too strong which is shown by the fact that, on a deterministic all-accepting wta M over a field, minimisation via forward bisimulation yields the unique (up to isomorphism) minimal deterministic all-accepting wta that recognises the same tree series as M (see Theorem 3.12).

Let us look at the local condition for forward bisimilar states in more detail. Suppose that we have a sequence $(q_1, \ldots, q_k)$ of states and a state p_i that is supposed to be bisimilar to some q_i (with $1 \leq i \leq k$). First, the final weight of p_i should be exactly the final weight of q_i (otherwise the futures would not coincide with respect to the context $\square$). Second, bisimilar states are indistinguishable, so we consider a group D of bisimilar states next. If the transition weights of σ-transitions from $(q_1, \ldots, q_k)$ to some state q in D sum to a, then the weights of σ-transitions from $(q_1, \ldots, q_{i-1}, p_i, q_{i+1}, \ldots, q_k)$ to states of D should also sum to a. The states p_i and q_i can only have the same futures if in all circumstances they can process the same input symbol such that bisimilar states p and q (which have the same future) are reached with the same weight.

Forward bisimulation minimisation is effective on (bottom-up) deterministic wta. In fact, we show that minimisation via forward bisimulation on a deterministic *all-accepting* [12] wta M over a field computes the unique (up to isomorphism) minimal deterministic all-accepting wta [12] recognising the same tree series as M (see Theorem 3.12). More importantly, it is shown in Corollary 3.33 that the (asymptotic) time-complexity of our minimisation algorithm on such a wta is $O(m \log n)$ where m is the number of transitions and n is the number of states. In [12, Lemma 3.7] a Myhill-Nerode theorem for these wta is shown (which is essentially an instance of [7, Theorem 7.4.1]), but the complexity of minimisation is not discussed.

The other type of bisimulation we will consider is called *backward bisimulation* in [10, 17]. 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 Section 2). Let us illustrate the local condition for backward bisimulation. Suppose that the states p and q are bisimilar. Again we group transitions by blocks $D_1, \dots, D_k$ of bisimilar states. For p and q to be bisimilar, it should be true that whenever the weights of σ-transitions from states of $D_1 \times \cdots \times D_k$ to p sum to a, then the weights of σ-transitions from $D_1 \times \cdots \times D_k$ to q should also sum to a. Of course, the condition should also hold with the roles of p and q exchanged. Since bisimilar states have the same past and the transitions into p and q have the same weight, the states p and q also have the same past.

1.4 Algorithms

As we shall see, the standard minimisation algorithm for deterministic tree automata [11] can be generalised to work with respect to either type of bisimulation: rather than discriminating between states based on their behaviour with respect to a right congruence, we ascertain that the local conditions of a bisimulation relation are fulfilled. Unfortunately, the counting argument used in [34] and later in [17] is no longer applicable: it was devised for the Boolean semiring and does not generalise. For unrestricted semirings, the resulting algorithms thus run in time $O(rmn)$, where r is the maximum rank of the input alphabet, m is the size of the transition table, and n is the number of states. However, in the special case when the underlying semiring is additively cancellative, i.e. when $a + b = a + c$ implies that $b = c$, this time bound can be reduced to $O(rm \log n)$ for forward bisimulation, and $O(r^2 m \log n)$ for backward bisimulation, using Hopcroft's "process the smaller half" strategy. When string languages are considered, r is equal to one and can thus be omitted from the complexity expressions.

Our general approach is iterative. Initially, all states are assumed to be bisimilar, and are for this reason grouped together in the one-block equivalence relation $\mathcal{R}_0$. If we then discover pairs of states in $\mathcal{R}_i$ that are provably not bisimilar, then we remove these to derive a finer relation $\mathcal{R}_{i+1}$. This refinement process continues until all offending pairs have been cleared, and yields upon termination the coarsest bisimulation relation on the state space of the input automaton. As indicated by Lemma 3.8, one could also work in the opposite direction by starting with the finest possible bisimulation, i.e. the identity, and then successively relaxing it. Both possibilities of constructing the coarsest bisimulation are discussed in [10] for string automata. In particular, the method of deriving the coarsest forward bisimulation that is suggested in Theorem 3.7 of [10] coincides with the non-optimised version of our forward bisimulation algorithm when the input signature is monadic.

The type of bisimulation introduced in [1] can in retrospect be seen as a combination of backward and forward bisimulation. Containing the restrictions of both, this hybrid is less efficient than backward bisimulation when applied to the minimisation of nondeterministic tree automata, although it is just as expensive to calculate, and unlike forward bisimulation it does not yield the standard algorithm when applied to deterministic tree automata. The pair of algorithms presented in this paper thus supersedes that of [1].

1.5 Applications

Automata theory in general has been of great use to the field of natural language processing, as the ability to represent linguistic functions in a formal computational model enables sound reasoning and clean computation. The representation of formalisms that operate on words and phrases as chains of nondeterministic acceptors and transducers has been shown to be useful for rapid system construction and greater understanding and modularity (e.g. [24, 3, 38, 36]). In more recent work, however, there has been a desire to use models that consider syntactic structure as an elementary unit, and thus tree automata are a more natural fit than the previously exploited string-based automata. Work has progressed on transformation of elaborate syntax-based natural language systems into simple sequences of tree acceptors and transducers [21]. And as the development of efficient general algorithms for string automata [31, 33] and their subsequent incorporation in toolkits [32] made the rapidly-constructed string-based systems possible, so too is there ongoing development of tree-based elevations of these efficient algorithms [29] with the goal of constructing a tree automata toolkit [28] that will benefit from the contributions of this work.

Both minimisation algorithms have been implemented as prototypes in Perl. There are advantages that support having two types of bisimulation minimisation. First, forward and backward bisimulation minimisation only yield a minimal wta with respect to the corresponding notion of bisimulation. Thus applying forward and backward bisimulation minimisation in an alternating fashion might yield an even smaller wta than just the application of a single forward or backward bisimulation minimisation. Since both minimisation procedures are very efficient, this approach also works in practice. For the problem of minimisation of lookup tables for *tree language modelling*, discussed in Section 6, 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.

Second, in certain domains only one type of bisimulation minimisation will be effective. Let us, for example, consider deterministic wta. We will provide some arguments that show that backward bisimulation will be ineffective on deterministic wta (provided that the wta has no useless states). For any two backward bisimilar states, their recognised series coincide (see Corollary 4.7). However, in a deterministic wta we have the property that, for any two states, the supports of the series recognised in those states do not overlap. This yields that every deterministic wta without useless states is minimal with respect to backward bisimulation; thus backward bisimulation minimisation is ineffective in that domain. This also means that the alteration technique is useless for deterministic devices, because two applications of forward bisimulation do not accomplish more than one, and backward bisimulation has no effect at all.

2 Preliminaries

Sets, numbers, and relations 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 membership $d_i \in D_i$ for all $i \in [1, k]$ as $d_1 \cdots d_k \in D_1 \cdots D_k$. Finally, we write ε for the empty word.

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 the equivalence relation $\mathcal{R}$ induces a partition $(S/\mathcal{R}) = \{[s] \mid s \in S\}$ of S.

Algebraic structures A *monoid* is a set A together with a binary operation $\cdot$ from $A \times A$ to A

and an element 1 in A that satisfy the following two axioms:

[M1] The operation $\cdot$ is *associative* in that for every three elements a, b, and c in A, it holds that $(a \cdot b) \cdot c$ is equal to $a \cdot (b \cdot c)$.

[M2] The element 1 is the *neutral* element with respect to $\cdot$; i.e., we have $a \cdot 1 = 1 \cdot a = a$, for all $a \in A$.

A monoid $(A, \cdot, 1)$ is *commutative* if $a \cdot b = b \cdot a$, for all a, b in A.
Moreover, a *semiring* is a tuple $(A, +, \cdot, 0, 1)$ where

[SR1] $(A, +, 0)$ is a commutative monoid and

[SR2] $(A, \cdot, 1)$ is a monoid.

[SR3] For every a, b, and c in A, it holds that $(a + b) \cdot c$ equals $(a \cdot c) + (b \cdot c)$ and $a \cdot (b + c)$ equals $(a \cdot b) + (a \cdot c)$, i.e. the multiplicative operation *distributes* over the additive.

[SR4] For every a in A, we have that $a \cdot 0 = 0 \cdot a = 0$. In other words, the element 0 is *absorptive.*

[SR5] Finally, 0 and 1 are distinct elements.

Operations with multiplicative symbols (like $\cdot$) are generally assumed to bind stronger than operations that are written additively (like $+$). Hence we interpret $a + b \cdot c$ as $a + (b \cdot c)$. We now list a number of properties that a semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$ may or may not have. The semiring $\mathcal{A}$ is said to be ...

- *commutative* if its multiplicative monoid $(A, \cdot, 1)$ is also a commutative monoid,
- *zero-sum free* if $a + b = 0$ implies that $a = b = 0$ for all elements a and b,
- *zero-divisor free* if $a \cdot b = 0$ implies that $a = 0$ or $b = 0$ for all elements a and b,
- *additively cancellative* if $a + b = a + c$ implies that $b = c$ for all elements a, b, and c,
- *multiplicatively cancellative* if $a \neq 0$, together with $a \cdot b = a \cdot c$ or $b \cdot a = c \cdot a$, implies that $b = c$ for all elements a, b, and c,
- *a ring* if for every element a there exists an *additive inverse* $(-a) \in A$ such that $a + (-a) = 0$,
- *a semifield* if for every nonzero element a there exists an *multiplicative inverse* $a^{-1} \in A$ such that $a \cdot a^{-1} = 1$, and
- *a field* if $\mathcal{A}$ is a ring and a semifield.

Tree automata 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 $f\, t_1 \cdots t_k$ is in T_Σ for every f in $\Sigma_{(k)}$ and all $t_1, \ldots, t_k$ in T_Σ. To improve readability 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 the 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 $\mathcal{A}\langle\langle \mathrm{T}_\Sigma \rangle\rangle$. We also introduce an alternative notation for a tree series $\mathcal{S} \in \mathcal{A}\langle\langle \mathrm{T}_\Sigma \rangle\rangle$: the image $\mathcal{S}(t) \in A$ of a tree $t \in \mathrm{T}_\Sigma$ is called the *coefficient of* t and, analogously to power series in analysis, the coefficient of t is denoted by $(\mathcal{S}, t)$. The tree series $\mathcal{S}$ can thus be rewritten as the formal sum $\sum_{t \in \mathrm{T}_\Sigma} (\mathcal{S}, t)\ t$. A tree $t \in \mathrm{T}_\Sigma$ is said to be in the *support* of $\mathcal{S}$, written *supp*$(\mathcal{S})$, if and only if $(\mathcal{S}, t)$ is non-zero.

A *weighted tree automaton* M (for short: wta) is a tuple $(Q, \Sigma, \mathcal{A}, F, \mu)$, where

- Q is a finite nonempty set of *states*;
- Σ is a ranked alphabet (aka. the *input alphabet*);
- $\mathcal{A} = (A, +, \cdot, 0, 1)$ is a semiring;
- $F \in A^Q$ is a *final weight distribution*; and
- $\mu = (\mu_k)_{k \in \mathbb{N}}$ with $\mu_k : \Sigma_{(k)} \to A^{Q^k \times Q}$, $k \in \mathbb{N}$, is a *tree representation.*

The *initial algebra semantics* of the wta M is determined by the mapping h_μ that takes T_Σ to A^Q and is given by

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

for every symbol $\sigma \in \Sigma_{(k)}$, $q \in Q$, and trees $t_1, \dots, t_k \in \mathrm{T}_\Sigma$. The tree series *recognised by* M, denoted by $\| M \|$, is defined by

$$(\| M \|, t) = \sum_{q \in Q} F_q \cdot h_\mu(t)_q$$

for every tree $t \in \mathrm{T}_\Sigma$. A tree series is *recognisable* if there exists a wta that recognises it.

3 Forward bisimulation

3.1 Foundation

In this section we introduce the concept of a *forward bisimulation.* Roughly speaking, a forward bisimulation is an equivalence relation on the states of a wta such that equivalent states react equivalently to future inputs. We enforce this behaviour with only a local condition on the tree representation and a condition on the final weight distribution.

First we introduce the notion of a *context.* A context is a string of states that has a placeholder (the special symbol $\square$) at exactly one position. Essentially, the forthcoming definition of forward bisimulation uses such contexts where the placeholder is replaced by a particular state.

Definition 3.1 Let Q be a set such that $\square \notin Q$. The set $C^Q_{(k)}$ of *contexts (over Q)* is given by

$$\{w \in (Q \cup \{\square\})^k \mid w \text{ contains } \square \text{ exactly once}\} \enspace ,$$

and for every $c \in C^Q_{(k)}$ and $q \in Q$ we write $c[\![q]\!]$ to denote the word that is obtained from c by replacing the special symbol $\square$ with q. □

Henceforth, we assume that the special symbol □ occurs in no set of states of any wta. By a simple renaming of states this property can easily be guaranteed.

Definition 3.2 (cf. [10, Definition 3.1]) Let $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ be a wta, and let $\mathcal{R} \subseteq Q \times Q$ be an equivalence relation on Q. We say that $\mathcal{R}$ is a *forward bisimulation on M* if for every (p, q) in $\mathcal{R}$

(i) $F(p) = F(q)$; and

(ii) for every symbol σ in $\Sigma_{(k)}$, block D in $(Q/\mathcal{R})$, and context c of $C^Q_{(k)}$ we have that

$$\sum_{r \in D} \mu_k(\sigma)_{c[p],r} = \sum_{r \in D} \mu_k(\sigma)_{c[q],r} \enspace . \qquad \square$$

We note that Condition (ii) of Definition 3.2 is trivially fulfilled for nullary symbols σ. Let us illustrate the definition of forward bisimulation on an example. We will define a tree series ZIGZAG: $T_\Delta \to \mathbb{N}$ and show that it is recognisable over the semiring of the natural numbers. We then guess an equivalence relation $\mathcal{P}$ on the states of the constructed wta N and validate that $\mathcal{P}$ is indeed a forward bisimulation on N.

Example 3.3 Let $\Delta = \Delta_{(0)} \cup \Delta_{(2)}$ be the ranked alphabet where $\Delta_{(0)} = \{\alpha\}$ and $\Delta_{(2)} = \{\sigma\}$. The mapping ZIGZAG from T_Δ to $\mathbb{N}$ is defined for every t_1, t_2, and t_3 in T_Δ by

$$\begin{aligned} \textsc{zigzag}(\alpha) &= 1 \\ \textsc{zigzag}(\sigma[\alpha, t_2]) &= 2 \\ \textsc{zigzag}(\sigma[\sigma[t_1, t_2], t_3]) &= 2 + \textsc{zigzag}(t_2) \enspace . \end{aligned}$$

Consider now the wta $N = (P, \Delta, \mathbb{N}, G, \nu)$ with the underlying semiring $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$ of natural numbers (with the usual addition and multiplication operations) that recognises ZIGZAG. The remaining components of N are given by $P = \{l, r, L, R, \bot\}$, $G(l) = G(L) = 1$ and $G(p) = 0$ for every $p \in \{r, R, \bot\}$, and

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

Note that all remaining entries in ν are 0. The wta N is displayed in Figure 1 (in the representation of [7]). A straightforward induction shows that N does indeed recognise ZIGZAG. Let us consider the equivalence relation $\mathcal{P} = \{l, L\}^2 \cup \{r, R\}^2 \cup \{\bot\}^2$ on P. We argue 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 3.2. We only demonstrate the computation on one example, namely for the symbol σ, the states l and L, the context $\bot\square$, and the block $\{r, R\}$.

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

Overall, it can be verified that $\mathcal{P}$ is a forward bisimulation on N. □

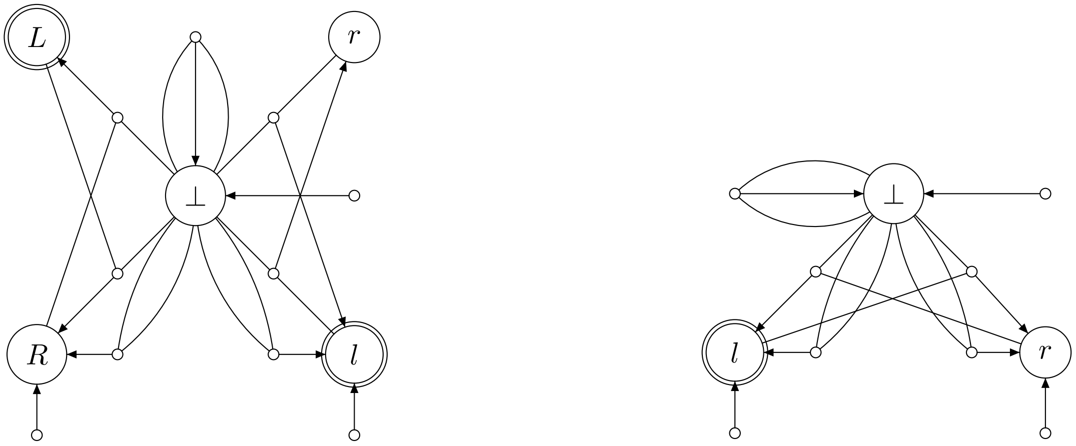


Figure 1: The wta of Example 3.3(left) and the aggregated wta of Example 3.5(right); every binary transition is labelled with $\sigma/1$ and every nullary transition with $\alpha/1$ and final weights are 1 for doubly-circled states and 0 otherwise.

We identify bisimilar states in order to reduce the size of the wta. For the moment, we suppose that a wta $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ and a forward bisimulation $\mathcal{R}$ on M are given. In Section 3.2 we show how a particular, namely the coarsest, forward bisimulation on M can be computed efficiently. Next we present how to reduce the size of M with the help of $\mathcal{R}$. In essence, we construct a wta $(M/\mathcal{R})$ that uses only one state per equivalence class of $\mathcal{R}$.

Definition 3.4 (cf. [10, Definition 3.3]) The *(forward) aggregated wta* $(M/\mathcal{R})$ [10] *(with respect to M and $\mathcal{R}$)* is the wta $(Q', \Sigma, \mathcal{A}, F', \mu')$ given by

- $Q' = (Q/\mathcal{R})$;
- $F'([q]) = F(q)$ for every state q of Q; and
- for every symbol $\sigma \in \Sigma_{(k)}$, word $q_1 \cdots q_k \in Q^k$, and block $D \in (Q/\mathcal{R})$

$$\mu'_k(\sigma)_{[q_1]\cdots[q_k],D} = \sum_{r \in D} \mu_k(\sigma)_{q_1 \cdots q_k, r} \enspace .$$

The wta $(M/\mathcal{R})$ is well-defined because $\mathcal{R}$ is a forward bisimulation on M. □

The above definition shows how states are collapsed. It is clear that the reduced wta $(M/\mathcal{R})$ will have as many states as there are equivalence classes with respect to $\mathcal{R}$, and thus never more states than M. Let us present an example of an aggregated wta.

Example 3.5 Recall the wta N and the forward bisimulation $\mathcal{P}$ of Example 3.3. Let us compute the wta $(N/\mathcal{P}) = (P', \Delta, \mathbb{N}, G', \nu')$. Clearly, we obtain the states $P' = \{[l], [r], [\bot]\}$, the final weights $G'([l]) = 1$ and $G'([r]) = G'([\bot]) = 0$ and the tree representation entries

$$\begin{array}{rcl} 1 &=& \nu'_0(\alpha)_{\varepsilon,[l]} \;=\; \nu'_0(\alpha)_{\varepsilon,[r]} \;=\; \nu'_0(\alpha)_{\varepsilon,[\bot]} \\ 1 &=& \nu'_2(\sigma)_{[r]\,[\bot],[l]} \;=\; \nu'_2(\sigma)_{[\bot]\,[l],[r]} \\ 1 &=& \nu'_2(\sigma)_{[\bot]\,[\bot],[l]} \;=\; \nu'_2(\sigma)_{[\bot]\,[\bot],[r]} \;=\; \nu'_2(\sigma)_{[\bot]\,[\bot],[\bot]} \enspace . \end{array}$$

All remaining entries in ν' are 0. A graphical representation of $(N/\mathcal{P})$ can be found in Figure 1. Thus we now have only 3 states and 8 transitions instead of the 5 states and 10 transitions we used in the wta of Example 3.3. □

Thus our reduction technique seems to work. Of course, 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 some collapsed state of $(M/\mathcal{R})$ works like the combination of its constituents in M.

Lemma 3.6 (cf. [10, Theorem 3.1]) Let $(M/\mathcal{R}) = (Q', \Sigma, \mathcal{A}, F', \mu')$. Then, for every tree $t \in \mathrm{T}_\Sigma$ and block $D \in (Q/\mathcal{R})$,

$$h_{\mu'}(t)_D = \sum_{q \in D} h_\mu(t)_q \enspace .$$

Proof We prove the statement by induction on t. Let $t = \sigma[t_1, \dots, t_k]$ for some symbol $\sigma \in \Sigma_{(k)}$ and sequence of subtrees $t_1, \dots, t_k \in \mathrm{T}_\Sigma$. For every index $i \in [1, k]$ and $D_i \in (Q/\mathcal{R})$ we have $h_{\mu'}(t_i)_{D_i} = \sum_{q_i \in D_i} h_\mu(t_i)_{q_i}$ by induction hypothesis. With this in mind, we reason as follows.

$$\begin{aligned}
&h_{\mu'}(\sigma[t_1, \dots, t_k])_D \\
={}& \sum_{D_1, \dots, D_k \in (Q/\mathcal{R})} \mu'_k(\sigma)_{D_1 \cdots D_k, D} \cdot \prod_{i \in [1,k]} h_{\mu'}(t_i)_{D_i} \\
={}& \sum_{D_1, \dots, D_k \in (Q/\mathcal{R})} \mu'_k(\sigma)_{D_1 \cdots D_k, D} \cdot \prod_{i \in [1,k]} \Bigl(\sum_{q_i \in D_i} h_\mu(t_i)_{q_i} \Bigr) && \text{(by induction hypothesis)} \\
={}& \sum_{\substack{D_1, \dots, D_k \in (Q/\mathcal{R}), \\ q_1 \cdots q_k \in D_1 \cdots D_k}} \mu'_k(\sigma)_{D_1 \cdots D_k, D} \cdot \prod_{i \in [1,k]} h_\mu(t_i)_{q_i} \\
={}& \sum_{\substack{D_1, \dots, D_k \in (Q/\mathcal{R}), \\ q_1 \cdots q_k \in D_1 \cdots D_k}} \Bigl(\sum_{q \in D} \mu_k(\sigma)_{q_1 \cdots q_k, q} \Bigr) \cdot \prod_{i \in [1,k]} h_\mu(t_i)_{q_i} && \text{(by definition of } \mu'\text{)} \\
={}& \sum_{q \in D} \Bigl(\sum_{q_1, \dots, q_k \in Q} \mu_k(\sigma)_{q_1 \cdots q_k, q} \cdot \prod_{i \in [1,k]} h_\mu(t_i)_{q_i} \Bigr) \\
={}& \sum_{q \in D} h_\mu(\sigma[t_1, \dots, t_k])_q
\end{aligned}$$

■

The final step establishes that $\|(M/\mathcal{R})\| = \|M\|$ using the previous lemma. Consequently, any wta obtained as $(M/\mathcal{R})$ with $\mathcal{R}$ a forward bisimulation on M recognises the same tree series as the wta M.

Theorem 3.7 (cf. [10, Theorem 3.1]) $\|(M/\mathcal{R})\| = \|M\|$.

Proof Let $(M/\mathcal{R}) = (Q', \Sigma, \mathcal{A}, F', \mu')$, and let $t \in \mathrm{T}_\Sigma$ be arbitrary. Then

$$\begin{aligned}
(\|(M/\mathcal{R})\|, t) &= \sum_{D \in (Q/\mathcal{R})} F'(D) \cdot h_{\mu'}(t)_D \\
&= \sum_{D \in (Q/\mathcal{R})} F'(D) \cdot \Bigl(\sum_{q \in D} h_\mu(t)_q \Bigr) && \text{(by Lemma 3.6)}
\end{aligned}$$

$$= \sum_{D \in (Q/\mathcal{R}), q \in D} F(q) \cdot h_\mu(t)_q \qquad \text{(by definition of } F'\text{)}$$

$$= \sum_{q \in Q} F(q) \cdot h_\mu(t)_q \; = \; (\|M\|, t) \; . \qquad \blacksquare$$

The coarser the forward bisimulation $\mathcal{R}$ on M, the smaller the aggregated wta $(M/\mathcal{R})$. Our aim is thus to find the coarsest forward bisimulation on M. An efficient algorithm that computes this bisimulation is presented in Section 3.2. Here we only show that a unique coarsest forward bisimulation on M exists.

Lemma 3.8 (cf. [10, Theorem 3.5]) Let $\mathcal{R}$ and $\mathcal{P}$ be forward bisimulations on M. Then there exists a forward bisimulation $\mathcal{R}'$ on M such that $\mathcal{R} \cup \mathcal{P} \subseteq \mathcal{R}'$.

Proof Let $\mathcal{R}'$ be the smallest equivalence containing $\mathcal{R} \cup \mathcal{P}$. We now show that $\mathcal{R}'$ is a forward bisimulation on M. Let $(p, q) \in \mathcal{R}'$. Thus there exist an integer $n \in \mathbb{N}$ and

$$(p_1, p_2), (p_2, p_3), \ldots, (p_{n-2}, p_{n-1}), (p_{n-1}, p_n) \in \mathcal{R} \cup \mathcal{P}$$

such that $p_1 = p$ and $p_n = q$. Thus $F(p) = F(p_1) = \cdots = F(p_n) = F(q)$. Now we move on to Condition (ii) of Definition 3.2. Let σ be a symbol of $\Sigma_{(k)}$, D a block in $(Q/\mathcal{R}')$, and c a context in $C^Q_{(k)}$. Clearly, the block D is a union of blocks of $(Q/\mathcal{R})$ as well as a union of blocks of $(Q/\mathcal{P})$. Thus Condition (ii) of Definition 3.2 is trivially met for all $(p_i, p_{i+1}) \in \mathcal{R} \cup \mathcal{P}$. Now we verify the condition for p and q.

$$\sum_{r \in D} \mu_k(\sigma)_{c[p_1], r} \; = \; \sum_{r \in D} \mu_k(\sigma)_{c[p_2], r} \; = \; \ldots \; = \; \sum_{r \in D} \mu_k(\sigma)_{c[p_{n-1}], r} \; = \; \sum_{r \in D} \mu_k(\sigma)_{c[p_n], r} \qquad \blacksquare$$

The above lemma does construct a coarser forward bisimulation given two forward bisimulations, but it does not yield an efficient algorithm that computes the coarsest forward bisimulation on M. However, since there are only finitely many equivalence relations on Q, there exists a unique coarsest forward bisimulation on M.

Theorem 3.9 (cf. [10, Theorem 3.2]) There exists a coarsest forward bisimulation $\mathcal{P}$ on M, and the identity is the only forward bisimulation on $(M/\mathcal{P})$.

Proof The existence of the coarsest forward bisimulation $\mathcal{P}$ follows from Lemma 3.8. The second part of the statement is proved by contradiction, so suppose that $(M/\mathcal{P}) = (Q', \Sigma, \mathcal{A}, F', \mu')$ admits a non-trivial forward bisimulation $\mathcal{R}'$. Whenever we write $[p]$ (with $p \in Q$) in the remaining proof, we mean $[p]_\mathcal{P}$. Let $\mathcal{P}' = \{(p, q) \mid ([p], [q]) \in \mathcal{R}'\}$. Obviously, $\mathcal{P}'$ is an equivalence relation on Q. We claim that $\mathcal{P}'$ is a forward bisimulation on M. This would contradict the assumption that $\mathcal{P}$ is the coarsest forward bisimulation on M because $\mathcal{P} \subseteq \mathcal{P}'$ and since $\mathcal{R}'$ is not the identity it follows that $\mathcal{P} \subset \mathcal{P}'$.

Let $(p, q) \in \mathcal{P}'$. We start with Condition (i) of Definition 3.2. Since $([p], [q]) \in \mathcal{R}'$ we have $F'([p]) = F'([q])$. Moreover, by Definition 3.4 we also have $F(p) = F'([p]) = F'([q]) = F(q)$. It remains to validate Condition (ii). To this end, let $\sigma \in \Sigma_{(k)}$ be a symbol, $c = q_1 \cdots q_{i-1} \Box q_{i+1} \cdots q_k$ be a context of $C^Q_{(k)}$ for some $q_1, \ldots, q_k \in Q$ and $i \in [1, k]$, and $D \in (Q/\mathcal{P}')$ be a block. Clearly, the block D is a union $D_1 \cup \cdots \cup D_n$ of pairwise different blocks $D_1, \ldots, D_n \in (Q/\mathcal{P})$. Moreover, $D' = \{D_1, \ldots, D_n\}$ is a block in $(Q'/\mathcal{R}')$. Finally, let $c' = [q_1] \cdots [q_{i-1}] \Box [q_{i+1}] \cdots [q_k]$, which is a context of $C^{Q'}_{(k)}$.

$$\begin{aligned}
\sum_{r \in D} \mu_k(\sigma)_{c[p],r} &= \sum_{D'' \in D'} \Big(\sum_{r \in D''} \mu_k(\sigma)_{c[p],r} \Big) \\
&= \sum_{D'' \in D'} \mu'_k(\sigma)_{c'[[p]],D''} && \text{(by Definition 3.4)} \\
&= \sum_{D'' \in D'} \mu'_k(\sigma)_{c'[[q]],D''} && \text{(because } ([p],[q]) \in \mathcal{R}'\text{)} \\
&= \sum_{D'' \in D'} \Big(\sum_{r \in D''} \mu_k(\sigma)_{c[q],r} \Big) = \sum_{r \in D} \mu_k(\sigma)_{c[q],r} && \text{(by Definition 3.4)}
\end{aligned}$$

Thus $\mathcal{P}'$ is a strictly coarser forward bisimulation on M than $\mathcal{P}$, which contradicts our assumption. Hence the identity is the only forward bisimulation on $(M/\mathcal{P})$. ■

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. It only admits the trivial forward bisimulation, and thus any further forward aggregated wta will have as many states as $(M/\mathcal{P})$.

We close this section with an analysis of forward bisimulation on (bottom-up) deterministic all-accepting [12] wta. The rest of this section can be skipped on first reading. Our goal is to prove that forward bisimulation minimisation coincides with classical minimisation and yields the unique (up to isomorphism) minimal deterministic all-accepting wta recognising the given tree series. Before we can discuss any details, let us formally define deterministic and complete weighted tree automata and thereafter the all-accepting property [12]. Note that our notions coincide with the notions for bottom-up weighted tree automata (see, e.g., [7]).

Definition 3.10 (cf. [7, Definition 4.1.1]) We say that M is *deterministic* (respectively, *complete*), if for every symbol σ in $\Sigma_{(k)}$, and word $q_1 \cdots q_k$ of states in Q there exists at most (respectively, at least) one state q of Q such that $\mu_k(\sigma)_{q_1 \cdots q_k, q} \neq 0$. □

It is an important observation that $(M/\mathcal{R})$ is deterministic whenever M is so. A similar observation can be made for completeness, however, we additionally need that the underlying semiring is zero-sum free. Finally, $(M/\mathcal{R})$ is deterministic and complete provided that M is so.

It is known that, for every tree series $\mathcal{S}$ that is recognised by some deterministic *all-accepting* [12, Section 3.2] wta over a multiplicatively cancellative and commutative semiring, there exists a unique (up to isomorphism) minimal deterministic and complete all-accepting wta that recognises $\mathcal{S}$ [12, 25]. First we need to recall the concept of a dead state. Let P be a subset of the state space Q. We call the states of P *dead states* if it holds that

- $F(p) = 0$ for every $p \in P$; and
- for every symbol σ in $\Sigma_{(k)}$, and sequence of states $q_1, \ldots, q_k, q \in Q$ we have $\mu_k(\sigma)_{q_1 \cdots q_k, q} = 0$ if $q \notin P$ and there exists $i \in [1, k]$ such that $q_i \in P$.

The above just restricts dead states to be non-final and inescapable. Now we can recall the all-accepting property from [12]. We say that M is *all-accepting* [12] if for every set $P \subseteq Q$ of dead states

- $|P| \leq 1$;
- $\mu_k(\sigma)_{q_1 \cdots q_k, p} \in \{0, 1\}$ for every symbol $\sigma \in \Sigma_{(k)}$, word $q_1 \cdots q_k$ of states in Q, and $p \in P$; and
- $F(q) = 1$ for all $q \in Q \setminus P$.

The name *all-accepting* is derived from the fact that alternatively we could have defined all-accepting wta to be such that the final weight of every state is 1. However, this definition is incompatible with the notion of completeness, so we presented a more involved definition.

We abbreviate *all-accepting wta* simply to *aa-wta*. Let M be a deterministic aa-wta. The tree series $\|M\|$ is *subtree-closed* [12, Section 3.1]; i.e., for every tree t with $(\|M\|, t) \neq 0$ also $(\|M\|, u) \neq 0$ for every subtree u of t. We repeat [12, Observation 3.1] for ease of reference.

Observation 3.11 (see [12, Observation 3.1]) Let M be a deterministic aa-wta. Then $\|M\|$ is subtree-closed. □

Finally, for every $q \in Q$ we denote by $\mathrm{supp}(M_q)$ the set $\{t \in T_\Sigma \mid h_\mu(t)_q \neq 0\}$. A state q of Q is said to be *useless* if $\mathrm{supp}(M_q) = \emptyset$. Note that, for a deterministic wta M over a zero-divisor free semiring $\mathcal{A}$, it can effectively be checked whether a state is useless. Now we can move on to our main statement.

Theorem 3.12 Let M be a deterministic and complete aa-wta without useless states over some multiplicatively cancellative and commutative semiring, and let $\mathcal{P}$ be the coarsest forward bisimulation on M. Then $(M/\mathcal{P})$ is the minimal deterministic and complete aa-wta recognising $\|M\|$.

Proof Let $M' = (Q', \Sigma, \mathcal{A}, F', \mu')$ be the unique (up to isomorphism) minimal deterministic and complete aa-wta [12, 25] such that $\|M'\| = \|M\|$. We prove that there exists a forward bisimulation $\mathcal{R}'$ on M such that $(M/\mathcal{R}')$ and M' are isomorphic. Since M' is minimal, such a forward bisimulation must be the coarsest forward bisimulation $\mathcal{P}$ on M.

The relation $\imath \subseteq Q \times Q'$ is defined for every state $q \in Q$ and state $q' \in Q'$ by

$$(q, q') \in \imath \quad \iff \quad \mathrm{supp}(M_q) \cap \mathrm{supp}(M'_{q'}) \neq \emptyset \ .$$

Since the wta M has no useless states we have that $\mathrm{supp}(M_q) \neq \emptyset$ for every state $q \in Q$. Moreover, for every state $q \in Q$ there exists a state $q' \in Q'$ such that $\mathrm{supp}(M_q) \subseteq \mathrm{supp}(M'_{q'})$. Hence for every state $q \in Q$ there exists a state $q' \in Q'$ such that $(q, q') \in \imath$. Moreover, since the wta M' is deterministic and complete, for every tree $t \in \mathrm{T}_\Sigma$ there exists exactly one state $q' \in Q'$ such that $t \in \mathrm{supp}(M'_{q'})$, and thus, for every state $q \in Q$ there exists at most one state $q' \in Q'$ such that $\mathrm{supp}(M_q) \cap \mathrm{supp}(M'_{q'}) \neq \emptyset$. Thus $\imath \colon Q \to Q'$ is a mapping.

Let $\mathcal{R} = \ker(\imath)$, which, by definition, is an equivalence relation. It remains to prove that $\mathcal{R}$ is a forward bisimulation on M. Thus, let $(p, q) \in \mathcal{R}$. First we verify Condition (i) of Definition 3.2. We observe that there exist trees t and u of T_Σ such that $t \in \mathrm{supp}(M_p)$ and $u \in \mathrm{supp}(M_q)$ because p and q are not useless. Since $\mathrm{supp}(M_p) \cup \mathrm{supp}(M_q) \subseteq \mathrm{supp}(M'_{\imath(p)})$ and $\|M'\| = \|M\|$, we observe that $F'(\imath(p)) = 1$ if and only if $F(p) = 1$ because $\mathcal{A}$ is zero-divisor free. In addition, $F'(\imath(q)) = 1$ if and only if $F(q) = 1$. Consequently, $F(p) = F(q)$.

Now let us move on to Condition (ii) of Definition 3.2. Let σ be a symbol of $\Sigma_{(k)}$, $D \in (Q/\mathcal{R})$ a block, and $c = q_1 \cdots q_{i-1}\Box q_{i+1} \cdots q_k$ be a context in $C^Q_{(k)}$ for some $q_1, \dots, q_k \in Q$ and index $i \in [1, k]$. We need to prove that

$$\sum_{r' \in D} \mu_k(\sigma)_{c[p],r'} = \sum_{r' \in D} \mu_k(\sigma)_{c[q],r'} \enspace .$$

Since the wta M is deterministic and complete, there exists exactly one state $p' \in Q$ and exactly one state $q' \in Q$ such that

$$\mu_k(\sigma)_{c[p],p'} \neq 0 \quad \text{and} \quad \mu_k(\sigma)_{c[q],q'} \neq 0 \enspace .$$

We show that $(p', q') \in \mathcal{R}$. First, we observe that $\imath(p') = \imath(q')$ if and only if there exists a state $r \in Q'$ such that $\mathrm{supp}(M_{p'}) \cap \mathrm{supp}(M'_r)$ and $\mathrm{supp}(M_{q'}) \cap \mathrm{supp}(M'_r)$ are nonempty. For every $i \in [1, k]$ there exist a state $r'_i \in Q'$ and a tree $t_i \in \mathrm{T}_\Sigma$ such that

$$t_i \in \mathrm{supp}(M_{q_i}) \cap \mathrm{supp}(M'_{r'_i})$$

because M has no useless states. Let $c' = r'_1 \cdots r'_{i-1}\Box r'_{i+1} \cdots r'_k$, which is a context of $C^{Q'}_{(k)}$. Since M' is deterministic and complete, we obtain that there exists a unique state $r \in Q'$ such that $\mu'_k(\sigma)_{c'[\imath(p)],r} \neq 0$. Thus

$$\{\sigma[t_1, \dots, t_{i-1}, t, t_{i+1}, \dots, t_k], \sigma[t_1, \dots, t_{i-1}, u, t_{i+1}, \dots, t_k]\} \subseteq \mathrm{supp}(M'_r) \enspace .$$

Clearly, $\sigma[t_1, \dots, t_{i-1}, t, t_{i+1}, \dots, t_k] \in \mathrm{supp}(M_{p'})$ and $\sigma[t_1, \dots, t_{i-1}, u, t_{i+1}, \dots, t_k] \in \mathrm{supp}(M_{q'})$. This proves that the intersections $\mathrm{supp}(M_{p'}) \cap \mathrm{supp}(M'_r)$ and $\mathrm{supp}(M_{q'}) \cap \mathrm{supp}(M'_r)$ are nonempty and thus $(p', q') \in \mathcal{R}$.

Thus our goal simplifies because

$$\sum_{r' \in D} \mu_k(\sigma)_{c[p],r'} = \mu_k(\sigma)_{c[p],p'} \qquad \text{and} \qquad \sum_{r' \in D} \mu_k(\sigma)_{c[q],r'} = \mu_k(\sigma)_{c[q],q'}$$

and we only have to prove that $\mu_k(\sigma)_{c[p],p'} = \mu_k(\sigma)_{c[q],q'}$. Obviously,

$$F(p') \cdot \mu_k(\sigma)_{c[p],p'} \cdot \Bigl(\prod_{j \in [1,k] \setminus \{i\}} h_\mu(t_j)_{q_j} \Bigr) \cdot h_\mu(t)_p \; = \; (\|M\|, \sigma[t_1, \dots, t_{i-1}, t, t_{i+1}, \dots, t_k])$$
$$= (\|M'\|, \sigma[t_1, \dots, t_{i-1}, t, t_{i+1}, \dots, t_k]) \; = \; F'(r) \cdot \mu'_k(\sigma)_{c'[\imath(p)],r} \cdot \Bigl(\prod_{j \in [1,k] \setminus \{i\}} h_{\mu'}(t_j)_{r'_j} \Bigr) \cdot h_{\mu'}(t)_{\imath(p)}$$

and

$$F(q') \cdot \mu_k(\sigma)_{c[q],q'} \cdot \Bigl(\prod_{j \in [1,k] \setminus \{i\}} h_\mu(t_j)_{q_j} \Bigr) \cdot h_\mu(u)_q \; = \; (\|M\|, \sigma[t_1, \dots, t_{i-1}, u, t_{i+1}, \dots, t_k])$$
$$= (\|M'\|, \sigma[t_1, \dots, t_{i-1}, u, t_{i+1}, \dots, t_k]) \; = \; F'(r) \cdot \mu'_k(\sigma)_{c'[\imath(q)],r} \cdot \Bigl(\prod_{j \in [1,k] \setminus \{i\}} h_{\mu'}(t_j)_{r'_j} \Bigr) \cdot h_{\mu'}(u)_{\imath(q)} \enspace .$$

We distinguish two cases: (i) $F(p') = 0$ and (ii) $F(p') = 1$. Let us consider the first case. Clearly, $F(p') = 0$ only if p' is the (then unique) dead state of M. We conclude that $p' = q'$ because $F(q') = 0$. Moreover, $\mu_k(\sigma)_{c\llbracket p\rrbracket,p'} = 1 = \mu_k(\sigma)_{c\llbracket q\rrbracket,q'}$ since M is all-accepting.

Let us continue with the second case. Suppose that $F(p') = 1$. Consequently, $F'(r) = 1 = F(q')$. With the help of Observation 3.11 we can simplify the above equations.

$$\mu_k(\sigma)_{c\llbracket p\rrbracket,p'} \cdot \Big(\prod_{j\in[1,k]\setminus\{i\}} (\|M\|, t_j)\Big) \cdot (\|M\|, t) = \mu'_k(\sigma)_{c'\llbracket\imath(p)\rrbracket,r} \cdot \Big(\prod_{j\in[1,k]\setminus\{i\}} (\|M'\|, t_j)\Big) \cdot (\|M'\|, t) \quad (1)$$

and

$$\mu_k(\sigma)_{c\llbracket q\rrbracket q'} \cdot \Big(\prod_{j\in[1,k]\setminus\{i\}} (\|M\|, t_j)\Big) \cdot (\|M\|, u) = \mu'_k(\sigma)_{c'\llbracket\imath(q)\rrbracket,r} \cdot \Big(\prod_{j\in[1,k]\setminus\{i\}} (\|M'\|, t_j)\Big) \cdot (\|M'\|, u) \quad (2)$$

From the above equations we obtain

$$\begin{aligned}
&\mu_k(\sigma)_{c\llbracket p\rrbracket,p'} \\
&= \mu'_k(\sigma)_{c'\llbracket\imath(p)\rrbracket,r} && \text{(by cancellation in (1) because } \|M\| = \|M'\|\text{)} \\
&= \mu'_k(\sigma)_{c'\llbracket\imath(q)\rrbracket,r} && \text{(by } \imath(p) = \imath(q)\text{)} \\
&= \mu_k(\sigma)_{c\llbracket q\rrbracket,q'} && \text{(by cancellation in (2) because } \|M\| = \|M'\|\text{)}
\end{aligned}$$

and thus $\mathcal{R}$ is a forward bisimulation.

It remains to prove that $(M/\mathcal{R})$ is isomorphic to M'. Clearly, $(M/\mathcal{R})$ is deterministic, complete, and all-accepting. Moreover, $(M/\mathcal{R})$ has $|Q'|$ states and $\|(M/\mathcal{R})\| = \|M\|$ by Theorem 3.7. Thus $(M/\mathcal{R})$ is a minimal deterministic and complete aa-wta that recognises $\|M\|$. Since the minimal deterministic and complete wta recognising $\|M\|$ is unique up to isomorphism, we obtain that $(M/\mathcal{R})$ and M' are isomorphic. ■

3.2 A forward bisimulation minimisation 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 stable candidate solution (see Definition 3.16). The relation $\mathcal{R}_{i+1}$ is derived from $\mathcal{R}_i$ by removing pairs 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 with $n = |Q|$ states, and let r be the maximum k such that $\Sigma_{(k)}$ is non-empty. Finally, we use the following shorthands in Algorithm 1.

Definition 3.13 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

```
input:            A wta $M = (Q, \Sigma, \mathcal{A}, F, \mu)$;

initially:        $\mathcal{P}_0$   := $Q \times Q$;
                  $\mathcal{R}_0$   := $\ker(F) \setminus \mathit{split}(Q)$;
                  $i$    := 0;

while $\mathcal{R}_i \neq \mathcal{P}_i$:  choose $S_i \in (Q/\mathcal{P}_i)$ and $B_i \in (Q/\mathcal{R}_i)$ such that
                        $B_i \subset S_i$ and $|B_i| \leq |S_i|/2$;
                  $\mathcal{P}_{i+1}$ := $\mathcal{P}_i \setminus \mathit{cut}(B_i)$;
                  $\mathcal{R}_{i+1}$ := $\big(\mathcal{R}_i \setminus \mathit{split}(B_i)\big) \setminus \mathit{split}(S_i \setminus B_i)$;
                  $i$    := $i + 1$;

return:           $(M/\mathcal{R}_i)$;
```

Algorithm 1: A forward bisimulation minimisation algorithm for weighted tree automata.

- $\mathit{split}(B)$ for the set of all pairs (p, q) in $Q \times Q$ such that

$$\sum_{r \in B} \mu_k(\sigma)_{c[p],r} \neq \sum_{r \in B} \mu_k(\sigma)_{c[q],r}$$

  for some $\sigma \in \Sigma_{(k)}$ and $c \in C^Q_{(k)}$. □

Let us illustrate Algorithm 1 on the wta of Example 3.3.

Example 3.14 Let $N = (P, \Delta, \mathbb{N}, G, \nu)$ be the wta of Example 3.3 that recognises the tree series ZIGZAG. We will show the iterations of the algorithm on this example wta. Let us start with the initialisation:

$$\mathcal{P}_0 = P \times P \quad \text{and} \quad \mathcal{R}_0 = \{l, L\}^2 \cup \{r, R\}^2 \cup \{\bot\}^2 \enspace .$$

Now, in the first iteration, we select $S_0 = P$ and $B_0 = \{l, L\}$ and thus compute

$$\mathcal{P}_1 = \{l, L\}^2 \cup \{r, R, \bot\}^2 \quad \text{and} \quad \mathcal{R}_1 = \mathcal{R}_0 \enspace .$$

Obviously, $\mathcal{P}_1$ is still different from $\mathcal{R}_1$, so the algorithm enters into a second iteration. We now let $S_1 = \{r, R, \bot\}$ and $B_1 = \{\bot\}$, which yields $\mathcal{R}_2 = \mathcal{P}_2$. The algorithm consequently terminates and returns the aggregated wta $(N/\mathcal{R}_2)$, which is displayed in Figure 1. □

3.2.1 Correctness and termination

As Lemma 3.19 will show, there exists a $t < n$ such that the algorithm terminates when $i = t$. We use the notations introduced in the algorithm for the discussion of its correctness and termination.

Lemma 3.15 The relation $\mathcal{R}_i$ is a refinement of $\mathcal{P}_i$ for all $i \in [0, t]$.

Proof The proof is by induction on i. The base case is satisfied by the initialisation of $\mathcal{P}_0$ to $Q \times Q$. For the induction step, we proceed as follows. Clearly, $\mathcal{R}_{i+1} \subseteq \mathcal{R}_i$ and $\mathcal{P}_{i+1} = \mathcal{P}_i \setminus \mathit{cut}(B_i)$.

Since $B_i \in (Q/\mathcal{R}_i)$, we also have the equality $\mathcal{R}_i \cap \mathit{cut}(B_i) = \emptyset$. Finally, by the induction hypothesis, the inclusion $\mathcal{R}_i \subseteq \mathcal{P}_i$ holds. It follows that

$$\mathcal{R}_{i+1} \subseteq \mathcal{R}_i = \mathcal{R}_i \setminus \mathit{cut}(B_i) \subseteq \mathcal{P}_i \setminus \mathit{cut}(B_i) = \mathcal{P}_{i+1} \ . \qquad \blacksquare$$

Lemma 3.15 ensures that $\mathcal{R}_i$ is a proper refinement of $\mathcal{P}_i$, for all $i \in [0, t-1]$. This means 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. For the next statement we an additional notion. Basically, stability tests Condition (ii) of Definition 3.2 with the block D chosen from a coarser equivalence relation.

Definition 3.16 Let $\mathcal{R}$ and $\mathcal{P}$ be two equivalence relations on Q such that $\mathcal{R} \subseteq \mathcal{P}$. We say that *$\mathcal{R}$ is stable with respect to $\mathcal{P}$* if, for every pair (p, q) in $\mathcal{R}$, symbol σ of $\Sigma_{(k)}$, context c of $C^Q_{(k)}$, and block D in $(Q/\mathcal{P})$

$$\sum_{r \in D} \mu_k(\sigma)_{c[p],r} = \sum_{r \in D} \mu_k(\sigma)_{c[q],r} \ .$$

Finally, we say that $\mathcal{R}$ is *stable* if it is stable with respect to itself. □

Note that a stable equivalence relation $\mathcal{R}$ such that $F(p) = F(q)$ for every $(p, q) \in \mathcal{R}$ is a forward bisimulation on M. Next we show that Algorithm 1 produces relations $\mathcal{R}_i$ and $\mathcal{P}_i$ such that $\mathcal{R}_i$ is stable with respect to $\mathcal{P}_i$.

Lemma 3.17 The relation $\mathcal{R}_i$ is stable with respect to $\mathcal{P}_i$ for all $i \in [0, t]$.

Proof By Lemma 3.15, the relation $\mathcal{P}_i$ is coarser than $\mathcal{R}_i$. The remaining proof is by induction on i. The base case follows from the definitions of $\mathcal{R}_0$ and $\mathcal{P}_0$. Now, let (p, q) be a pair in $\mathcal{R}_{i+1}$, let σ be a symbol in $\Sigma_{(k)}$, let c be a context in $C^Q_{(k)}$, and let D be a block in $(Q/\mathcal{P}_{i+1})$. We show that

$$\sum_{r \in D} \mu_k(\sigma)_{c[p],r} = \sum_{r \in D} \mu_k(\sigma)_{c[q],r} \ . \tag{3}$$

Depending on D, there are three cases: In the first case, the intersection between D and S_i is empty. This implies that D is also a block of $(Q/\mathcal{P}_i)$. Furthermore, the fact that (p, q) is an element of $\mathcal{R}_{i+1}$ means that (q, q') is also an element of the coarser relation $\mathcal{R}_i$. Consequently, Equation (3) holds by induction hypothesis. Alternatively, either $D = B_i$ or $D = S_i \setminus B_i$. In these cases Equation (3) trivially holds because otherwise $(p, q) \in \mathit{split}(D)$. ■

Lemma 3.18 Every forward bisimulation $\mathcal{R}$ on M is a refinement of $\mathcal{R}_i$ for every $i \in [0, t]$.

Proof The proof is by induction on i. In the base case we need to show that for every $(p, q) \in \mathcal{R}$ we can conclude that $(p, q) \in \mathcal{R}_0$. Clearly, $(p, q) \in \ker(F)$ by Condition (i) in Definition 3.2. Finally, we need to show that

$$\sum_{r \in Q} \mu_k(\sigma)_{c[p],r} = \sum_{r \in Q} \mu_k(\sigma)_{c[q],r}$$

holds for every symbol σ of $\Sigma_{(k)}$ and context c in $C^Q_{(k)}$. We compute as follows.

$$\begin{aligned}
&\sum_{r \in Q} \mu_k(\sigma)_{c[p],r} \\
&= \sum_{D \in (Q/\mathcal{R})} \Big(\sum_{r \in D} \mu_k(\sigma)_{c[p],r} \Big) \\
&= \sum_{D \in (Q/\mathcal{R})} \Big(\sum_{r \in D} \mu_k(\sigma)_{c[q],r} \Big) \\
&= \sum_{r \in Q} \mu_k(\sigma)_{c[q],r} && \text{(because } (p,q) \in \mathcal{R}\text{)}
\end{aligned}$$

To cover the induction step, we show that if (p,q) is in $\mathcal{R}$, then (p,q) is also in $\mathcal{R}_{i+1}$. This is done by examining how the minimisation algorithm obtains $\mathcal{R}_{i+1}$ from $\mathcal{R}_i$. It is required that (p,q) is in $\mathcal{R}_i$, and this is satisfied by the induction hypothesis. Moreover, it is required that $\sum_{r \in D} \mu_k(\sigma)_{c[p],r} = \sum_{r \in D} \mu_k(\sigma)_{c[q],r}$ for every symbol $\sigma \in \Sigma_{(k)}$, context $c \in C^Q_{(k)}$, and $D \in \{B_i, S_i \setminus B_i\}$. By the following computation, also this condition is met.

$$\begin{aligned}
&\sum_{r \in D} \mu_k(\sigma)_{c[p],r} \\
&= \sum_{D' \in (D/\mathcal{R})} \Big(\sum_{r \in D'} \mu_k(\sigma)_{c[p],r} \Big) && \text{(Since } \mathcal{R} \subseteq \mathcal{R}_i \text{ by induction hypothesis)} \\
&= \sum_{D' \in (D/\mathcal{R})} \Big(\sum_{r \in D'} \mu_k(\sigma)_{c[q],r} \Big) && \text{(Because } (p,q) \text{ is in } \mathcal{R}\text{, and } \mathcal{R} \text{ is stable)} \\
&= \sum_{r \in D} \mu_k(\sigma)_{c[q],r} && \text{(Since } \mathcal{R} \subseteq \mathcal{R}_i \text{ by induction hypothesis.)}
\end{aligned}$$

Hence the pair (p,q) is in $\mathcal{R}_{i+1}$, so the proof is complete. ■

Finally, we should guarantee that Algorithm 1 terminates in less than n iterations.

Lemma 3.19 There exists a $t < n$ such that $\mathcal{R}_t = \mathcal{P}_t$.

Proof Clearly, Algorithm 1 only terminates if $\mathcal{R}_t$ and $\mathcal{P}_t$ coincide for some $t > 0$. Up until termination, i.e., for all $i < t$, we have that

$$|(Q/\mathcal{R}_i)| > |(Q/\mathcal{P}_i)| \quad \text{and} \quad |(Q/\mathcal{P}_{i+1})| > |(Q/\mathcal{P}_i)|$$

hold by Lemma 3.15. The size of both $(Q/\mathcal{R}_i)$ and $(Q/\mathcal{P}_i)$ is bound from above by n. Should the algorithm reach iteration $n-1$ before terminating, we have by necessity that both $|(Q/\mathcal{P}_{n-1})|$ and $|(Q/\mathcal{R}_{n-1})|$ are equal to n, so $\mathcal{R}_{n-1}$ and $\mathcal{P}_{n-1}$ coincide. Consequently, there exists an integer $t < n$ such that $\mathcal{R}_t$ and $\mathcal{P}_t$ are equal. ■

Theorem 3.20 Algorithm 1 returns the forward bisimulation minimal wta $(M/\mathcal{P})$ where $\mathcal{P}$ is the coarsest forward bisimulation on M.

Proof Lemma 3.19 guarantees that the algorithm terminates with $\mathcal{R}_t = \mathcal{P}_t$. According to Lemma 3.17, $\mathcal{R}_t$ is stable with respect to $\mathcal{P}_t$. Consequently, $\mathcal{R}_t$ is stable because $\mathcal{R}_t = \mathcal{P}_t$. Moreover, $F(p) = F(q)$ for every $(p, q) \in \mathcal{R}_t$ because $\mathcal{R}_t \subseteq \mathcal{R}_0$. Hence, $\mathcal{R}_t$ meets Conditions (i) and (ii) of Definition 3.2 and is thus a forward bisimulation. Finally, $\mathcal{R}_t$ is coarser than any forward bisimulation on M by Lemma 3.18. By Theorem 3.9, this yields that $\mathcal{R}_t = \mathcal{P}$ is the coarsest forward bisimulation on M and that $(M/\mathcal{R}_t)$ is minimal with respect to forward bisimulation. ■

3.2.2 Time complexity

In this section, we analyse the running time of the general minimisation algorithm on input $M = (Q, \Sigma, \mathcal{A}, F, \mu)$. We use n to denote the size of the set Q, r to denote the maximum integer k such that $\Sigma_{(k)}$ is nonempty, and m to denote the size of the μ;

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

For the present we assume that the tree representation is not sparse, i.e. that it contains some $\Omega\big(\sum_{k \in [0,r]} \Sigma_{(k)} n^{k+1}\big)$ entries. The time complexity when the tree representation is sparse is briefly discussed in Section 4.3. We also assume that semiring addition can be performed in constant time.

As our computation model we choose the random access machine [35], which supports indirect addressing, and thus allows the use of pointers. This means that we can represent each block in a partition of Q with respect to $\mathcal{R}$ as a record of two-way pointers to its elements, and thus determine $[q]_{\mathcal{R}}$ in constant time. Pointers can also be used to link a state to its occurrences in the tree representation μ, so given a set of states B, the part μ^f_B of μ that contains entries of the form $\mu_k(\sigma)_{q_1 \cdots q_k, q}$, where $q \in B$, can be obtained in time $O(|\mu^f_B|)$.

As usual, μ is represented as an indexed set of matrices $(M_\sigma)_{\sigma \in \Sigma}$ over A, where M_σ is of dimension $k + 1$ if σ is in $\Sigma_{(k)}$, and each dimension is in turn indexed by Q: the value of $\mu_k(\sigma)_{q_1 \cdots q_k, q}$ is stored at coordinates $q_1, \ldots, q_k, q$ in M_σ. We assume that μ is not sparse; that is, the size of this representation of μ is equal to its abstract size m.

During the computation we also maintain the following data structures:

$\mathcal{R}$-blocks A linked list where each entry represents a block in the partition $(Q/\mathcal{R}_i)$, i being the current iteration of the algorithm. Initially, *$\mathcal{R}$-blocks* contains the entries F and $Q \setminus F$

$\mathcal{P}$-blocks A linked list where each entry S represents a block in the partition $(Q/\mathcal{P}_i)$. S contains a pointer to each entry B in *$\mathcal{R}$-blocks* such that $B \subseteq S$, labelled with the size of B. Initially, *$\mathcal{P}$-blocks* contains the single block Q, which has pointers to F and $Q \setminus F$ labelled with $|F|$ and $|Q \setminus F|$, respectively.

$\mathcal{P}$-compound A linked list of pointers to those blocks in *$\mathcal{P}$-blocks* that are composed of more than one block in $(Q/\mathcal{R}_i)$. This list is empty only if $\mathcal{R}_i = \mathcal{P}_i$.

To avoid pairwise comparisons between states, we employ a technique that uses hash functions: for each state q in Q and $k \in [0, r]$ a map $obsf^k_q : \Sigma_{(k)} \times C^Q_{(k)} \times \mathfrak{P}(Q) \to A$ is computed, where $obsf^k_q(\sigma, c, B)$ is the sum

$$\sum_{r \in B} \mu_k(\sigma)_{c[q], r} \enspace .$$

The states are then hashed using $obsf_q^k$ as key for the state q, and afterwards we simply inspect which states have been sent to the same position in the hash table. Since a random access machine has unlimited memory, we can always implement a collision free hash h; e.g., by interpreting the binary representation of $(obsf_q^k)_{k\in[0,r]}$ as a memory address. The time required to hash a state q is then proportional to the size of the representation of $(obsf_q^k)_{k\in[0,r]}$, which in turn is proportional to the size of its support.

Observation 3.21 The overall time complexity of the algorithm is

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

where $\textsc{Init}^f$, $\textsc{Select}_i$, $\textsc{Cut}_i$, $\textsc{Split}_i^f$, and $\textsc{Aggregate}^f$ 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 3.22 $\textsc{Init}^f$ is in $O(m+n)$.

Proof The single block of $\mathcal{P}_0$ can be initialised in $O(n)$ steps. By Lemma 3.25, we can calculate *split*(Q), and thus initialise $\mathcal{R}_0$, in time $O(|\mu_Q^f|) = O(|\mu|) = O(m)$. ■

Lemma 3.23 $\textsc{Select}_i$ is in $O(1)$.

Proof We choose as S_i the block pointed to by the first entry in $\mathcal{P}$*-compound*. We then consider the blocks in $(Q/\mathcal{R}_i)$ that are contained in S_i. Clearly, at most one of these blocks can be of size greater than half of the size of S_i, so we need not consider more than two blocks before we find a suitable B_i. ■

Lemma 3.24 $\textsc{Cut}_i$ is in $O(|B_i|)$.

Proof The entry S_i is removed from $\mathcal{P}$*-compound*, and the corresponding entry S_i in $\mathcal{P}$*-blocks* is split into two entries, $S_i \setminus B_i$ and B_i, by iterating over the states pointed to by B_i in time $O(|B_i|)$. The entry representing $S_i \setminus B_i$ points to every block in $(Q/\mathcal{R}_i)$ that the entry representing S_i pointed to, except of course B_i. If $S_i \setminus B_i$ is still compound, then it is added to $\mathcal{P}$*-compound*. The entry representing B_i only points to B_i. ■

Lemma 3.25 The derivation of *split*(B), where $B \subseteq Q$, is in $O(r\,|\mu_B^f|)$.

Proof The computation is divided into two parts: for each entry $\mu_k(\sigma)_{q_1\cdots q_k,q}$ in μ_B^f and i in $[1,k]$, the value of $obsf_{q_i}^k(\sigma, q_1\cdots q_{i-1}\Box\, q_{i+1}\cdots q_k, B)$ is adjusted according to $\mu_k(\sigma)_{q_1\cdots q_k,q}$. Since this requires us to iterate the entire sequence of states on the left-hand side of each entry in μ_B^f, the time required is in $O(r\,|\mu_B^f|)$.

The set *split*(B) can thereafter be derived by hashing each state q in Q using $obsf_q^k$ on the domain

$$\bigcup_{k\in[0,r]} (\Sigma_{(k)} \times C_{(k)}^Q \times \{B\})$$

as key. Parallel to the hash process the lists $\mathcal{R}$-*blocks*, $\mathcal{P}$-*blocks*, and $\mathcal{P}$-*compound* are updated to reflect the changes in $\mathcal{R}$. We note that each entry $\mu_k(\sigma)_{c[q],q'}$ in μ_B^f can give rise to at most r elements in the support of $(\mathit{obsf}_q^k)_{q \in Q, k \in \mathbb{N}}$. In other words, the size of the support of $(\mathit{obsf}_q^k)_{q \in Q, k \in \mathbb{N}}$, does not exceed $O(r\,|\mu_B^f|)$, so the time required to hash the states in Q is determined by the time it takes to compute their hash keys. ■

Lemma 3.26 $\textsc{Split}_i^f$ is in $O(r\,|\mu_{S_i}^f|)$.

Proof By Lemma 3.25, we can split against B_i in time $O(r\,|\mu_{B_i}^f|)$, and against $S_i \setminus B_i$ in time $O(r\,|\mu_{S_i \setminus B_i}^f|)$, so the overall time complexity of $\textsc{Split}_i^f$ becomes $O(r\,|\mu_{S_i}^f|)$. ■

Lemma 3.27 $\textsc{Aggregate}^f$ is in $O(m + n)$.

Proof The complexity of deriving the aggregated wta $(M/\mathcal{R}_t) = (Q', \Sigma, \mathcal{A}, F', \mu')$ is a follows: The components Σ and $\mathcal{A}$ are identical to those in M, an can be determined in constant time. The components Q' and F' are both given by the entries in $\mathcal{R}$-*blocks*; to determine if a block in $\mathcal{R}$-*blocks* is final, just follow the pointer to one of its constituent states and check if that state is final. This list can be read in time $O(n)$. To obtain μ' it suffices to iterate over μ, as each state is linked to its equivalence class. This requires another $O(m)$ computation steps. ■

Theorem 3.28 The general algorithm has time complexity $O(rmn)$.

Proof By Observation 3.21, in combination with Lemmata 3.22 – 3.27, the total time consumed can be written

$$O\Big((m+n) + \sum_{i \in [0,t-1]} \big(1 + |B_i| + r\,|\mu_{S_i}^f|\,\big) + (m+n)\Big) \enspace ,$$

In the worst case, $|S_i|$ equals $n - i$, and the expression simplifies to $O(rmn)$. ■

When M is a deterministic wta, the size m of the tree representation is bounded from above by $|\Sigma|\,n^r$, as opposed to $|\Sigma|\,n^{r+1}$ in the nondeterministic case. We thus have Corollary 3.29.

Corollary 3.29 The algorithm is in $O\big(r\,|\Sigma|\,n^{r+1}\big)$ when restricted to deterministic wta.

3.2.3 Additively cancellative semirings

In this section, we show a simplification of Algorithm 1 for additively cancellative semirings. In essence, the second split in the computation of $\mathcal{R}_{i+1}$ can be omitted. Thus for the remainder of this section, let $\mathcal{A}$ be an additively cancellative semiring.

Lemma 3.30 Without effect on the overall algorithm we can replace the computation of $\mathcal{R}_{i+1}$ in Algorithm 1 simply by $\mathcal{R}_{i+1} = \mathcal{R}_i \setminus \mathit{split}(B_i)$.

Proof We strengthen this claim by proving, through induction on i, that

$$\mathcal{R}_i \setminus \mathit{split}(B_i) = \big(\mathcal{R}_i \setminus \mathit{split}(B_i)\big) \setminus \mathit{split}(S_i \setminus B_i)$$

holds for all i. Clearly, an element from the right-hand side is also in the left-hand side. It remains to show that every (p, q) in the left-hand side is also in the right-hand side. Suppose that $(p, q) \in \mathcal{R}_i$

and $(p,q) \notin \mathit{split}(B_i)$. Let σ be a symbol in $\Sigma_{(k)}$, and c be a context in $C^Q_{(k)}$. This immediately yields that $\sum_{r\in B_i} \mu_k(\sigma)_{c[p],r} = \sum_{r\in B_i} \mu_k(\sigma)_{c[q],r}$. By Lemma 3.17 and the induction hypothesis, $\mathcal{R}_i$ is stable with respect to $\mathcal{P}_i$. Thus we conclude that $\sum_{r\in S_i} \mu_k(\sigma)_{c[p],r} = \sum_{r\in S_i} \mu_k(\sigma)_{c[q],r}$. We compute as follows.

$$\Big(\sum_{r\in S_i\setminus B_i} \mu_k(\sigma)_{c[p],r}\Big) + \Big(\sum_{r\in B_i} \mu_k(\sigma)_{c[p],r}\Big) = \sum_{r\in S_i} \mu_k(\sigma)_{c[p],r}$$
$$= \sum_{r\in S_i} \mu_k(\sigma)_{c[q],r} = \Big(\sum_{r\in S_i\setminus B_i} \mu_k(\sigma)_{c[q],r}\Big) + \Big(\sum_{r\in B_i} \mu_k(\sigma)_{c[q],r}\Big)$$

Since we additionally have that $\sum_{r\in B_i} \mu_k(\sigma)_{c[p],r}$ is equal to $\sum_{r\in B_i} \mu_k(\sigma)_{c[q],r}$, we can apply the cancellation law to the equation

$$\Big(\sum_{r\in S_i\setminus B_i} \mu_k(\sigma)_{c[p],r}\Big) + \Big(\sum_{r\in B_i} \mu_k(\sigma)_{c[p],r}\Big) = \Big(\sum_{r\in S_i\setminus B_i} \mu_k(\sigma)_{c[q],r}\Big) + \Big(\sum_{r\in B_i} \mu_k(\sigma)_{c[q],r}\Big)$$

and obtain that $\sum_{r\in S_i\setminus B_i} \mu_k(\sigma)_{c[p],r}$ is equal to $\sum_{r\in S_i\setminus B_i} \mu_k(\sigma)_{c[q],r}$. Consequently, (p,q) is not in $\mathit{split}(S_i \setminus B_i)$ and the statement is proved. ■

The following lemma shows an important observation that is needed to prove the improvement in the time complexity for the revised algorithm for wta on additively cancellative semirings.

Lemma 3.31 For each $q \in Q$ we have that $|\{i \in [0,t-1] \mid q \in B_i\}| \leq \log n$.

Proof Let i and j be such that $i < j$ and $q \in B_i \cap B_j$. Since $\mathcal{R}_j$ is a refinement of $\mathcal{R}_i$, we have that B_j is a subset of B_i. We know then that $|B_j|$ is less or equal to $|B_i|/2$, or else B_j would violate the selection criteria for the B-blocks. If we order the B-blocks in which q occurs in descending order (with respect to their size), we have that each block in the list is at most half of the size of its predecessor. The first block in which q occurs cannot be larger than n, and the last block cannot be smaller than a singleton. Hence, the number of blocks that include q is at most $\log n$. ■

We now consider the time complexity of the algorithm optimised for cancellative semirings on input $M = (Q, \Sigma, \mathcal{A}, F, \mu)$. The notations and assumptions are identical to those of the previous section.

Theorem 3.32 The algorithm for additively cancellative semirings is in $O(rm\log n)$.

Proof By Lemma 3.26, the computation of $\mathit{split}(B_i)$ is in $O(r\,|\mu^f_{B_i}|)$, so the time consumed by Iteration 0 to $t-1$ is in

$$O\left(r \sum_{i\in[0,t-1]} |\mu^f_{B_i}|\right) .$$

According to Lemma 3.31, no state occurs in more than $\log n$ distinct B-blocks, and an entry in μ can only contribute to $|\mu^f_{B_i}|$ if its last state is in B_i. Consequently, no entry $\mu_k(\sigma)_{q_1\cdots q_k,q}$ can contribute by more than $\log n$ to the total sum, and since there are $\sum_{k\in[0,r]} |\Sigma_{(k)}|\, n^{k+1}$ entries, the sum can be rewritten as $O(r\sum_{k\in[0,r]} |\Sigma_{(k)}|\, n^{k+1}\log n)$, which simplifies to $O(rm\log n)$. ■

Corollary 3.33 The algorithm optimised for cancellative semirings is in $O(r\,|\Sigma|\,n^r\log n)$ when restricted to deterministic wta.

4 Backward bisimulation

4.1 Foundation

In this section we investigate backward bisimulation [10]. We introduce the following supporting notation. Let Π be a partition of a set Q. For every integer $k \in \mathbb{N}$ we write $\Pi_{(k)}$ for the set $\{D_1 \times \cdots \times D_k \mid D_1, \ldots, D_k \in \Pi\}$. Finally, we write $\Pi_{(\leq k)}$ for the set $\Pi_{(0)} \cup \cdots \cup \Pi_{(k)}$.

Definition 4.1 (cf. [10, Definition 4.1]) Let $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ be a wta, and let $\mathcal{R}$ be an equivalence relation on Q. We say that $\mathcal{R}$ is a *backward bisimulation on M* if for every $(p, q) \in \mathcal{R}$ and every symbol σ in $\Sigma_{(k)}$ and word $L \in (Q/\mathcal{R})_{(k)}$

$$\sum_{w \in L} \mu_k(\sigma)_{w,p} = \sum_{w \in L} \mu_k(\sigma)_{w,q} \quad . \qquad \square$$

Example 4.2 Let us present another wta that recognises the tree series ZIGZAG. As in Example 3.3 let $\Delta = \Delta_{(0)} \cup \Delta_{(2)}$ with $\Delta_{(0)} = \{\alpha\}$ and $\Delta_{(2)} = \{\sigma\}$ and $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$ be the semiring of natural numbers. We present the wta $N = (P, \Delta, \mathbb{N}, G, \nu)$ with $P = \{l, r, L, R, \bot\}$ and $G(l) = 1$ and $G(p) = 0$ for every $p \in \{r, L, R, \bot\}$ and

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

All remaining entries in ν are 0. The wta N is displayed in Figure 2. It can be shown that the wta N recognises the tree series ZIGZAG (this tree series is also recognised by the wta of Examples 3.3 and 3.5). If we apply Algorithm 1 to N, we will note that N only admits the trivial forward bisimulation and is thus minimal with respect to forward bisimulation minimisation.

Let us try to find a backward bisimulation on N. We propose $\mathcal{P} = \{l\}^2 \cup \{r\}^2 \cup \{L, R, \bot\}^2$. We will now demonstrate that $\mathcal{P}$ is a backward bisimulation on M. We immediately note that $\nu_0(\alpha)_{\varepsilon,L} = \nu_0(\alpha)_{\varepsilon,R} = \nu_0(\alpha)_{\varepsilon,\bot}$. Finally, we also observe that for every $p_1, p_2 \in P$ such that $(p_1, \bot) \notin \mathcal{P}$ and $(p_2, \bot) \notin \mathcal{P}$ we have $\nu_2(\sigma)_{p_1 p_2, p} = 0$ for every $p \in \{L, R, \bot\}$ and

$$\begin{aligned}
\sum_{p_1 p_2 \in [\bot][\bot]} \nu_2(\sigma)_{p_1 p_2, L} &= \nu_2(\sigma)_{R\bot,L} = 1 \\
\sum_{p_1 p_2 \in [\bot][\bot]} \nu_2(\sigma)_{p_1 p_2, R} &= \nu_2(\sigma)_{\bot L,R} = 1 \\
\sum_{p_1 p_2 \in [\bot][\bot]} \nu_2(\sigma)_{p_1 p_2, \bot} &= \nu_2(\sigma)_{\bot\bot,\bot} = 1 \quad .
\end{aligned}$$

Thus $\mathcal{P}$ is a backward bisimulation on N. □

Since we are again only interested in reducing the size of the wta, we define how to collapse a wta M with respect to a backward bisimulation $\mathcal{R}$ on M. For the rest of this section, let $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ be a wta, and let $\mathcal{R}$ be a backward bisimulation on M.

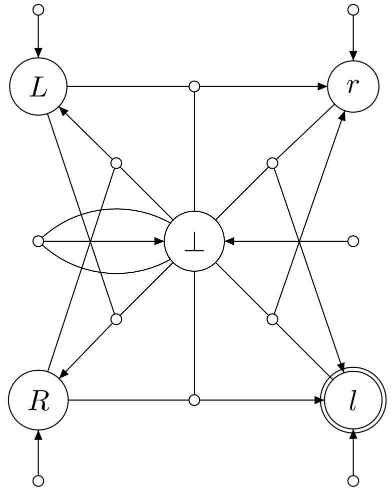


Figure 2: The wta of Example 4.2; every binary transition is labelled with $\sigma/1$ and every nullary transition with $\alpha/1$ and final weights are 1 for doubly-circled states and 0 otherwise.

Definition 4.3 (cf. [10, Definition 3.3]) The *(backward) aggregated wta* $(M/\mathcal{R})$ [10] *(with respect to* M *and* $\mathcal{R}$*)* is the wta $(Q', \Sigma, \mathcal{A}, F', \mu')$ given by

- $Q' = (Q/\mathcal{R})$;
- $F'(D) = \sum_{q \in D} F(q)$ for every block D of $(Q/\mathcal{R})$; and
- for every symbol σ in $\Sigma_{(k)}$, word $D_1 \cdots D_k$ of blocks in $(Q/\mathcal{R})$, and state $q \in Q$

$$\mu'_k(\sigma)_{D_1 \cdots D_k,[q]} = \sum_{w \in D_1 \cdots D_k} \mu_k(\sigma)_{w,q} \enspace .$$

Clearly, the wta $(M/\mathcal{R})$ is well-defined because $\mathcal{R}$ is a backward bisimulation on M. □

The above definition captures the size reduction. It is clear that the aggregated wta $(M/\mathcal{R})$ will have as many states as there are equivalence classes with respect to $\mathcal{R}$, and also that $(M/\mathcal{R})$ cannot have more states than M. In fact, our aim is to collapse as many states as possible into a single state. Let us show an example of a backward aggregated wta.

Example 4.4 Recall the wta N and the backward bisimulation $\mathcal{P}$ from Example 4.2. According to Definition 4.3 we obtain the aggregated wta $(N/\mathcal{P}) = (P', \Delta, \mathbb{N}, G', \nu')$ with $P' = \{[l], [r], [\bot]\}$ and $G'([l]) = 1$ and $G'([r]) = G([\bot]) = 0$ and

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

All remaining entries in ν' are 0. As we will see in Theorem 4.7 also $(N/\mathcal{P})$ recognises ZIGZAG. In fact, $(N/\mathcal{P})$ is isomorphic to the forward aggregated wta in Example 3.5. Thus, the graphical representation can be found in Figure 1. □

Next we prepare Theorem 4.7, which will show that wta M and $(M/\mathcal{R})$ recognise the same series. Our helping lemma will relate a state q of M to the state $[q]$ of $(M/\mathcal{R})$. The linking property is that those two states (in their respective wta) recognise the same tree series.

Lemma 4.5 (cf. [10, Theorem 4.2] and [2, Lemma 5.2]) Let $(M/\mathcal{R}) = (Q', \Sigma, \mathcal{A}, F', \mu')$ be as defined in Definition 4.3. Then $h_{\mu'}(t)_{[q]} = h_\mu(t)_q$ for every state $q \in Q$ and tree $t \in \mathrm{T}_\Sigma$.

Proof We prove the statement inductively. Suppose that $t = \sigma[t_1, \ldots, t_k]$ for some $\sigma \in \Sigma_{(k)}$ and $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$.

$$\begin{aligned}
&h_{\mu'}(\sigma[t_1, \ldots, t_k])_{[q]} \\
= &\sum_{D_1,\ldots,D_k \in (Q/\mathcal{R})} \mu'_k(\sigma)_{D_1 \cdots D_k,[q]} \cdot h_{\mu'}(t_1)_{D_1} \cdot \ldots \cdot h_{\mu'}(t_k)_{D_k} \\
= &\sum_{D_1,\ldots,D_k \in (Q/\mathcal{R})} \Big(\sum_{q_1 \cdots q_k \in D_1 \cdots D_k} \mu_k(\sigma)_{q_1 \cdots q_k, q} \Big) \cdot h_{\mu'}(t_1)_{D_1} \cdot \ldots \cdot h_{\mu'}(t_k)_{D_k} \\
= &\text{ (by induction hypothesis applied } k \text{ times)} \\
&\sum_{\substack{D_1,\ldots,D_k \in (Q/\mathcal{R}),\\ q_1 \cdots q_k \in D_1 \cdots D_k}} \mu_k(\sigma)_{q_1 \cdots q_k, q} \cdot h_\mu(t_1)_{q_1} \cdot \ldots \cdot h_\mu(t_k)_{q_k} \\
= &\, h_\mu(\sigma[t_1, \ldots, t_k])_q
\end{aligned}$$

■

The previous lemma establishes a nice property of bisimilar states. Let us make the observation that bisimilar states recognise the same tree series explicit here.

Corollary 4.6 (of Lemma 4.5) It holds that $h_\mu(t)_p = h_\mu(t)_q$ for every pair $(p, q) \in \mathcal{R}$ of bisimilar states and every tree $t \in \mathrm{T}_\Sigma$. □

With Lemma 4.5 we can easily prove that the aggregated wta $(M/\mathcal{R})$ recognises the same tree series as the wta M. In particular, this shows that the approach via backward bisimulation is sound.

Theorem 4.7 (cf. [10, Theorem 4.2] and [2, Lemma 5.3]) $\|(M/\mathcal{R})\| = \|M\|$.

Proof Let $(M/\mathcal{R}) = (Q', \Sigma, \mathcal{A}, F', \mu')$. For every tree $t \in \mathrm{T}_\Sigma$

$$\begin{aligned}
&(\|(M/\mathcal{R})\|, t) = \sum_{D \in (Q/\mathcal{R})} F'(D) \cdot h_{\mu'}(t)_D = \sum_{D \in (Q/\mathcal{R})} \Big(\sum_{q \in D} F(q) \Big) \cdot h_{\mu'}(t)_D \\
= &\sum_{D \in (Q/\mathcal{R}), q \in D} F(q) \cdot h_\mu(t)_q \qquad \text{(by Lemma 4.5)} \\
= &\, (\|M\|, t) \ .
\end{aligned}$$

■

Clearly, among all backward bisimulations on M the coarsest one yields the smallest aggregated wta. Further, this wta admits only the trivial backward bisimulation. We will show that there exists a unique coarsest backward bisimulation $\mathcal{P}$ on M and then show that $(M/\mathcal{P})$ is really minimal with respect to backward bisimulation.

Lemma 4.8 (cf. [10, Theorem 3.5]) Let $\mathcal{R}$ and $\mathcal{P}$ be backward bisimulations on M. Then there exists a backward bisimulation $\mathcal{R}'$ on M such that $\mathcal{R} \cup \mathcal{P} \subseteq \mathcal{R}'$.

Proof Let $\mathcal{R}'$ be the smallest equivalence containing $\mathcal{R} \cup \mathcal{P}$. We now show that $\mathcal{R}'$ is a backward bisimulation on M. Let $(p, q) \in \mathcal{R}'$. Thus there exist $n \in \mathbb{N}$ and

$$(p_1, p_2), (p_2, p_3), \ldots, (p_{n-2}, p_{n-1}), (p_{n-1}, p_n) \in \mathcal{R} \cup \mathcal{P}$$

such that $p_1 = p$ and $p_n = q$. Clearly, every block $D \in (Q/\mathcal{R}')$ is a union of blocks of $(Q/\mathcal{R})$ as well as a union of blocks of $(Q/\mathcal{P})$. Thus the condition of Definition 4.1 is trivially met for all $(p_i, p_{i+1}) \in \mathcal{R} \cup \mathcal{P}$ with $i \in [1, n-1]$. Now we verify the condition for p and q. Thus, let σ be a symbol of $\Sigma_{(k)}$, and let $L \in (Q/\mathcal{R}')_{(k)}$.

$$\sum_{w \in L} \mu_k(\sigma)_{w,p_1} = \sum_{w \in L} \mu_k(\sigma)_{w,p_2} = \ldots = \sum_{w \in L} \mu_k(\sigma)_{w,p_{n-1}} = \sum_{w \in L} \mu_k(\sigma)_{w,p_n} \qquad \blacksquare$$

Theorem 4.9 (cf. [10, Theorem 3.2]) There exists a coarsest backward bisimulation $\mathcal{P}$ on M, and the identity is the only backward bisimulation on $(M/\mathcal{P})$.

Proof The existence of the coarsest backward bisimulation is a direct consequence of Lemma 4.8. We prove the remaining part by contradiction. Assume that $(M/\mathcal{P}) = (Q', \Sigma, \mathcal{A}, F', \mu')$ admits a non-trivial backward bisimulation $\mathcal{R}'$. Whenever we write $[p]$ with $p \in Q$, we mean $[p]_\mathcal{P}$. We construct the relation $\mathcal{P}' = \{(p, q) \mid ([p], [q]) \in \mathcal{R}'\}$. Trivially, $\mathcal{P}'$ is an equivalence relation on Q. Further, we claim that $\mathcal{P}'$ is a backward bisimulation on M. To validate the claim, let $(p, q) \in \mathcal{P}'$. Moreover, let $\sigma \in \Sigma_{(k)}$ be a symbol and $D_1 \cdots D_k \in (Q/\mathcal{P}')^k$ be a sequence of blocks. Clearly, for every $i \in [1, k]$ the block D_i is a union $D_{i,1} \cup \cdots \cup D_{i,n_i}$ of pairwise different blocks $D_{i,1}, \ldots, D_{i,n_i} \in (Q/\mathcal{P})$. Moreover, $D'_i = \{D_{i,1}, \ldots, D_{i,n_i}\}$ is a block in $(Q'/\mathcal{R}')$.

$$\begin{aligned}
\sum_{w \in D_1 \cdots D_k} \mu_k(\sigma)_{w,p} &= \sum_{D''_1 \cdots D''_k \in D'_1 \cdots D'_k} \Big(\sum_{w \in D''_1 \cdots D''_k} \mu_k(\sigma)_{w,p} \Big) \\
&= \sum_{D''_1 \cdots D''_k \in D'_1 \cdots D'_k} \mu'_k(\sigma)_{D''_1 \cdots D''_k,[p]} && \text{(by Definition 4.3)} \\
&= \sum_{D''_1 \cdots D''_k \in D'_1 \cdots D'_k} \mu'_k(\sigma)_{D''_1 \cdots D''_k,[q]} && \text{(by } ([p],[q]) \in \mathcal{R}'\text{)} \\
&= \sum_{D''_1 \cdots D''_k \in D'_1 \cdots D'_k} \Big(\sum_{w \in D''_1 \cdots D''_k} \mu_k(\sigma)_{w,q} \Big) && \text{(by Definition 4.3)} \\
&= \sum_{w \in D_1 \cdots D_k} \mu_k(\sigma)_{w,q}
\end{aligned}$$

Thus $\mathcal{P}'$ is a backward bisimulation on M. Moreover, $\mathcal{P}'$ is coarser than $\mathcal{P}$, and since $\mathcal{R}'$ is not the identity it follows that $\mathcal{P} \subset \mathcal{P}'$. This contradicts the assumption that $\mathcal{P}$ is the coarsest backward bisimulation on M. Consequently, $(M/\mathcal{P})$ only admits the identity as backward bisimulation. ■

4.2 A backward bisimulation minimisation algorithm

We now show how the minimisation algorithm of Section 3.2 can be modified so as to minimise with respect to backward bisimulation. For the rest of this section, let $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ be an arbitrary but fixed wta with $n = |Q|$ states.

Intuitively, $\sum_{w \in D_1 \cdots D_k} \mu_k(\sigma)_{w,q}$ 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 4.1). 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.

In the new algorithm, we need the following additional notations.

Definition 4.10 Let B, B' be subsets of 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 $\mathit{split}^b(\mathcal{L})$ 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 $\sum_{w \in L} \mu_k(\sigma)_{w,p} \neq \sum_{w \in L} \mu_k(\sigma)_{w,q}$.

We can now construct a minimisation algorithm based on backward bisimulation by replacing the initialisation of $\mathcal{R}_0$ in Algorithm 1 with

$$\mathcal{R}_0 = \mathcal{P}_0 \setminus \mathit{split}^b((Q/\mathcal{P}_0)_{\leq r}) \ ,$$

and the computation of $\mathcal{R}_{i+1}$ with

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

The modified algorithm is shown in Algorithm 2.

Example 4.11 Consider the execution of the backward bisimulation minimisation algorithm on the wta $N = (P, \Delta, \mathbb{N}, G, \nu)$ of Example 4.2. Clearly, $\mathcal{P}_0$ is $P \times P$. In the computation of $\mathcal{P}_0 \setminus \mathit{split}^b(\mathcal{L}_0)$, 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 $\mathcal{R}_1$. □

4.2.1 Correctness and termination

We now verify that the algorithm is correct. Note that Lemmata 3.15 and 3.19 are still valid, since they do not depend on the definition of the *split* function. Thus, Algorithm 2 terminates in $t < n$ iterations and $\mathcal{R}_i \subseteq \mathcal{P}_i$ for every $i \in [0, t]$. We provide Lemmata 4.13 and 4.14 as respective replacements for Lemmata 3.17 and 3.18.

```
input:              A wta $M = (Q, \Sigma, \mathcal{A}, F, \mu)$;

initially:          $\mathcal{P}_0 := Q \times Q$;
                    $\mathcal{L}_0 := (Q/\mathcal{P}_0)_{(\leq r)}$;
                    $\mathcal{R}_0 := \mathcal{P}_0 \setminus \mathit{split}^b(\mathcal{L}_0)$;
                    $i := 0$;

while $\mathcal{R}_i \neq \mathcal{P}_i$:    choose $S_i \in (Q/\mathcal{P}_i)$ and $B_i \in (Q/\mathcal{R}_i)$ such that
                          $B_i \subset S_i$ and $|B_i| \leq |S_i|/2$;
                    $\mathcal{P}_{i+1} := \mathcal{P}_i \setminus \mathit{cut}(B_i)$;
                    $\mathcal{L}_{i+1} := (Q/\mathcal{P}_{i+1})_{(\leq r)}$;
                    $\mathcal{R}_{i+1} := \big(\mathcal{R}_i \setminus \mathit{split}^b(\mathcal{L}_{i+1}(B_i))\big) \setminus \mathit{split}^b(\mathcal{L}_{i+1}(S_i \setminus B_i, \neg B_i))$;
                    $i := i + 1$;

return:             $(M/\mathcal{R}_i)$;
```

Algorithm 2: A backward bisimulation minimisation algorithm for weighted tree automata.

Definition 4.12 Let $\mathcal{R}$ and $\mathcal{P}$ be two equivalence relations on Q, where $\mathcal{P}$ is coarser than $\mathcal{R}$. We say that $\mathcal{R}$ *is stable with respect to* $\mathcal{P}$ if for every pair (p, q) in $\mathcal{R}$ the sums $\sum_{w \in L} \mu_k(\sigma)_{w,p}$ and $\sum_{w \in L} \mu_k(\sigma)_{w,q}$ are equal for every $\sigma \in \Sigma_{(k)}$ and $L \in (Q/\mathcal{P})_{(k)}$. □

We note that $\mathcal{R}$ is a backward bisimulation on M if and only if it is stable with respect to itself.

Lemma 4.13 The relation $\mathcal{R}_i$ is stable with respect to $\mathcal{P}_i$ for all $i \in [0, t]$.

Proof By Lemma 3.15, the relation $\mathcal{P}_i$ is coarser than $\mathcal{R}_i$. The remaining proof is by induction on i. The base case follows from the definitions of $\mathcal{R}_0$ and $\mathcal{P}_0$. Now, let (p, q) be a pair in $\mathcal{R}_{i+1}$, let σ be a symbol in $\Sigma_{(k)}$, and let L be a language in $(Q/\mathcal{P}_{i+1})_{(k)} = \mathcal{L}_{i+1} \cap \mathfrak{P}(Q^k)$. We show that

$$\sum_{w \in L} \mu_k(\sigma)_{w,p} = \sum_{w \in L} \mu_k(\sigma)_{w,q} \quad . \tag{4}$$

Depending on L, there are three cases: First let L be in $\mathcal{L}_i \cap \mathfrak{P}(Q^k) = (Q/\mathcal{P}_i)_{(k)}$. The fact that (p, q) is an element of $\mathcal{R}_{i+1}$ means that (p, q) is also an element of the coarser relation $\mathcal{R}_i$. Supporting ourselves on the induction hypothesis, we have that (4) holds.

Second, let L be in the set $\mathcal{L}_{i+1}(B_i) \cap \mathfrak{P}(Q^k)$, and finally let L be in $\mathcal{L}_{i+1}(S_i \setminus B_i, \neg B_i) \cap \mathfrak{P}(Q^k)$. In both cases, Equation (4) holds by the computation of $\mathcal{R}_{i+1}$ in Algorithm 2 because $L \in \mathcal{L}_{i+1}(B_i)$ or $L \in \mathcal{L}_{i+1}(S_i \setminus B_i, \neg B_i)$. ■

Lemma 4.14 Every backward bisimulation $\mathcal{R}$ on M is a refinement of $\mathcal{R}_i$ for every $i \in [0, t]$.

Proof The proof is by induction on i. In the induction base, let $(p, q) \in \mathcal{R}$. It remains to show that $\sum_{w \in Q^k} \mu_k(\sigma)_{w,p} = \sum_{w \in Q^k} \mu_k(\sigma)_{w,q}$ for every symbol $\sigma \in \Sigma_{(k)}$. Clearly, $Q^k = \bigcup (Q/\mathcal{R})_{(k)}$. We compute as follows.

$$\sum_{w \in Q^k} \mu_k(\sigma)_{w,p} = \sum_{L \in (Q/\mathcal{R})_{(k)}} \Big(\sum_{w \in L} \mu_k(\sigma)_{w,p} \Big) = \sum_{L \in (Q/\mathcal{R})_{(k)}} \Big(\sum_{w \in L} \mu_k(\sigma)_{w,q} \Big) = \sum_{w \in Q^k} \mu_k(\sigma)_{w,q}$$

To cover the induction step, we show that if (p,q) is in $\mathcal{R}$, then (p,q) is also in $\mathcal{R}_{i+1}$. This is done by examining how the minimisation algorithm obtains $\mathcal{R}_{i+1}$ from $\mathcal{R}_i$. It is required that (p,q) is in $\mathcal{R}_i$, and this is satisfied by the induction hypothesis. Moreover, it must hold that $\sum_{w \in L} \mu_k(\sigma)_{w,p} = \sum_{w \in L} \mu_k(\sigma)_{w,q}$ for every symbol $\sigma \in \Sigma_{(k)}$, and language L in the sets $\mathcal{L}_{i+1}(B_i) \cap \mathfrak{P}(Q^k)$ and $\mathcal{L}_{i+1}(S_i \setminus B_i, \neg B_i) \cap \mathfrak{P}(Q^k)$. Since $\mathcal{R} \subseteq \mathcal{R}_i$ by induction hypothesis and $\mathcal{R}_i \subseteq \mathcal{P}_{i+1}$ we immediately observe that $L = L_1 \cup \dots \cup L_n$ for some pairwise disjoint $L_1, \dots, L_n \in (Q/\mathcal{R})_{(k)}$. By the following computation, also this condition is met.

$$\sum_{w \in L} \mu_k(\sigma)_{w,p} = \sum_{i \in [1,n]} \Big(\sum_{w \in L_i} \mu_k(\sigma)_{w,p} \Big) = \sum_{i \in [1,n]} \Big(\sum_{w \in L_i} \mu_k(\sigma)_{w,q} \Big) = \sum_{w \in L} \mu_k(\sigma)_{w,q}$$

where the second equality holds because $(p,q) \in \mathcal{R}$. ■

Theorem 4.15 Algorithm 2 returns the backward bisimulation minimal wta $(M/\mathcal{P})$ where $\mathcal{P}$ is the coarsest backward bisimulation on M.

Proof Lemma 3.19 guarantees that the algorithm terminates with $\mathcal{R}_t = \mathcal{P}_t$. According to Lemma 4.13, $\mathcal{R}_t$ is stable with respect to $\mathcal{P}_t$. Consequently, $\mathcal{R}_t$ is a backward bisimulation because $\mathcal{R}_t = \mathcal{P}_t$. Finally, $\mathcal{R}_t$ is coarser than every backward bisimulation on M by Lemma 4.14. By Theorem 4.9, this yields that $\mathcal{R}_t = \mathcal{P}$ is the coarsest backward bisimulation on M and that $(M/\mathcal{R}_t)$ is minimal with respect to backward bisimulation. ■

4.2.2 Time complexity

We now compute the time complexity of the backward bisimulation algorithm, using the same assumptions and notations as in Section 3.2.2. 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 $B_1 \cdots B_k$ for some $B_1 \cdots B_k \in L$.

We still want to substitute hashing for pairwise comparisons, so we define, for each $q \in Q$ and $k \in [0,r]$, the mapping $obsb^k_q : \Sigma_{(k)} \times \mathfrak{P}(Q)^*$ that is given by

$$obsb^k_q(\sigma, D_1 \cdots D_k) = \sum_{q_1 \cdots q_k \in D_1 \cdots D_k} \mu_k(\sigma)_{q_1 \cdots q_k, q} \enspace .$$

Observation 4.16 The overall time complexity of the algorithm is

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

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

Lemma 4.17 $\textsc{Init}^b$ is in $O(rm + n)$.

Proof The relation $\mathcal{P}_0$ can clearly be initialised in $O(n)$, and by Lemma 4.18 the initialisation of $\mathcal{R}_0$ to $\mathcal{P}_0 \setminus \mathit{split}^b(\mathcal{L}_0)$ is in $O\big(r\,|\mu^b_{\mathcal{L}_0}|\big) = O(rm)$. ■

Lemma 4.18 Let $\mathcal{R}$ and $\mathcal{P}$ be equivalence relations on Q. The computation of $\mathcal{R} \backslash \mathit{split}^b(L)$, where $L \subseteq (Q/\mathcal{P})^*$, is in $O\big(r\,|\mu^b_L|\big)$.

Proof Let $L_{(k)} = \{w \in L \mid w \text{ is of length } k\}$. We begin the computation by adjusting the value of $\mathit{obsb}^k_q(\sigma, [q_1]_{\mathcal{P}_{i+1}} \cdots [q_k]_{\mathcal{P}_{i+1}})$ for each $\mu_k(\sigma)_{q_1 \cdots q_k, q}$ in μ^b_L, at the cost of $O\big(r\,\big|\mu^b_L\big|\big)$ computation steps. The elements of $\mathcal{R}_i$ that are to be retained in $\mathcal{R}_{i+1}$ can thereafter be determined by hashing each state q in Q using $(\mathit{obsb}^k_q)_{k \in \mathbb{N}}$ on the domain $\cup_{k \in [0,r]}(\Sigma_{(k)} \times L_{(k)})$ as key, and then keeping those pairs of states that end up at the same memory address.

We now make the observation that if the pair $\sigma \in \Sigma_{(k)}$ and $D_1 \cdots D_k \in L_{(k)}$ is in the support of obsb^k_q for some $q \in Q$, then there is an entry $\mu_k(\sigma)_{q_1 \cdots q_k}$ in μ such that $q_1 \cdots q_k \in D_1 \cdots D_k$. In other words, size of the support of $(\mathit{obsb}^k_q)_{q \in Q, k \in \mathbb{N}}$ on the domain $\cup_{k \in [0,r]}(\Sigma_{(k)} \times L_{(k)})$ does not exceed the size of μ^b_L. We conclude that the total time required to hash the states is determined by the time it takes to calculate their hash key. ■

Lemma 4.19 Split^b_i is in $O\big(r\,|\mu^b_{\mathcal{L}_i(S_i)}|\big)$.

Proof By Lemma 4.18, we can split against B_i in time $O(r\,|\mu^b_{\mathcal{L}_{i+1}(B_i)}|)$, and against $S_i \setminus B_i$ in $O(r\,|\mu^b_{\mathcal{L}_{i+1}(S_i \setminus B_i, \neg B_i)}|)$. Since $|\mu^b_{\mathcal{L}_i(S_i)}| = |\mu^b_{\mathcal{L}_{i+1}(S_i \setminus B_i, \neg B_i)}| + |\mu^b_{\mathcal{L}_{i+1}(B_i)}|$, the overall time complexity of Split^b_i becomes $O\big(r\,|\mu^b_{\mathcal{L}_i(S_i)}|\big)$. ■

Lemma 4.20 Aggregate^b is in $O(m + n)$.

We omit the proofs of Lemma 4.20 and Theorem 4.21, as they are quite similar those of Lemma 3.27 and Theorem 3.28, respectively.

Theorem 4.21 The backward bisimulation algorithm is in $O(rmn)$.

4.2.3 Additively cancellative semirings

In this section we again consider an optimisation for additively cancellative semirings. Thus, for the remainder of this section, let $\mathcal{A}$ be an additively cancellative semiring.

Lemma 4.22 Without effect on the overall algorithm we can compute $\mathcal{R}_{i+1}$ as $\mathcal{R}_i \backslash \mathit{split}^b(\mathcal{L}_{i+1}(B_i))$.

Proof We prove, through induction on i, that

$$\mathcal{R}_i \setminus \mathit{split}^b(\mathcal{L}_{i+1}(B_i)) = (\mathcal{R}_i \setminus \mathit{split}^b(\mathcal{L}_{i+1}(B_i))) \setminus \mathit{split}^b(\mathcal{L}_{i+1}(S_i \setminus B_i, \neg B_i)) \enspace .$$

Clearly, an element from the right-hand side is also in the left-hand side. It remains to show that every (p, q) in the left-hand side is also in the right-hand side. Suppose that (p, q) is in $\mathcal{R}_i$ but not in $\mathit{split}^b(\mathcal{L}_{i+1}(B_i))$. This immediately yields that $\sum_{w \in L} \mu_k(\sigma)_{w,p} = \sum_{w \in L} \mu_k(\sigma)_{w,q}$ for every $\sigma \in \Sigma_{(k)}$ and language L in $\mathcal{L}_{i+1}(B_i) \cap \mathfrak{P}(Q^k)$.

By Lemma 4.13 and the induction hypothesis, $\mathcal{R}_i$ is stable with respect to $\mathcal{P}_i$. From this, we conclude that $\sum_{w \in L} \mu_k(\sigma)_{w,p} = \sum_{w \in L} \mu_k(\sigma)_{w,q}$, for every $\sigma \in \Sigma_{(k)}$ and L in $\mathcal{L}_i \cap \mathfrak{P}(Q^k)$.

Let σ be a symbol in $\Sigma_{(k)}$ and $L = D_1 \times \cdots \times D_k$ be a language in $\mathcal{L}_{i+1}(S_i \setminus B_i, \neg B_i)$ for some $D_1, \ldots, D_k \in (Q/\mathcal{P}_{i+1})$. Clearly, none of the blocks $D_1, \ldots, D_k$ equals B_i. Thus for each $j \in [1, k]$

we have that either $D_j = S_i \setminus B_i$ or $D_j \in (Q/\mathcal{P}_i)$. Let $J = \{j \in [1,k] \mid D_j = S_i \setminus B_i\}$ be the set of those indices where the block $S_i \setminus B_i$ appears.

Let $L' = D'_1 \times \cdots \times D'_k$, and for every $j \in J$ let $L''_j = D''_{j,1} \times \cdots \times D''_{j,k}$ where

$$D'_n = \begin{cases} S_i & \text{if } D_n = S_i \setminus B_i \\ D_n & \text{otherwise} \end{cases} \quad \text{and} \quad D''_{j,n} = \begin{cases} B_i & \text{if } j = n \\ D_n & \text{otherwise} \end{cases}$$

for every $i \in [1,k]$. By definition, L' is a language of $\mathcal{L}_i(S_i) \cap \mathfrak{P}(Q^k)$ and L''_j is a language of $\mathcal{L}_{i+1}(B_i) \cap \mathfrak{P}(Q^k)$ for every $j \in J$. Moreover, we observe that $\{L\} \cup \{L''_j \mid j \in J\}$ is a partition of L'. Thus, in particular, $L' = L \cup \bigcup_{j \in J} L''_j$.

$$\Big(\sum_{w \in L} \mu_k(\sigma)_{w,p}\Big) + \sum_{j \in J}\Big(\sum_{w \in L''_j} \mu_k(\sigma)_{w,p}\Big) = \sum_{w \in L'} \mu_k(\sigma)_{w,p}$$

$$= \sum_{w \in L'} \mu_k(\sigma)_{w,q} = \Big(\sum_{w \in L} \mu_k(\sigma)_{w,q}\Big) + \sum_{j \in J}\Big(\sum_{w \in L''_j} \mu_k(\sigma)_{w,q}\Big) \qquad \text{(because } L' \in \mathcal{L}_i \cap \mathfrak{P}(Q^k))$$

Since we additionally have that $\sum_{w \in L''_j} \mu_k(\sigma)_{w,p}$ is equal to $\sum_{w \in L''_j} \mu_k(\sigma)_{w,q}$ for every $j \in J$ because $L''_j \in \mathcal{L}_{i+1}(B_i) \cap \mathfrak{P}(Q^k)$, we can apply the cancellation law to obtain that $\sum_{w \in L} \mu_k(\sigma)_{w,p}$ is equal to $\sum_{w \in L} \mu_k(\sigma)_{w,q}$. Consequently, $\mathit{split}^b(\mathcal{L}_{i+1}(S_i \setminus B_i, \neg B_i))$ does not contain (p,q), so the statement is proved. ■

Time complexity We now consider the time complexity for the optimised backward algorithm.

Theorem 4.23 The backward minimisation algorithm is in $O\big(r^2 m \log n\big)$.

Proof By Lemmata 4.17 and 4.18, time complexity of the algorithm can be written as

$$O\Big(m + \sum_{i \in [0,t-1]} 1 + |B_i| + r\,|\mu^b_{\mathcal{L}_{i+1}(B_i)}|\Big) \ .$$

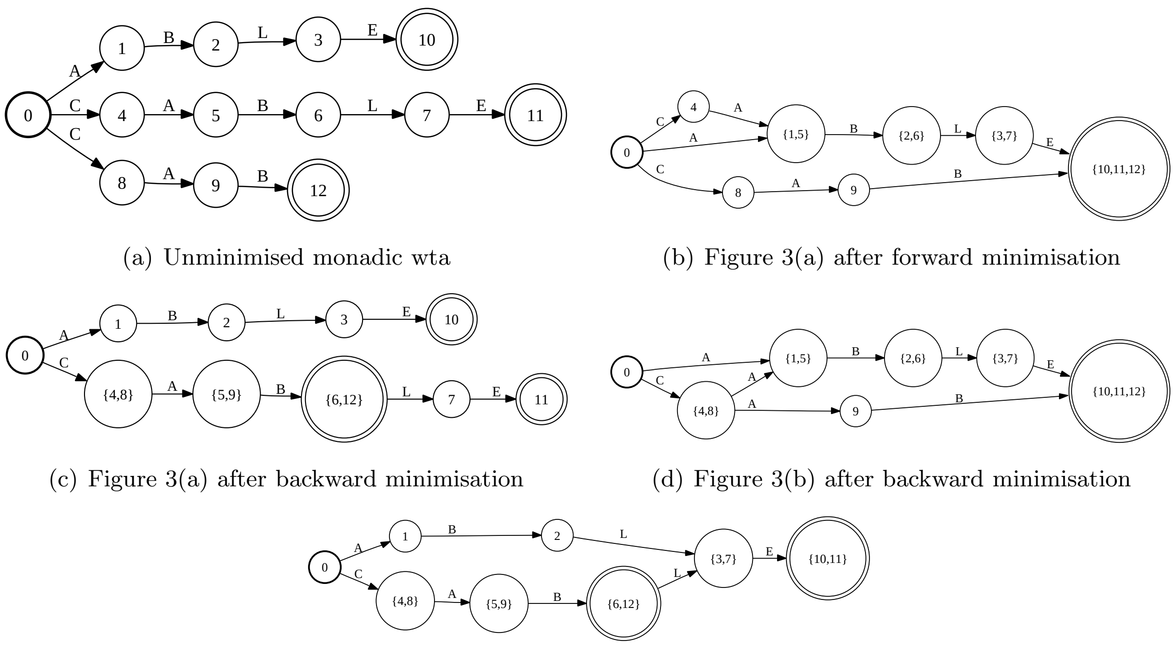
Omitting the smaller terms and reordering, we obtain $r \sum_{i \in [0,t-1]} |\mu^b_{\mathcal{L}_{i+1}(B_i)}|$.

By Lemma 3.31, no state occurs in more than $\log n$ distinct B-blocks. An entry $\mu_k(\sigma)_{q_1 \cdots q_k, q}$ in μ will only be contained in $\mu^b_{\mathcal{L}_{i+1}(B_i)}$ if $q_j \in B_i$ for some $j \in [1,k]$, so no entry will contribute by more $r \log n$ to the sum. The expression hence simplifies to $O\big(r^2\, m \log n\big)$. ■

4.3 Sparse tree representations

When the tree representation μ is very sparse, it is more efficiently represented by a list of tuples over $\Sigma \times Q^* \times Q \times A$ than by a matrix. Although the size of this representation of μ is $m' = (r+3)m$, it is still preferable to the matrix representation, which can be exponentially large in size.

In case of forward bisimulation, there is however an upside to the list representation: since we must now read the entire sequence of states $q_1, \ldots, q_k, q$ every time we process an entry $\mu_k(\sigma)_{q_1 \cdots q_k, q}$, it matters little that we also have to update one counter per state. This means that with the list representation the time complexity of the forward algorithm remains in $O(m'n) = O(rmn)$, and in $O(m' \log n) = O(rm \log n)$ if the underlying semiring is additively cancellative.



(a) Unminimised monadic wta

(b) Figure 3(a) after forward minimisation

(c) Figure 3(a) after backward minimisation

(d) Figure 3(b) after backward minimisation

(e) Figure 3(c) after forward minimisation

Figure 3: wsa interpretation of a fta and several minimisations of it

Also in case of backward bisimulation we see this effect. Reading the state sequence of an entry adds another $O(r)$ computation steps, but these do not dominate the amount of work that we do at an entry, so the backward algorithm is in $O(m'n) = O(rmn)$, and in $O(rm' \log n) = O\big(r^2 m \log n\big)$ if the underlying semiring is cancellative.

5 Iterative Application

In the following example we demonstrate the different nature of the forward and backward algorithms by considering how different sequences of repeated applications converge to different backward and forward bisimulation minimal wta. To motivate this example we consider the application of *dictionary compression.* We wish to construct an automaton that recognises the strings “ABLE”, “CABLE”, and “CAB” in as little space (counting rules and states) as possible.

Example 5.1 Consider the wta $M = (Q, \Sigma, \mathbb{B}, F, \mu)$ with

- $Q = [1, 12]$;
- $\Sigma = \Sigma_{(1)} \cup \Sigma_{(0)}$ with $\Sigma_{(1)} = \{E, L, B, A\}$ and $\Sigma_{(0)} = \{\underline{A}, C\}$;
- $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$ with the usual operations of “or” $\vee$ and “and” $\wedge$;
- $F(q) = 1$ for every $q \in [10, 12]$ and $F(q) = 0$ for every $q \in [1, 9]$;

- the non-zero tree representation entries

$$\begin{aligned}1 &= \mu_0(\underline{A})_{\varepsilon,1} = \mu_1(B)_{1,2} = \mu_1(L)_{2,3} = \mu_1(E)_{3,10}\\ 1 &= \mu_0(C)_{\varepsilon,4} = \mu_1(A)_{4,5} = \mu_1(B)_{5,6} = \mu_1(L)_{6,7} = \mu_1(E)_{7,11}\\ 1 &= \mu_0(C)_{\varepsilon,8} = \mu_1(A)_{8,9} = \mu_1(B)_{9,12}\ .\end{aligned}$$ □

Let us discuss the introduced automaton im more detail (note that we will no longer distinguish between the unary A and the nullary $\underline{A}$). The automaton recognises monadic trees whose labels, strung together, form the desired strings. As noted in Section 1.2, wta generalise weighted string automata, of which M is an example. By introducing a start state 0, from which all nullary transitions lead away, we can represent the wsa form of M in a canonical graphical representation that is easy to understand, as in Figure 3(a). The initial automaton has 12 states and 12 arcs (we ignore the artificially inserted start state). If we apply Algorithm 1 common suffix paths and states are merged (in this case, the "-BLE" in "ABLE" and "CABLE"), resulting in the automaton depicted in Figure 3(b), which has 7 states and 9 arcs. If we instead apply the backward minimisation described in Section 4.2, common prefix paths and states are merged (i.e. "CAB" and "CABLE"), resulting in the automaton depicted in Figure 3(c), which has 9 states and 9 arcs. We can then apply alternating algorithms on the resultant automata, as depicted in Figures 3(d) and 3(e) until no further reduction is possible. Since the order of minimisation (i.e. forward-backward-... vs. backward-forward-...) causes different state mergings which cannot later be undone, there is no guarantee that a particular ordering will result in the smallest obtainable automaton.

6 Implementation

In this section we present some experimental results that we obtained by applying an implementation of Algorithms 1 and 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 many 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. Language models are typically 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 [20, 26]. 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.

However, while classical language models use as their subsequence structure a substring of some chosen window of words (generally referred to as an *n-gram*), recent research in natural language processing has focused on using tree-based models to capture syntactic dependencies in applications such as machine translation [13, 37]. We thus require a language model of trees, and the subsequences we will represent are subtrees. We thus also require the tree automata minimisation algorithms presented in this work.

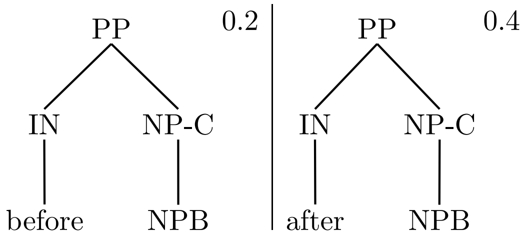


Figure 4: Trees represented in Example 6.1

6.1 Experiment

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 [27]. An initial wta was constructed by representing each 3-subtree in a single path. Example 6.1 demonstrates how such initial wtas are constructed.

Example 6.1 Let the (tree, weight) pairs in Figure 6.1 be those we want to represent in a wta. Then we construct the wta $M = (Q, \Sigma, \mathbb{R}, F, \mu)$, where

- $Q = \{a, b, c, d, e, f, g, h, i, j\}$;
- $\Sigma = \Sigma_{(2)} \cup \Sigma_{(1)} \cup \Sigma_{(0)}$ with $\Sigma_{(2)} = \{\text{PP}\}$, $\Sigma_{(1)} = \{\text{IN}, \text{NP-C}\}$, and $\Sigma_{(0)} = \{\text{before}, \text{NPB}, \text{after}\}$;
- $\mathbb{R} = (\mathbb{R}, +, \cdot, 0, 1)$ is the field of reals;
- $F(i) = 0.2$, $F(j) = 0.4$, and $F(x) = 0$ for every $x \in Q \setminus \{i, j\}$;
- the only non-zero tree representations are

$$\begin{aligned} 1 &= \mu_0(\text{before})_{\varepsilon,a} = \mu_0(\text{after})_{\varepsilon,b} = \mu_0(\text{NPB})_{\varepsilon,c} = \mu_0(\text{NPB})_{\varepsilon,d} \\ 1 &= \mu_1(\text{IN})_{a,e} = \mu_1(\text{IN})_{b,f} = \mu_1(\text{NP-C})_{c,g} = \mu_1(\text{NP-C})_{d,h} \\ 1 &= \mu_2(\text{PP})_{eg,i} = \mu_2(\text{PP})_{fh,j} \ . \end{aligned}$$ □

We then wrote an implementation of the forward and backward variants of Algorithm 1 in Perl and applied them to wta created from data sets of various sizes of 3-subtrees. Example 6.2 demonstrates the result of applying the backward algorithm to Example 6.1. In this case, the forward algorithm would not change the size of the input automaton, but this is of course not true in the general case.

Example 6.2 Let M be the wta described in Example 6.1. Then $M' = (P, \Sigma, \mathbb{R}, G, \nu)$ is the wta obtained by applying Algorithm 2 to M where Σ and $\mathbb{R}$ are as before and

- $P = \{a, b, (cd), e, f, (gh), i, j\}$;
- $G(i) = 0.2$, $G(j) = 0.4$, and $G(x) = 0$ for every $x \in P \setminus \{i, j\}$;
- the only non-zero tree representation entries are

$$\begin{aligned} 1 &= \nu_0(\text{before})_{\varepsilon,a} = \nu_0(\text{after})_{\varepsilon,b} = \nu_0(\text{NPB})_{\varepsilon,(cd)} \\ 1 &= \nu_1(\text{IN})_{a,e} = \nu_1(\text{IN})_{b,f} = \nu_1(\text{NP-C})_{(cd),(gh)} \\ 1 &= \nu_2(\text{PP})_{e(gh),i} = \nu_2(\text{PP})_{f(gh),j} \ . \end{aligned}$$ □

As noted in Section 5, automata can be minimised by iteratively running Algorithms 1 and 2 in alternating succession until no changes are observed. However, although the choice of initial minimisation can affect the end automaton, we found the choice of initial minimisation to not have a great impact on the final automata in our experiments. In the 31 experimental setups summarised below in Table 1, only 4 yielded a different automaton depending on the initial minimisation, and these differences were never more than 2 rules or states off. We thus indicate in Table 1, for each experiment, the number of states and rules initially, after one iteration of forward minimisation, after one iteration of backward minimisation, and after convergence, which was generally achieved after the sequence (backward, forward), equivalent to the sequence (forward, backward, forward). The differences between the two sequences, if any, were inconsequential.

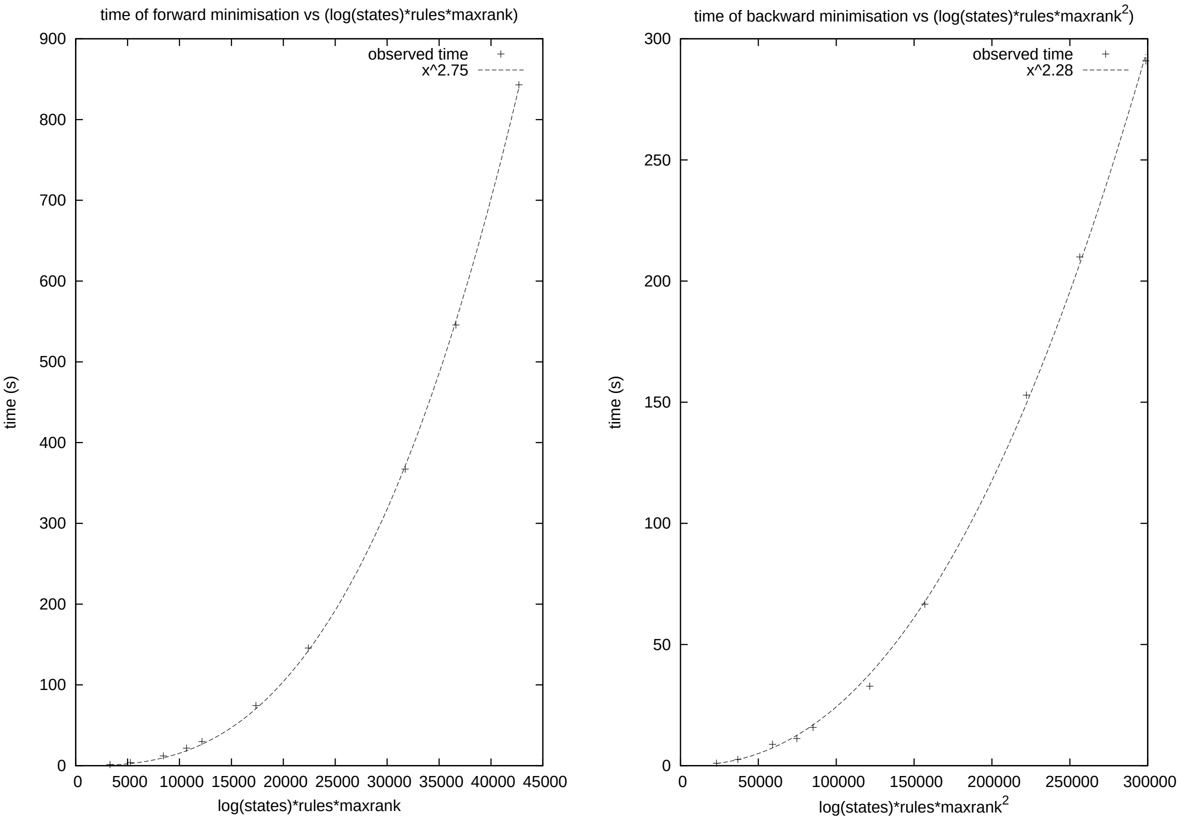
6.2 Performance

As noted in Section 4.2.3, the time complexity of the forward algorithm for cancellative semirings is $O(rm \log n)$ and the complexity for the backward algorithm for cancellative semirings is $O\big(r^2 m \log n\big)$. To determine the empirical runtime of our implementation, we constructed several automata of increasing size and ran both minimisation algorithms on them, noting the time to minimise. We summarise the empirical runtimes observed in our experiment in Figure 5 by plotting the appropriate dependency (i.e. $rm \log n$ for forward and $r^2 m \log n$ for backward) vs. the time and fitting the curve ax^b to the plot, then determining the value of b. As the figure shows, the observed time exceeds the ideal time. This is due to the invalidity of the perfect hash assumption described in Section 3.2.2.

Acknowledgements: The authors truly appreciate the help of Lisa Kaati, who provided excellent references and test data for our implementation. Further we would like to thank Frank Drewes for proof-reading the manuscript and suggesting numerous improvements and Kevin Knight for suggestions regarding the implementation section. The remaining errors and imperfections are solely our fault.

TREES	ORIGINAL		FORWARD		BACKWARD		CONVERGENCE	
	states	rules	states	rules	states	rules	states	rules
5	23	23	18	22	21	21	16	20
15	88	88	74	87	70	70	56	69
25	162	162	141	161	136	136	115	135
35	246	246	210	243	191	191	155	188
45	295	295	248	290	209	209	161	203
55	357	357	304	353	244	244	189	238
65	424	424	365	421	303	303	242	298
75	507	507	431	501	340	340	259	329
85	526	526	436	516	365	365	271	351
95	614	614	518	603	430	430	331	416
105	707	707	598	694	475	475	361	457
115	708	708	571	679	484	484	343	451
125	803	803	670	788	526	526	380	498
135	843	843	699	822	553	553	403	526
145	915	915	759	893	594	594	415	549
155	1000	1000	834	979	631	631	455	600
165	1087	1087	899	1054	672	672	468	623
175	1118	1118	924	1089	689	689	474	639
185	1189	1189	984	1157	772	772	542	715
195	1238	1238	1025	1210	768	768	522	707
205	1366	1366	1130	1324	840	840	575	769
215	1349	1349	1094	1304	834	834	549	759
225	1470	1470	1219	1432	910	910	628	841
235	1448	1448	1186	1410	867	867	567	791
245	1593	1593	1312	1546	947	947	637	871
255	1621	1621	1328	1567	935	935	605	844
265	1694	1694	1400	1650	983	983	652	902
275	1738	1738	1419	1682	1039	1039	687	950
285	1838	1838	1513	1786	1056	1056	690	963
295	1922	1922	1594	1876	1116	1116	748	1030
305	1996	1996	1630	1924	1143	1143	735	1029

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



(a) Forward minimisation timing statistics

(b) Backward minimisation timing statistics

Figure 5: Time to minimise vs. rule size

References

[1] P. A. Abdulla, L. Kaati, and J. Högberg. Bisimulation minimization of tree automata. In *Proc. 11th Int. Conf. Implementation and Application of Automata*, volume 4094 of *LNCS*, pages 173–185. Springer Verlag, 2006.

[2] P. A. Abdulla, L. Kaati, and J. Högberg. Bisimulation minimization of tree automata. Technical Report UMINF 06.25, Umeå University, 2006.

[3] S. Bangalore and G. Riccardi. Stochastic finite-state models for spoken language machine translation. *Machine Translation*, 17(3):165–184, 2002.

[4] J. Berstel and C. Reutenauer. Recognizable formal power series on trees. *Theoretical Computer Science*, 18(2):115–148, 1982.

[5] B. Borchardt. The Myhill-Nerode theorem for recognizable tree series. In *Proc. 7th Int. Conf. Developments in Language Theory*, volume 2710 of *LNCS*, pages 146–158. Springer Verlag, 2003.

[6] B. Borchardt and H. Vogler. Determinization of finite state weighted tree automata. *Journal of Automata, Languages and Combinatorics*, 8(3):417–463, 2003.

[7] Björn Borchardt. *The Theory of Recognizable Tree Series.* Akademische Abhandlungen zur Informatik. Verlag für Wissenschaft und Forschung, 2005.

[8] S. Bozapalidis. Effective construction of the syntactic algebra of a recognizable series on trees. *Acta Informatica*, 28:351–363, 1991.

[9] S. Bozapalidis and A. Alexandrakis. Representation matricielles de series d'arbre reconnaissables. *Theoretical Informatics and Applications*, 23(4):449–459, 1989.

[10] P. Buchholz. Bisimulation relations for weighted automata. unpublished, 2007.

[11] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata: Techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997.

[12] F. Drewes and H. Vogler. Learning deterministically recognizable tree series. *J. Automata, Languages and Combinatorics*, 2007. to appear.

[13] M. Galley, M. Hopkins, K. Knight, and D. Marcu. What's in a translation rule? In *Proc. 2004 Human Language Technology Conf. of the North American Chapter of the Association for Computational Linguistics*, pages 273–280, 2004.

[14] F. Gécseg and M. Steinby. *Tree Automata.* Akadémiai Kiadó, 1984.

[15] F. Gécseg and M. Steinby. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer Verlag, 1997.

[16] G. Gramlich and G. Schnitger. Minimizing nfas and regular expressions. In *Proc. 22nd Int. Symp. Theoretical Aspects of Computer Science*, volume 3404 of *LNCS*, pages 399–411. Springer Verlag, 2005.

[17] J. Högberg, A. Maletti, and J. May. Backward and forward bisimulation minimisation of tree automata. Technical Report ISI-TR-633, University of Southern California, 2007.

[18] J. E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Z. Kohavi, editor, *Theory of Machines and Computations*. Academic Press, 1971.

[19] F. Jelinek. Continuous speech recognition by statistical methods. *Proc. IEEE*, 64(4):532–557, 1976.

[20] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*, chapter 6. Prentice Hall, 2000.

[21] K. Knight and J. Graehl. An overview of probabilistic tree transducers for natural language processing. In *Proc. 6th Int. Conf. Computational Linguistics and Intelligent Text Processing*, volume 3406 of *LNCS*, pages 1–24. Springer Verlag, 2005.

[22] D. Kozen. On the Myhill-Nerode theorem for trees. *Bulletin of the EATCS*, 47:170–173, 1992.

[23] W. Kuich. Formal power series over trees. In S. Bozapalidis, editor, *Proc. 3rd Int. Conf. Developments in Language Theory*, pages 61–101. Aristotle University of Thessaloniki, 1997.

[24] S. Kumar and W. Byrne. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proc. 2003 Human Language Technology Conf. of the North American Chapter of the Association for Computational Linguistics*, pages 63–70, 2003.

[25] A. Maletti. Myhill-Nerode theorem for recognizable tree series — revisited. unpublished, 2007.

[26] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*, chapter 6. MIT Press, 1999.

[27] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[28] J. May and K. Knight. Tiburon: A weighted tree automata toolkit. In *Proc. 11th Int. Conf. Implementation and Application of Automata*, volume 4094 of *LNCS*, pages 102–113. Springer Verlag, 2006.

[29] Jonathan May and Kevin Knight. A better n-best list: Practical determinization of weighted finite tree automata. In *Proc. 2006 Human Language Technology Conf. of the North American Chapter of the Association for Computational Linguistics*, pages 351–358, 2006.

[30] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. 13th Annual Symp. Foundations of Computer Science*, pages 125–129. IEEE Computer Society, 1972.

[31] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[32] M. Mohri, F. Pereira, and M. Riley. A rational design for a weighted finite-state transducer library. In *Proc. 2nd Int. Workshop Implementing Automata*, volume 1436 of *LNCS*, pages 144–158. Springer Verlag, 1997.

[33] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88, 2002.

[34] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[35] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[36] F. Pereira and M. D. Riley. Speech recognition by composition of weighted finite automata. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing.* MIT Press, 1997.

[37] K. Yamada and K. Knight. A syntax-based statistical translation model. In *Proc. 39th Meeting of the Association for Computational Linguistics*, pages 523–530. Morgan Kaufmann, 2001.

[38] B. Zhou, S. Chen, and Y. Gao. Folsom: A fast and memory-efficient phrase-based approach to statistical machine translation. In *Poster Proc. IEEE/ACL Workshop on Spoken Language Technology*, 2006.