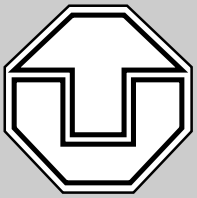
TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik



Technische Berichte
Technical Reports

ISSN 1430-211X

TUD-FI08-03 — März 2008

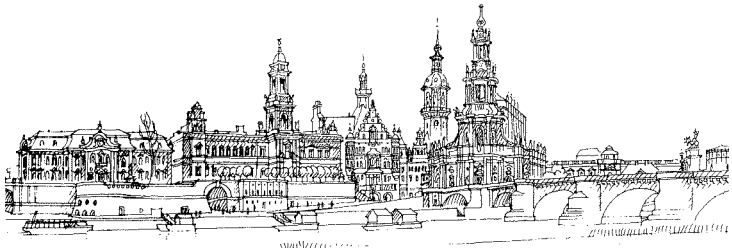
J. Högberg[1], A. Maletti[2], H. Vogler[3]

[1] **Department of Computing Science, Umeå University**

[2] **Int. Computer Science Institute, Berkeley**

[3] **Inst. für Theoretische Informatik, TU Dresden**

Bisimulation Minimisation of Weighted Automata on Unranked Trees



Technische Universität Dresden
Fakultät Informatik
D-01062 Dresden
Germany

URL: http://www.inf.tu-dresden.de/

Bisimulation Minimisation of Weighted Automata on Unranked Trees

Johanna Högberg[1] Andreas Maletti[2] Heiko Vogler[3]

[1] Department of Computing Science, Umeå University
S-90187 Umeå, Sweden

[2] International Computer Science Institute
1947 Center Street, Suite 600, CA-94704 Berkeley, USA

[3] Faculty of Computer Science, Technische Universität Dresden
D-01062 Dresden, Germany

March 3, 2008

Abstract

Two examples of automata-theoretic models for the validation of xml documents against user-defined schema are the stepwise unranked tree automaton (suta) and the parallel unranked tree automaton (puta). By adding a weight, taken from some semiring, to every transition we generalise these two qualitative automata models to quantitative models, thereby obtaining weighted stepwise unranked tree automata (wsuta) and weighted parallel unranked tree automata (wputa); the qualitative automata models are reobtained by choosing the BOOLEAN semiring. We deal with the minimisation problem of wsuta and wputa by using (forward and backward) bisimulations and we prove the following results: (1) for every wsuta an equivalent forward (resp. backward) bisimulation minimal wsuta can be computed in time $O(mn)$ where n is the number of states and m is the number of transitions of the given wsuta; (2) the same result is proved for wputa instead of wsuta; (3) if the semiring is additive cancellative or the BOOLEAN semiring, then the bound can be improved to $O(m \log n)$ for both wsuta and wputa; (4) for every deterministic puta we can compute a minimal equivalent deterministic puta in time $O(m \log n)$; (5) the automata models wsuta, wputa, and weighted unranked tree automaton have the same computational power.

1 Introduction

A recent product of language technology is the *extensible markup language* (xml), a tag-based language for structuring document data [2, 24]. In xml there is no fixed set of available tags. These are instead defined through customisable

xml schemas to suit the situation at hand. An xml document that complies with a particular schema is *valid* with respect to it. There is, at present, a number of verification tools designed to validate xml documents against a user-provided schema. Some of these use unranked tree automata as an internal representation for the target language: an input document d is only approved if the internal automaton accepts the tree structure contained in d. Because validation is an operation that is frequently used, it is important that it can be done in a time- and memory-efficient way. For this reason the automaton that carries out the verification should be as small as possible. A minimisation algorithm for unranked tree automata would thus be a useful instrument to tune the performance of xml validators.

Several models for automata on unranked trees have been described in the literature. The most prominent of these are probably the *unranked tree automaton* (uta) [6, 29], the *stepwise unranked tree automaton* (suta) [8], and the *parallel unranked tree automaton* (puta) [9]. Minimisation of bottom-up and left-to-right deterministic versions of uta, suta, and puta are investigated in [26]. MARTENS and NIEHREN show that there is no unique minimal deterministic uta recognising a given language, and finding any minimal deterministic uta is NP-complete, whereas deterministic suta and puta allow efficient minimisation. Regarding representation size, it is stated that a deterministic uta can always be converted into an equivalent deterministic puta, but not without risking an exponential blow-up. In addition, deterministic suta are known to be more succinct than deterministic puta, but only quadratically so.

In this paper, we consider the minimisation problem for, not necessarily deterministic, suta and puta. As unranked tree automata generalise both ranked tree automata (fta) [15, 16] and finite-state string automata (fsa), many results about minimisation carry over immediately. For instance, it is well-known that there is, in general, no unique solution to the minimisation problem for fsa. There can be several minimal, but pairwise non-isomorphic, fsa that recognise a given regular language. Furthermore, minimising fsa is PSPACE-complete [27, 23, 25], and the minimisation problem for fsa with n states cannot even be efficiently approximated within the factor $o(n)$ unless P = PSPACE [18]. In other words, supposing that P $\neq$ PSPACE we cannot find a natural number c such that there is an algorithm with polynomial run-time that, for every fsa, returns an equivalent fsa of size at most c times the size of a minimal equivalent fsa.

As these statements also hold for automata on unranked trees, we see that any algorithmic solution to the minimisation problem for this family of devices must be a heuristic. The type of heuristic that we have chosen to work with here is called *bisimulation*. Bisimulation equivalence, as interpreted for various devices, typically implies language equality (it does in the cases considered here), however the opposite does not hold in general [28]. So-called *backward* and *forward bisimulation* is introduced in [7] for weighted string automata (wsa) and generalised to fta [19] and weighted tree automata (wta) [21].

Also in this paper we choose to work with weighted automata. These gener-

alise the BOOLEAN (i.e. the unweighted) case, but are arguably more useful, as they let us express not only qualitative but also quantitative properties such as probabilities and costs. Originally, wta [4] used fields as weight structure, but they were later generalised to semirings [3]. The wta generalises the fta and the wsa. We take this generalisation one step further, as the weighted versions of the uta, the suta, and the puta (wuta, wsuta, and wputa, respectively) that are presented in this paper all generalise the wta.

Let us attempt an intuitive description of these weighted devices. Suppose that Q is a finite set of states and Σ is an input alphabet. A common trait is that they all parse their input tree bottom-up, one node at a time. The weight c with which a state q is assigned to an f-labelled position w in the input tree is obtained by running a wsa on the sequence of states $q_1 \cdots q_k$ that have already been assigned to the positions $w1, \ldots, wk$ immediately below w. How c is computed differs between the models:

- A wuta has one wsa $A_{(p,f')}$ for every $p \in Q$ and $f' \in \Sigma$. It computes c by running $A_{(q,f)}$ on $q_1 \cdots q_k$.
- A wputa has one wsa $A_{f'}$ with Q-output for every $f' \in \Sigma$. It computes c by taking the q-component of the vector that results when A_f is run on $q_1 \cdots q_k$.
- A wsuta has only a single wsa, but it has one initial vector $\lambda_{f'}$ for every $f' \in \Sigma$. It computes c as the weight with which the state q is reached on the word $q_1 \cdots q_k$ using the initial vector λ_f.

When comparing the weighted unranked tree automata models we find that wuta, wputa, and wsuta are all equally powerful (see Theorem 6.6). The construction that converts a wuta to a wputa merely re-groups the states and transitions of the wuta and is thus size-preserving. When converting a wsuta to a wuta we make one copy of its transition graph for every state and input symbol, increasing the size by a factor $n|\Sigma|$ where n is the number of states. Changing a wputa into a wsuta requires $O(mn)$ computation steps, where m is the number of transitions and n is the number of states, but the construction preserves the size.

The main contribution of this paper is bisimulation minimisation for wsuta and wputa, which is obtained by reducing the problem to bisimulation minimisation for wta (see [21, Theorems 11–29] and [20, Theorems 12–27]). We show that for every wsuta an equivalent forward (resp., backward) bisimulation minimal wsuta can be computed in time $O(mn)$ [see Theorem 4.9]. If the used semiring is additively cancellative (i.e., $a + b = a + c$ implies that $b = c$ for all a, b, c) or the BOOLEAN semiring (i.e., the wsuta is essentially a suta), then we can improve the running time to $O(m \log n)$. In particular, for every deterministic suta, we can use the forward bisimulation minimisation algorithm to compute an equivalent minimal deterministic suta in time $O(m \log n)$ [see Corollary 4.10]. It is known [26, 19] that the minimisation problem for deterministic suta can be solved in $O(m \log n)$. Similarly, for every wputa M an equivalent forward (resp., backward) bisimulation minimal wputa can be computed in time $O(mn)$

[see Theorems 5.15 and 5.10, respectively], which can be improved to $O(m \log n)$ for additively cancellative semirings and the BOOLEAN semiring. Moreover, it is known [9] that deterministic puta can be minimised in quadratic time. We improve this result by showing that for every deterministic puta we can construct a minimal equivalent deterministic puta in time $O(m \log n)$ [see Theorem 5.16].

The results of [21, 20] concerning bisimulation minimisation of wta are readily transferable to wsuta since wsuta can be seen as wta over a special encoding (cf. [26]) of the input tree. Due to the similarity between wsuta and wputa, the same results can also be adopted for wputa, albeit with a bit more work. The situation is different for uta (and hence its generalisation wuta), as here the deterministic case does not submit to MYHILL-NERODE flavoured minimisation. The reason is, as pointed out in [26], that a deterministic uta as defined in [26] is, although unambiguous, not truly deterministic. Hence we cannot hope to obtain an efficient minimisation algorithm for deterministic uta with the help of bisimulation minimisation. However, it might still be possible to obtain efficient forward and backward bisimulation minimisation algorithms, but we leave this as an open problem.

Outline. In Section 2 we give the technical preliminaries needed to make this paper self-contained. For the same reason we also restate adopted versions of the main theorems on bisimulation of wta [19, 21] in Section 3. Section 4 and Section 5 treat bisimulation minimisation of wsuta and wputa, respectively, and are followed by Section 6 in which it is shown that wuta, wputa, and wsuta are equally expressive.

2 Preliminaries

Sets, numbers, and relations. We denote the set of all reals by $\mathbb{R}$ and the set $\mathbb{R} \cup \{\kappa\}$ by $\mathbb{R}^\kappa$ for every $\kappa \notin \mathbb{R}$. The set of all natural numbers including 0 (resp., excluding 0) is denoted by $\mathbb{N}$ (resp., $\mathbb{N}_+$). The subset $\{1, 2, \ldots, k\}$ of $\mathbb{N}_+$ is abbreviated by $[k]$. Note that $[0] = \emptyset$. An *alphabet* Σ is a finite nonempty set of symbols. We denote the empty string by ε and the length of a string $w \in \Sigma^*$ by $|w|$. Let $w = f_1 \cdots f_k$ with $f_i \in \Sigma$ for every $i \in [k]$. The *label* at position $i \in [k]$ of w is $w(i) = f_i$.

Let $\mathcal{P}$ and $\mathcal{R}$ be equivalence relations on a set Q. 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 $q \in Q$ with respect to $\mathcal{R}$ is the set $[q]_\mathcal{R} = \{p \mid (q, p) \in \mathcal{R}\}$. Whenever $\mathcal{R}$ is obvious from the context, we simply write $[q]$ instead of $[q]_\mathcal{R}$. It should be clear that $[q]$ and $[p]$ are equal if $(q, p) \in \mathcal{R}$, and disjoint otherwise, so $\mathcal{R}$ induces a partition $Q/\mathcal{R} = \{[q] \mid q \in Q\}$ of Q.

Algebraic structures. Let K be a nonempty set, and let $\cdot$ be an associative binary operation on K. If K contains an element 1 such that $1 \cdot \kappa = \kappa = \kappa \cdot 1$ for every $\kappa \in K$, then $(K, \cdot)$ is a *monoid* with *identity* 1. A monoid $(K, \cdot)$ is

commutative if the equation $\kappa_1 \cdot \kappa_2 = \kappa_2 \cdot \kappa_1$ holds for every $\kappa_1, \kappa_2 \in K$. We henceforth adopt the convention of identifying an algebraic structure with its carrier set.

A *commutative semiring* is a nonempty set K on which a binary addition $+$ and a binary multiplication $\cdot$ have been defined such that the following conditions are satisfied:

- $(K, +)$ and $(K, \cdot)$ are commutative monoids with identities 0 and 1, respectively;
- the operation $\cdot$ distributes over $+$ from both sides; and
- 0 is absorbing (i.e., $0 \cdot \kappa = 0 = \kappa \cdot 0$ for every $\kappa \in K$).

Important commutative semirings are for example:

- the BOOLEAN semiring $(\mathbb{B}, \vee, \wedge, 0, 1)$;
- the arctic semiring $(\mathbb{R}^{-\infty}, \max, +, -\infty, 0)$; and
- the semiring $(\mathbb{N}, +, \cdot, 0, 1)$ of natural numbers.

The semiring K is *additively cancellative* if $\kappa + \kappa_1 = \kappa + \kappa_2$ implies $\kappa_1 = \kappa_2$ for every $\kappa, \kappa_1, \kappa_2 \in K$. Note that if Q is a set and K a commutative semiring, then also K^Q with component-wise addition and multiplication is a commutative semiring (we will use $\odot$ to denote componentwise multiplication). With the usual matrix addition $+$ and multiplication $\cdot$, the structure $(K^{Q \times Q}, +, \cdot)$ is a noncommutative semiring (i.e., multiplication is not commutative). For a more thorough introduction to semirings we refer the reader to [17].

From now on, we will drop the prefix commutative *when we talk about semirings. We assume that the used semirings are always commutative. This allows us to freely reorder factors except in matrix multiplications.*

Formal power series. Let K be a semiring and Z a set. A *formal power series* over K and Z is a mapping $\mathcal{S} : Z \to K$. The set of all power series over K and Z is denoted by $K\langle\langle Z \rangle\rangle$. As convention bids, for every $z \in Z$ we usually write the value $\mathcal{S}(z)$ as $(\mathcal{S}, z)$.

Weighted string automata. A *weighted string automaton* (wsa) [30, 13] is a tuple $A = (Q, \Sigma, K, \lambda, \mu, \nu)$ where

- Q is an alphabet (of *states*);
- Σ is an alphabet (of *input symbols*);
- K is a semiring;
- $\mu : \Sigma \to K^{Q \times Q}$ assigns a *transition weight matrix* to each symbol; and
- $\lambda, \nu \in K^Q$ are an *initial* and a *final weight vector*, respectively, where λ is considered to be a $(1 \times Q)$ matrix and ν a $(Q \times 1)$ matrix.

The *size* of the wsa A, denoted by $size(A)$, is $n + m$, where n is the number of states and m is the total number of nonzero entries in the transitions weight matrices.

The mapping $\mu : \Sigma \to K^{Q \times Q}$ uniquely extends to a monoid homomorphism $\overline{\mu}$ from $(\Sigma^*, \cdot)$ to $(K^{Q \times Q}, \cdot)$. The formal power series $\|A\| \in K\langle\langle \Sigma^* \rangle\rangle$ *recognised*

by A is defined for every $w \in \Sigma^*$ by $(\|A\|, w) = \lambda \cdot \overline{\mu}(w) \cdot \nu$ (where λ and ν are seen as row and column vector, respectively). We denote by $Rec(K, \Sigma)$ the set of all power series of $K\langle\langle\Sigma^*\rangle\rangle$ that are recognisable by some wsa.

For later convenience, for every $q \in Q$ we define the wsa $A^q = (Q, \Sigma, K, \lambda, \mu, \nu')$ where $\nu'(q) = \nu(q)$ and $\nu'(p) = 0$ for every $p \in Q \setminus \{q\}$. The wsa A is *deterministic* if (i) there exists at most one $q \in Q$ such that $\lambda(q) \neq 0$; and (ii) for every $f \in \Sigma$ and $p \in Q$ there exists at most one $q \in Q$ such that $\mu(f)_{p,q} \neq 0$.

Next, we recall certain constructions that modify wsa. We start with a slightly non-standard construction, which will prove useful in Section 5. Let $\mathcal{P}$ be an equivalence relation on Σ. We define the wsa $A^{\mathcal{P}} = (Q, \Sigma/\mathcal{P}, K, \lambda, \mu', \nu)$ where $\mu'(B)_{p,q} = \sum_{f \in B} \mu(f)_{p,q}$ for every $B \in \Sigma/\mathcal{P}$ and $p, q \in Q$. A straightforward proof shows that $\overline{\mu}'(B_1 \cdots B_k)$ is equal to $\sum_{f_1 \in B_1, \ldots, f_k \in B_k} \overline{\mu}(f_1 \cdots f_k)$ and that $(\|A^{\mathcal{P}}\|, B_1 \cdots B_k)$ is equal to $\sum_{f_1 \in B_1, \ldots, f_k \in B_k} (\|A\|, f_1 \cdots f_k)$ for every $B_1, \ldots, B_k \in \Sigma/\mathcal{P}$.

The following recalls notions from [7]. Let $\mathcal{R}$ be an equivalence relation on Q. We say that $\mathcal{R}$ is a *forward bisimulation on* A if for every $(p_1, p_2) \in \mathcal{R}$

(i) $\nu(p_1) = \nu(p_2)$ and
(ii) $\sum_{q \in B} \mu(f)_{p_1,q} = \sum_{q \in B} \mu(f)_{p_2,q}$ for every $f \in \Sigma$ and $B \in Q/\mathcal{R}$.

The *forward-aggregated wsa*, denoted by $A/\mathcal{R}$, is the wsa $(Q/\mathcal{R}, \Sigma, K, \lambda', \mu', \nu')$ with

- $\lambda'(B) = \sum_{q \in B} \lambda(q)$ for every $B \in Q/\mathcal{R}$;
- $\mu'(f)_{B,[q]} = \sum_{p \in B} \mu(f)_{p,q}$ for every $f \in \Sigma$, $q \in Q$, and $B \in Q/\mathcal{R}$; and
- $\nu'([q]) = \nu(q)$ for every $q \in Q$.

The *reversed wsa* $\mathrm{Rev}(A)$ is $(Q, \Sigma, K, \nu, \mu', \lambda)$ with $\mu'(f) = \mu(f)^{\mathrm{T}}$ [i.e., the transpose of $\mu(f)$] for every $f \in \Sigma$. With this at hand, $\mathcal{R}$ is a *backward bisimulation on* A if $\mathcal{R}$ is a forward bisimulation on $\mathrm{Rev}(A)$. The *backward-aggregated wsa* is $\mathrm{Rev}(\mathrm{Rev}(A)/\mathcal{R})$. We will denote it by $A/\mathcal{R}$ provided that it is understood that $\mathcal{R}$ is a backward bisimulation on A.

Finally, we consider a slight extension of wsa. Let P be a set. A *wsa with P-output* is a tuple $A = (Q, \Sigma, K, \lambda, \mu, \nu)$ where Q, Σ, K, λ, and μ are as in a wsa and $\nu \in K^{Q \times P}$. The semantics $\|A\|$ is defined as for wsa with the difference that now $\|A\| \in K^P\langle\langle\Sigma^*\rangle\rangle$. For consistency, we will try to avoid to handle wsa with P-output directly. Instead, we commonly assume that $P \cap Q = \emptyset$ and simulate A by the wsa $sim(A) = (Q \cup P, \Sigma \cup \{\Diamond\}, K, \lambda', \mu', \nu')$ where

- $\Diamond \notin \Sigma$ is a new special symbol that we reserve for exactly this use;
- $\lambda'(q) = \lambda(q)$ for every $q \in Q$;
- $\mu'(f)_{q,q'} = \mu(f)_{q,q'}$ for every $f \in \Sigma$ and $q, q' \in Q$;
- $\mu'(\Diamond)_{q,p} = \nu_{q,p}$ for every $q \in Q$ and $p \in P$; and
- $\nu'(p) = 1$ for every $p \in P$.
- All remaining entries in λ', μ', and ν' are 0.

The following informal description of $sim(A)$ shall illustrate its definition. We present its matrices λ', μ', and ν' by blocks where we assume that the states

of Q have smaller indices than those of P.

$$\lambda' = \begin{pmatrix} \lambda \\ 0 \end{pmatrix} \quad \forall f \in \Sigma\colon \mu'(f) = \begin{pmatrix} \mu(f) & 0 \\ 0 & 0 \end{pmatrix} \quad \mu'(\Diamond) = \begin{pmatrix} 0 & \nu \\ 0 & 0 \end{pmatrix} \quad \nu' = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

It is easily seen that $(\|sim(A)^p\|, w\Diamond) = (\|A\|, w)_p$ for every $p \in P$ and $w \in \Sigma^*$. A wsa with P-output is *deterministic* if $sim(A)$ is deterministic. Moreover, the *size* of A, denoted $size(A)$, is simply the size of $sim(A)$.

Unranked trees and ranked trees. Let Σ be an alphabet. The set U_Σ of *(unranked) trees over* Σ is the smallest subset of $(\Sigma \cup \{[,]\} \cup \{,\})^*$ such that for every $f \in \Sigma$, $k \in \mathbb{N}$, and $t_1, \ldots, t_k \in U_\Sigma$ also $f[t_1, \ldots, t_k] \in U_\Sigma$. To improve readability, we henceforth identify $f[]$ with f. The set of *positions* in the tree t, denoted by $Pos(t)$, is inductively defined for every $t = f[t_1, \ldots, t_k] \in U_\Sigma$ by $Pos(t) = \{\varepsilon\} \cup \{iw \mid 1 \leq i \leq k \text{ and } w \in Pos(t_i)\}$.

Let $t = f[t_1, \ldots, t_k] \in U_\Sigma$, $u \in U_\Sigma$, and $w \in Pos(t)$. The *height* of t, denoted by $height(t)$, is 0 if $k = 0$, and $1 + \max\{height(t_i) \mid i \in [k]\}$ otherwise. The *rank of t at w*, the *label of t at w*, the *subtree of t at w*, and the *replacement of the subtree at w in t by u* are denoted by $rank_t(w)$, $t(w)$, $t|_w$, and $t[u]_w$, respectively. They are defined as follows: $rank_t(\varepsilon) = k$, $t(\varepsilon) = f$, $t|_\varepsilon = t$, and $t[u]_\varepsilon = u$; and $rank_t(iv) = rank_{t_i}(v)$, $t(iv) = t_i(v)$, $t|_{iv} = t_i|_v$, and $t[u]_{iv} = f[t_1, \ldots, t_{i-1}, t_i[u]_v, t_{i+1}, \ldots, t_k]$ for every $i \in [k]$ and $v \in Pos(t_i)$. Now let $\Delta \subseteq \Sigma$. We define the set $Pos_\Delta(t) = \{w \in Pos(t) \mid t(w) \in \Delta\}$ for every $t \in U_\Sigma$.

A *ranked alphabet* is an alphabet Σ together with a mapping $rk : \Sigma \to \mathbb{N}$. The set T_Σ of *ranked trees* over Σ is the subset of U_Σ that is given by

$$T_\Sigma = \{t \in U_\Sigma \mid \forall w \in Pos(t) : rank_t(w) = rk(t(w))\} \ .$$

Since ranked trees are particular unranked trees, the notions of position, subtrees, label, and replacement are also defined for ranked trees. For every $k \geq 0$, we let $\Sigma_{(k)} = rk^{-1}(k)$ and $max_\Sigma = \max\{k \mid \Sigma_{(k)} \neq \emptyset\}$. Let Σ be a ranked alphabet and Q a finite set disjoint with Σ. We define the ranked alphabet Δ by $\Delta_{(0)} = \Sigma_{(0)} \cup Q$ and $\Delta_{(k)} = \Sigma_{(k)}$ for every $k \geq 1$. We henceforth use $T_\Sigma(Q)$ as an alias for T_Δ.

A power series $\mathcal{S} \in K\langle\langle Z \rangle\rangle$ such that $Z \subseteq U_\Sigma$ is called *(unranked) tree series*. When $Z \subseteq T_\Sigma$ we also say that $\mathcal{S}$ is a *ranked tree series*.

3 Weighted tree automata

In this section we briefly recall the main definitions and required results on weighted (ranked) tree automata. For more details on weighted tree automata we refer to [4, 3, 14, 11, 10]. Let us start with the syntax.

Definition 3.1 A *weighted tree automaton* (wta) is a tuple $M = (Q, \Sigma, K, \mu, \omega)$ where Q is an alphabet (of *states*), Σ is a ranked alphabet (of *input symbols*),

K is a semiring, μ is an indexed family $(\mu_k)_{k\in\mathbb{N}}$ of mappings $\mu_k : \Sigma_{(k)} \to K^{Q^k \times Q}$, and $\omega \in K^Q$ is a vector (of *root weights*). The wta M is *deterministic* if for every symbol $f \in \Sigma_{(k)}$ and word $w \in Q^k$ there exists at most one state $q \in Q$ such that $\mu_k(f)_{w,q} \neq 0$. □

For the remainder of this section, let $M = (Q, \Sigma, K, \mu, \omega)$ be a wta. To simplify the following discussion, we assume, without loss of generality, that Σ is disjoint to Q. Let us proceed with the semantics. We will use the definition of the run semantics from [11] because we need the added flexibility in the next section.

Definition 3.2 Let $t \in T_\Sigma(Q)$ be an input tree. A *run of M on t* is a mapping $r\colon Pos(t) \to Q$ such that $r(w) = t(w)$ for every $w \in Pos_Q(t)$. The set of all runs of M on t is denoted by $run_M(t)$. The *weight* of r is

$$weight_M(r) = \prod_{\substack{w \in Pos_\Sigma(t) \\ k = rk(t(w))}} \mu_k(t(w))_{r(w1)\cdots r(wk), r(w)} \quad .$$

The *ranked tree series* recognised by M is $\|M\|$ such that

$$(\|M\|, t) = \sum_{r \in run_M(t)} weight_M(r) \cdot \omega(r(\varepsilon))$$

for every $t \in T_\Sigma$. Any ranked tree series that can be recognised by a wta is said to be *recognisable*. □

It should be noted that we use final weights whereas [11] uses only final states (see [5] for a discussion). The following decomposition result for runs can be obtained by instantiating [11, Observation 4.6]; note that the final weights do not influence the result.

Lemma 3.3 Let $t \in T_\Sigma$ and $w_1, \ldots, w_n \in Pos(t)$ be such that $w_i \neq w_j w$ for all $i, j \in [n]$ with $i \neq j$ and $w \in \mathbb{N}^*$ (i.e., $w_1, \ldots, w_n$ are pairwise incomparable with respect to the prefix order on $\mathbb{N}^*$). Then for every $q \in Q$

$$\sum_{\substack{r \in run_M(t) \\ r(\varepsilon) = q}} weight_M(r) = \sum_{\substack{r' \in run_M(t_w), r'(\varepsilon) = q \\ \forall i \in [n] : r_i \in run_M(t|_{w_i}) \\ w = r_1(\varepsilon)\cdots r_n(\varepsilon)}} weight_M(r') \cdot \prod_{i=1}^{n} weight_M(r_i)$$

where $t_w = (\cdots((t[w(1)]_{w_1})[w(2)]_{w_2})\cdots)[w(n)]_{w_n}$. □

In Section 4, the computation of forward and backward bisimulations for wsuta is reduced to reinterpreting the input wsuta M as a wta M' running on a binary encoding of the unranked input trees, and then computing the corresponding bisimulations on M' using the results of [21]. For the sake of this reduction we now restate the definitions of backward and forward bisimulation for wta

and the definitions of the corresponding aggregated wta [21]. Note that these details are only needed to verify some statements, so that on first reading these definitions can safely be skipped.

Definition 3.4 An equivalence relation $\mathcal{R}$ on Q is a *backward bisimulation on* M if for every $(q_1, q_2) \in \mathcal{R}$, $f \in \Sigma_{(k)}$, and $B_1, \ldots, B_k \in Q/\mathcal{R}$

$$\sum_{p_1 \in B_1, \ldots, p_k \in B_k} \mu_k(f)_{p_1 \cdots p_k, q_1} = \sum_{p_1 \in B_1, \ldots, p_k \in B_k} \mu_k(f)_{p_1 \cdots p_k, q_2} \quad .$$

The *backward-aggregated wta* is the wta $M/\mathcal{R} = (Q/\mathcal{R}, \Sigma, K, \mu', \omega')$ with

- $\omega'(B) = \sum_{q \in B} \omega(q)$ for every $B \in Q/\mathcal{R}$; and
- for every $f \in \Sigma_{(k)}$, $B_1, \ldots, B_k \in Q/\mathcal{R}$, and $q \in Q$

$$\mu'_k(f)_{B_1 \cdots B_k, [q]} = \sum_{p_1 \in B_1, \ldots, p_k \in B_k} \mu_k(f)_{p_1 \cdots p_k, q} \quad . \qquad \square$$

Let $\square \notin Q$ and $k \geq 0$. The set of *contexts* is $C^Q_{(k)} = \bigcup_{k=i+j+1} Q^i \times \{\square\} \times Q^j$. 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.

Definition 3.5 An equivalence relation $\mathcal{R}$ on Q is a *forward bisimulation on* M if for every $(q_1, q_2) \in \mathcal{R}$

(i) $\omega(q_1) = \omega(q_2)$ and
(ii) for every $f \in \Sigma_{(k)}$, $B \in Q/\mathcal{R}$, and $c \in C^Q_{(k)}$

$$\sum_{p \in B} \mu_k(f)_{c[\![q_1]\!], p} = \sum_{p \in B} \mu_k(f)_{c[\![q_2]\!], p} \quad .$$

The *forward-aggregated wta* is the wta $M/\mathcal{R} = (Q/\mathcal{R}, \Sigma, K, \mu', \omega')$ with

- $\omega'([q]) = \omega(q)$ for every $q \in Q$; and
- for every $f \in \Sigma_{(k)}$, $q_1, \ldots, q_k \in Q$, and $B \in Q/\mathcal{R}$

$$\mu'_k(f)_{[q_1] \cdots [q_k], B} = \sum_{p \in B} \mu_k(f)_{q_1 \cdots q_k, p} \quad . \qquad \square$$

4 Weighted stepwise unranked tree automata

In this section, we introduce one of our main models: a generalisation of the stepwise unranked tree automaton (suta) of [26, 8]. The main difference between the suta and other models of unranked tree automata is that the fsa generating the recognisable languages of states share one transition graph and additionally work with the same states as the overall suta.

Definition 4.1 (cf. [26, Sect. 4.2]) A *weighted stepwise unranked tree automaton* (wsuta) is a tuple $M = (Q, \Sigma, K, (\lambda_f)_{f\in\Sigma}, \mu, \nu, \omega)$, where

- Q is an alphabet (of *states*);
- Σ is an alphabet (of *input symbols*);
- K is a semiring;
- $\lambda_f \in K^Q$ is an *initial weight vector* for every $f \in \Sigma$;
- $\mu : Q \to K^{Q\times Q}$ assigns a matrix (of *transitions weights*) to each state;
- $\nu \in K^Q$ is a *final weight vector*; and
- $\omega \in K^Q$ is a *root weight vector*.

The wsuta M is *deterministic* if (i) for every $f \in \Sigma$ there exists at most one $q \in Q$ such that $\lambda_f(q) \neq 0$; and (ii) the wsa $(Q, Q, K, \lambda_f, \mu, \nu)$ is deterministic for every $f \in \Sigma$. □

In the unweighted case (i.e., the semiring K is the BOOLEAN semiring), the model coincides exactly with the model of [26] apart from the final weight vector, which we have added. However, we will show that this slight change does not increase the computational power.

Henceforth, let $M = (Q, \Sigma, K, (\lambda_f)_{f\in\Sigma}, \mu, \nu, \omega)$ be a wsuta. For every $f \in \Sigma$ we denote by M_f the wsa $(Q, Q, K, \lambda_f, \mu, \nu)$. This identifies the wsa that is responsible for the input symbol f. As evidenced by the definition, only the initial state vector depends on the symbol f. Let us proceed with the definition of the semantics of wsuta.

Definition 4.2 Let $t \in U_\Sigma$. A *run of M on t* is a mapping $r\colon Pos(t) \to Q$. The set of all runs of M on t is denoted by $run_M(t)$. The *weight* of r is

$$weight_M(r) = \prod_{\substack{w\in Pos(t)\\ k=rank_t(w)}} (\|M^{r(w)}_{t(w)}\|, r(w1)\cdots r(wk)) \ .$$

Note that M_f^q stands for $(M_f)^q$. The *unranked tree series* recognised by M, denoted by $\|M\|$, is defined for every $t \in U_\Sigma$ by

$$(\|M\|, t) = \sum_{r\in run_M(t)} weight_M(r) \cdot \omega(r(\varepsilon)) \ .$$

Any unranked tree series that is recognised by a wsuta is *recognisable*. □

Two wsuta are *equivalent* if they recognise the same unranked tree series. It is not difficult to show that every recognisable ranked tree series is also unranked recognisable, using the fact that, for every $k \in \mathbb{N}$, every power series of $K\langle\langle Q^k\rangle\rangle$ is recognisable.

Example 4.3 Let $\Sigma = \{a, b\}$ and $t \in U_\Sigma$. A sequence of positions in t is *horizontally-connected* if it is equal to ε, or can be written in the form $wi, \ldots, wj$ for some position $w \in Pos(t)$ and indices $i, j \in [rank_t(w)]$, such that $i \leq j$.

Consider now the unranked tree series $\mathcal{S} \in \mathbb{R}\langle\langle U_\Sigma \rangle\rangle$ such that $(\mathcal{S}, t)$ is the length of the longest horizontally-connected sequence of a-labelled positions in t. To recognise $\mathcal{S}$, we introduce the wsuta $M = (Q, \Sigma, \mathrm{Arct}, (\lambda_f)_{f\in\Sigma}, \mu, \nu, \omega)$ where $Q = \{q_a, q_x, q, q_\perp\}$, $\mathrm{Arct} = (\mathbb{R}^{-\infty}, \max, +, -\infty, 0)$, the mappings ν and ω take every state to 0, and λ_a, λ_b, and μ are given in Figure 1. All transitions with weight $-\infty$ have been omitted. The wsuta M computes the coefficient of a tree t in $\mathcal{S}$ as follows. In each run, M nondeterministically selects a number of positions labelled a and counts them. It then verifies that these positions form a horizontally connected sequence by entering into the states q and $q_\perp$. The transitions of M are such that it cannot count the length of two distinct sequences in a single run. The weight of each run of M on t is thus the length of a horizontally connected sequence of a-labelled positions in t, and the summation over all runs (i.e., the maximum in Arct) yields the maximal such length.

Let us demonstrate M on the tree $t = b[aa[aab]a]$. Figure 2 shows two runs of M on t that have non-zero weight. Here, the state assigned to a position $w \in Pos(t)$ is one possible outcome of running the wsa $M_{t(w)}$ on the word $t(w1)\cdots t(wk)$, where $k = rank_t(w)$. If we examine the left run r_1 closer, then we see that the states $r_1(21) = q_a$, $r_1(22) = q_x$, and $r_1(23) = q_x$ can be obtained by executing M_a, M_a, and M_b, respectively, all on the empty word in Q^*, guessing the initial states q_a, q_x, and q_x, respectively. The states $r_1(1) = q_x$, $r_1(2) = q_\perp$, and $r_1(3) = q_x$ can be derived by running M_a on ε, on $q_a q_x q_x$, and on ε, respectively, each time guessing the initial state q_x. The state $r_1(\varepsilon) = q_\perp$ is the result of running M_b on the word $q_x q_\perp q_x$ again guessing $q_\perp$ as initial state. An easy computation yields that $weight_M(r_1) = 1$. Had M instead guessed q_a as initial state when computing $r_1(22)$, then the local optimum of 2 would have been found. The global optimum is however identified by run r_2 with weight $weight_M(r_2) = 3$, viz. the length of the longest horizontally-connected sequence of a-labelled positions in t. □

The final mapping of the wsuta discussed in Example 4.3 takes every element to 0 (the multiplicative identity of the arctic semiring). As we shall see, this is not a restriction, but something that can always be achieved. The wsuta $M = (Q, \Sigma, K, (\lambda_f)_{f\in\Sigma}, \mu, \nu, \omega)$ is in *final weight normal form* if $\nu = \vec{1}$ (i.e., $\nu(q) = 1$ for every $q \in Q$).

Lemma 4.4 There exists a wsuta M' in final weight normal form that is equivalent to M. If M is deterministic, then so is M'.

Proof Intuitively speaking, we move the final weights from ν to the transitions of μ and to the root weights ω. Formally, we construct the wsuta $M' = (Q, \Sigma, K, (\lambda_f)_{f\in\Sigma}, \mu', \vec{1}, \omega')$ where $\mu'(q) = \mu(q)\cdot\nu(q)$ and $\omega'(q) = \omega(q)\cdot\nu(q)$ for every $q \in Q$. Note that M' is deterministic if M is so. It remains to prove that M and M' are equivalent. To this end, we first observe that

$$\overline{\mu}'(w) = \prod_{i=1}^{|w|} \mu'(w(i)) = \prod_{i=1}^{|w|} \Big(\mu(w(i)) \cdot \nu(w(i))\Big) = \overline{\mu}(w) \cdot \prod_{i=1}^{|w|} \nu(w(i))$$

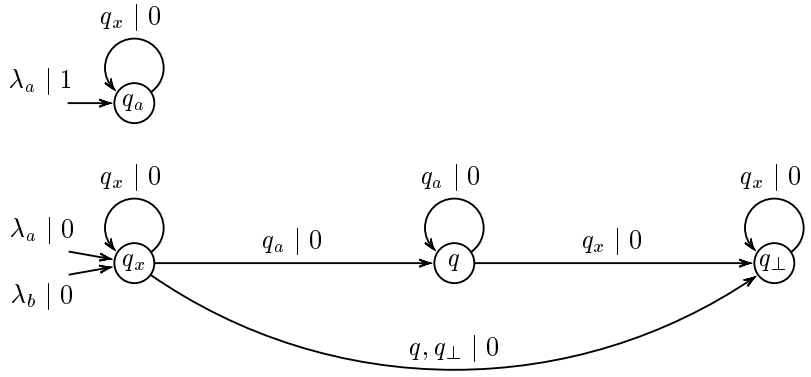


Figure 1: The wsuta M maps each $t \in U_{\{a,b\}}$ to the length of of the longest horizontally-connected sequence of a-labelled positions in t.

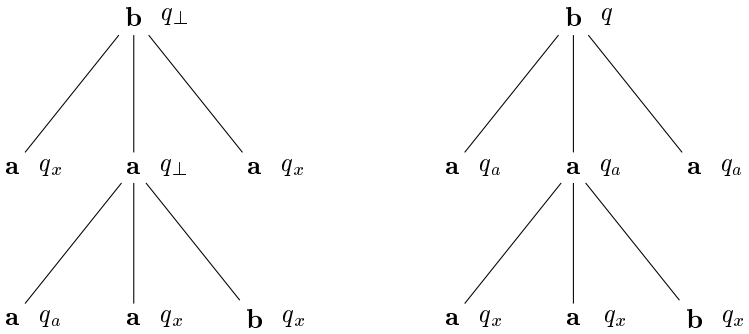


Figure 2: Two runs of the wsuta M in Example 4.3 on the input tree $b[aa[aab]a]$. The run on the left has weight 1, whereas the run on the right has weight 3.

for every $w \in Q^*$. From this we can easily conclude

$$(\|(M')_f^q\|, w) = \big(\lambda_f \cdot \overline{\mu}(w)\big)_q \cdot \prod_{i=1}^{|w|} \nu(w(i)) \tag{1}$$

for every $f \in \Sigma$, $q \in Q$, and $w \in Q^*$. Now, let $t \in U_\Sigma$ and $r \in run_M(t)$. Clearly, $run_M(t) = run_{M'}(t)$. Then

$$weight_{M'}(r) \cdot \nu(r(\varepsilon)) = \prod_{\substack{v \in Pos(t) \\ n = rank_t(v) \\ w = r(v1)\cdots r(vn)}} (\|(M')_{t(v)}^{r(v)}\|, w) \cdot \nu(r(\varepsilon))$$

$$= \{\text{by (1)}\}$$

$$\prod_{\substack{v\in Pos(t)\\ n=rank_t(v)\\ w=r(v1)\cdots r(vn)}} \big(\lambda_{t(v)}\cdot\overline{\mu}(w)\big)_{r(v)}\cdot\prod_{i=1}^{n}\nu(w(i))\cdot\nu(r(\varepsilon))$$

$$= \prod_{\substack{v\in Pos(t)\\ n=rank_t(v)\\ w=r(v1)\cdots r(vn)}} \big(\lambda_{t(v)}\cdot\overline{\mu}(w)\cdot\nu\big)_{r(v)} = \prod_{\substack{v\in Pos(t)\\ n=rank_t(v)\\ w=r(v1)\cdots r(vn)}} (\|M^{r(v)}_{t(v)}\|, w)$$

$$= weight_M(r) \ .$$

This allows us to complete the proof for every $t \in U_\Sigma$ as follows:

$$\begin{aligned}(\|M'\|, t) &= \sum_{r\in run_{M'}(t)} weight_{M'}(r)\cdot\omega(r(\varepsilon))\cdot\nu(r(\varepsilon))\\ &= \sum_{r\in run_M(t)} weight_M(r)\cdot\omega(r(\varepsilon)) = (\|M\|, t) \ .\end{aligned}$$ ∎

In the sequel, we assume that M is in final weight normal form. Consequently, we drop the final weight component and just write $(Q, \Sigma, K, (\lambda_f)_{f\in\Sigma}, \mu, \omega)$. We now present definitions of backward and forward bisimulation for M, but later we will show that these are the notions known for wta [21].

Definition 4.5 (cf. Def. 3.4) An equivalence relation $\mathcal{R}$ on Q is a *backward bisimulation on* M if for every $(q_1, q_2) \in \mathcal{R}$

- $\lambda_f(q_1) = \lambda_f(q_2)$ for every $f \in \Sigma$; and
- for every $B_1, B_2 \in Q/\mathcal{R}$

$$\sum_{p_1\in B_1, p_2\in B_2} \mu(p_1)_{p_2,q_1} = \sum_{p_1\in B_1, p_2\in B_2} \mu(p_1)_{p_2,q_2} \ .$$ □

Definition 4.6 (cf. Def. 3.5) An equivalence relation $\mathcal{R}$ on Q is a *forward bisimulation on* M if for every $(q_1, q_2) \in \mathcal{R}$

- $\omega(q_1) = \omega(q_2)$,
- $\sum_{p\in B} \mu(q)_{q_1,p} = \sum_{p\in B} \mu(q)_{q_2,p}$ for every $q \in Q$ and $B \in Q/\mathcal{R}$; and
- $\sum_{p\in B} \mu(q_1)_{q,p} = \sum_{p\in B} \mu(q_2)_{q,p}$ for every $q \in Q$ and $B \in Q/\mathcal{R}$. □

Our objective is now to provide minimisation algorithms based on Definitions 4.5 and 4.6. It was already remarked in [26] that suta can be seen as fta over a special encoding of the input trees. Since this correspondence will yield all of the desired results, let us make this precise here. Let $\sigma \notin \Sigma$ be a special symbol of rank 2. We define the binary encoding $c : U_\Sigma \to T_{\Sigma\cup\{\sigma\}}$ as follows. Let $u = f[u_1, \ldots, u_n]$ for some $f \in \Sigma$ and $u_1, \ldots, u_n \in U_\Sigma$. We define $c(u) = \sigma[\sigma[\cdots\sigma[\sigma[f, c(u_1)], c(u_2)]\cdots, c(u_{n-1})], c(u_n)]$. Note that $c(f) = f$ and that c is a bijection. We extend c to c: $K\langle\langle U_\Sigma\rangle\rangle \to K\langle\langle T_{\Sigma\cup\{\sigma\}}\rangle\rangle$ by $(c(\mathcal{S}), t) = (\mathcal{S}, c^{-1}(t))$, for every $\mathcal{S} \in K\langle\langle U_\Sigma\rangle\rangle$ and $t \in T_{\Sigma\cup\{\sigma\}}$. It is easy to see that also c: $K\langle\langle U_\Sigma\rangle\rangle \to K\langle\langle T_{\Sigma\cup\{\sigma\}}\rangle\rangle$ is a bijection.

Definition 4.7 The *wta corresponding to* M is $c(M) = (Q, \Sigma \cup \{\sigma\}, K, \mu', \omega)$ where

- every symbol of Σ has rank 0 and σ has rank 2;
- $\mu'_0(f)_{\varepsilon,q} = \lambda_f(q)$ for every $f \in \Sigma$ and $q \in Q$; and
- $\mu'_2(\sigma)_{q_1q_2,q} = \mu(q_2)_{q_1,q}$ for every $q, q_1, q_2 \in Q$. □

Clearly, we can reconstruct M from $c(M)$. More generally, given an arbitrary wta M' over the ranked alphabet $\Sigma \cup \{\sigma\}$ we can construct a wsuta M'' over Σ such that $c(M'') = M'$. So, we established a one-to-one correspondence; in fact, $c(M)$ is essentially just a reinterpretation of M. Note that M is deterministic if and only if $c(M)$ is deterministic. Next we show the semantic relation between M and $c(M)$.

Lemma 4.8 $c(\|M\|) = \|c(M)\|$.

Proof Let $c(M) = (Q, \Sigma \cup \{\sigma\}, K, \mu', \omega)$ and

$$t = \sigma[\sigma[\cdots\sigma[\sigma[f, t_1], t_2]\cdots, t_{n-1}], t_n]$$

for some $f \in \Sigma$ and $t_1, \ldots, t_n \in T_{\Sigma\cup\{\sigma\}}$. Note that for $n = 0$, we obtain $t = f$. Let $w \in Q^*$ with $|w| = n$, and let $t_w = \sigma[\cdots\sigma[\sigma[f, w(1)], w(2)]\cdots, w(n)]$. First, we calculate for every $q \in Q$ and $f \in \Sigma$ as follows:

$$(\|M_f^q\|, w) = \left(\lambda_f \cdot \overline{\mu}(w)\right)_q = \sum_{\substack{p_0,\ldots,p_n \in Q \\ p_n = q}} \lambda_f(p_0) \cdot \prod_{i=1}^{n} \mu(w(i))_{p_{i-1},p_i}$$

$$= \{\text{set } r(1^i) = p_{n-i} \text{ for every } 1 \le i \le n \text{ and vice versa}\}$$

$$\sum_{\substack{r \in run_{c(M)}(t_w) \\ r(\varepsilon) = q}} \mu'_0(f)_{\varepsilon, r(1^n)} \cdot \prod_{i=n-1}^{0} \mu'_2(\sigma)_{r(1^{i+1})w(n-i), r(1^i)}$$

$$= \sum_{\substack{r \in run_{c(M)}(t_w) \\ r(\varepsilon) = q}} \left(\mu'_0(t_w(1^n))_{\varepsilon, r(1^n)} \cdot \prod_{i=0}^{n-1} \mu'_2(t_w(1^i))_{r(1^i1)r(1^i2), r(1^i)}\right)$$

$$= \{\text{note that } \{1^i \mid 0 \le i \le n\} = Pos_\Sigma(t_w)\}$$

$$\sum_{\substack{r \in run_{c(M)}(t_w) \\ r(\varepsilon) = q}} weight_{c(M)}(r) \quad . \tag{2}$$

We can now show by induction on the height of t that for every $q \in Q$,

$$\sum_{\substack{r \in run_M(c^{-1}(t)) \\ r(\varepsilon) = q}} weight_M(r) = \sum_{\substack{r \in run_{c(M)}(t) \\ r(\varepsilon) = q}} weight_{c(M)}(r) \quad . \tag{3}$$

We compute as follows

$$\sum_{\substack{r\in run_M(c^{-1}(t))\\ r(\varepsilon)=q}} weight_M(r) = \sum_{\substack{r\in run_M(f[c^{-1}(t_1),\ldots,c^{-1}(t_n)])\\ r(\varepsilon)=q}} weight_M(r)$$

$$= \sum_{\substack{\forall i\in[n]:\ r_i\in run_M(c^{-1}(t_i))\\ w=r_1(\varepsilon)\cdots r_n(\varepsilon)}} (\|M_f^q\|, w)\cdot \prod_{i=1}^{n} weight_M(r_i)$$

$$= \{\text{by distributivity}\}$$

$$\sum_{w\in Q^n} (\|M_f^q\|, w)\cdot \prod_{i=1}^{n}\Big(\sum_{\substack{r_i\in run_M(c^{-1}(t_i))\\ r_i(\varepsilon)=w(i)}} weight_M(r_i)\Big)$$

$$= \{\text{by (2) and the induction hypothesis}\}$$

$$\sum_{\substack{w\in Q^n\\ r'\in run_{c(M)}(t_w)\\ r'(\varepsilon)=q}} weight_{c(M)}(r')\cdot \prod_{i=1}^{n}\Big(\sum_{\substack{r_i\in run_{c(M)}(t_i)\\ r_i(\varepsilon)=w(i)}} weight_{c(M)}(r_i)\Big)$$

$$= \{\text{by distributivity}\}$$

$$\sum_{\substack{r'\in run_{c(M)}(t_w), r'(\varepsilon)=q\\ \forall i\in[n]:\ r_i\in run_{c(M)}(t_i)\\ w=r_1(\varepsilon)\cdots r_n(\varepsilon)}} weight_{c(M)}(r')\cdot \prod_{i=1}^{n} weight_{c(M)}(r_i)$$

$$= \{\text{by Lemma 3.3}\}$$

$$\sum_{\substack{r\in run_{c(M)}(t)\\ r(\varepsilon)=q}} weight_{c(M)}(r) \quad .$$

Finally, we can complete the proof with the following calculation:

$$(\|M\|, c^{-1}(t)) = \sum_{q\in Q}\Big(\sum_{\substack{r\in run_M(c^{-1}(t))\\ r(\varepsilon)=q}} weight_M(r)\Big)\cdot\omega(q)$$

$$= \{\text{by (3)}\}$$

$$\sum_{q\in Q}\Big(\sum_{\substack{r\in run_{c(M)}(t)\\ r(\varepsilon)=q}} weight_{c(M)}(r)\Big)\cdot\omega(q) = (\|c(M)\|, t) \qquad \blacksquare$$

This close relation can now be used to transfer a multitude of results (e.g., on determinisation, minimisation, bisimulation minimisation) from the ranked case to the unranked case. Here, we are only interested in bisimulation minimisation, which allows us to reduce the number of states of a (potentially nondeterministic) wsuta. Fortunately, every backward (resp., forward) bisimulation on M is

also a backward (resp., forward) bisimulation on $c(M)$ and vice versa. This allows us to use the existing algorithms [21, 20, 1] in order to minimise wsuta. We say that M is *backward* (resp., *forward*) *bisimulation minimal* if it admits only the identity as backward (resp., forward) bisimulation. For the reader's convenience, we show the adaptation of the main theorems of [21, 20] to the case of wsuta (the results of [1] are superseded by the others). Let n be the number of states of M (i.e., $n = |Q|$) and m be the number of nonzero entries in μ.

Theorem 4.9 (see [21, Theorems 11–29] and [20, Theorems 12–27]) For every wsuta M an equivalent backward (resp., forward) bisimulation minimal wsuta can be computed in time $O(mn)$. If the used semiring is additively cancellative or the BOOLEAN semiring, then we can achieve it in time $O(m \log n)$.

Proof We first construct $c(M)$. This is a purely representational issue and thus can be implemented in constant time. By [21, Theorems 11,13,25,27], backward (resp., forward) bisimulation minimisation on $c(M)$ run in time $O(mn)$ and return an equivalent backward (resp., forward) bisimulation minimal wta M'. Next, we construct a wsuta M'' such that $c(M'') = M'$ in constant time. By Lemma 4.8, $c(\|M''\|) = \|M'\| = \|c(M)\| = c(\|M\|)$. Since c is a bijection, we obtain $\|M''\| = \|M\|$. It should immediately be clear that M'' is backward (resp., forward) bisimulation minimal because every bisimulation on M'' would also be a bisimulation on M'.

In the special case that the underlying semiring is additively cancellative or the BOOLEAN semiring, the minimisation can be implemented in time $O(m \log n)$ by [21, Theorems 15 & 29] and [20, Theorems 12,15,26,27]. ∎

Finally, let us present a result for minimisation of deterministic suta (i.e., wsuta over the BOOLEAN semiring).

Corollary 4.10 (see [20, Theorems 27–29]) For every deterministic suta, we can compute a minimal equivalent deterministic suta in time $O(m \log n)$.

5 Weighted parallel unranked tree automata

In this section, we consider a second automaton model used on unranked trees. *Weighted parallel unranked tree automata* (wputa) are a generalisation of parallel unranked tree automata (puta) [9]. Although similar to wsuta, wputa differ from wsuta in that their constituent wsa have Q-output and each has its own state space and transition graph. Not surprisingly, wputa and wsuta are equally expressive (see Theorem 6.6). It will show in the course of this section that bisimulation minimisation of wsuta can be readily applied but we need to take care not to merge states from different wsa. Apart from this minor issue, we basically follow the programme of the previous section and start with the formal definition of wputa.

Definition 5.1 A *weighted parallel unranked tree automaton* (wputa) is a tuple $M = (Q, \Sigma, K, (A_f)_{f\in\Sigma}, \omega)$ where

- Q is an alphabet (of *states*);
- Σ is an alphabet (of *input symbols*);
- K is a semiring;
- $A_f = (Q_f, Q, K, \lambda_f, \mu_f, \nu_f)$ is a wsa with Q-output for every $f \in \Sigma$; and
- $\omega \in K^Q$ is a *root weight vector*.

A wputa M is *deterministic* if A_f is deterministic for every $f \in \Sigma$. □

For the rest of this section, let $M = (Q, \Sigma, K, (A_f)_{f\in\Sigma}, \omega)$ be a wputa. The size of M is $size(M) = |Q| + \sum_{f\in\Sigma} size(A_f)$. For the sake of simplicity, we henceforth assume, without loss of generality, that $Q_f \cap Q_{f'} = \emptyset$ and $Q_f \cap Q = \emptyset$, for every $f, f' \in \Sigma$ such that $f \neq f'$. Moreover, suppose that $\Diamond \notin Q$ and $\Diamond \notin Q_f$ for every $f \in \Sigma$. Let us continue with the definition of the semantics.

Definition 5.2 Let $t \in U_\Sigma$. A *run* r of M on t is a mapping $r\colon Pos(t) \to Q$, and we denote by $run_M(t)$ the set of all runs of M on t. Furthermore, the *weight* of r is

$$weight_M(r) = \prod_{\substack{w\in Pos(t)\\ k=rank_t(w)}} (\|A_{t(w)}\|, r(w1)\cdots r(wk))_{r(w)} \ .$$

The *unranked tree series* recognised by M is defined for every $t \in U_\Sigma$ by

$$(\|M\|, t) = \sum_{r\in run_M(t)} weight_M(r) \cdot \omega(r(\varepsilon)) \ . \qquad \square$$

Example 5.3 We return to the problem of recognising the tree series $\mathcal{S}$ of Example 4.3, this time using a wputa. A suitable wputa can be constructed as follows. First we make one copy of the wsuta of Example 4.3 for each input symbol. Then we adjust the initial weight vector in each copy. Finally we remove the unreachable states. In this way, we obtain the wputa $M' = (Q, \Sigma, \mathrm{Arct}, (A'_f)_{f\in\Sigma}, \omega)$ where Q, Σ, Arct, and ω are as in Example 4.3, and the wsa A'_a and A'_b are given in Figure 5. All transitions with weight $-\infty$ have been omitted, and every remaining transition has weight 0, unless otherwise stated. □

We now prepare a semantic relation between wputa and wsuta for the purpose of reducing the problem of bisimulation minimisation from wputa to wsuta. To simplify the construction, we will immediately work with the wsa $sim(A_f)$ that simulates the wsa A_f with Q-output. To this end, let $sim(A_f) = (Q \cup Q_f, Q \cup \{\Diamond\}, K, \lambda_f, \mu_f, \nu_f)$ for every $f \in \Sigma$. Now we simply combine the various $sim(A_f)$ and a special wsa $A_\Diamond$ into a wsuta. This wsuta is essentially the union of the mentioned wsa where $A_\Diamond = (Q \cup Q_\Diamond, Q \cup \{\Diamond\}, K, \lambda_\Diamond, \mu_\Diamond, \nu_\Diamond)$ with (i) $Q_\Diamond = \{\Diamond\}$, (ii) $\lambda_\Diamond(\Diamond) = 1$, (iii) $\mu_\Diamond(\Diamond)_{\Diamond,\Diamond} = 1$, (iv) $\nu_\Diamond(\Diamond) = 1$, and (v) all remaining entries in $\lambda_\Diamond$, $\mu_\Diamond$, and $\nu_\Diamond$ are 0. Henceforth, let $\Sigma' = \Sigma \cup \{\Diamond\}$.

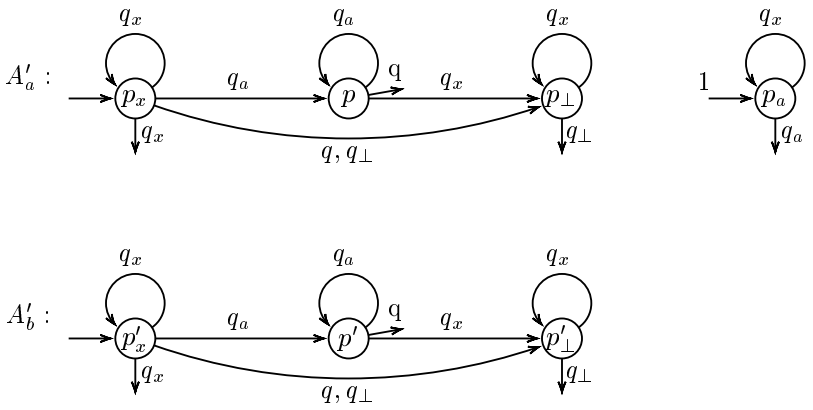


Figure 3: The wputa above is equivalent to the wsuta of Example 4.3.

Definition 5.4 The *wsuta corresponding to* M is

$$c(M) = (Q', \Sigma', K, (\lambda'_f)_{f\in\Sigma'}, \mu', \omega') \ ,$$

where

- $Q' = Q \cup \bigcup_{f\in\Sigma'} Q_f$;
- $\lambda'_f(q) = \lambda_f(q)$ for every $f \in \Sigma'$ and $q \in Q_f$;
- $\mu'(q)_{p,p'} = \mu_f(q)_{p,p'}$ for every $f \in \Sigma'$, $p \in Q_f$, $q \in Q \cup \{\Diamond\}$, and $p' \in Q \cup Q_f$;
- $\omega'(q) = \omega(q)$ for every $q \in Q$; and
- all remaining entries in λ'_f, μ', and ω' are 0. □

We note that $c(M)$ is deterministic if and only if M is deterministic. Second, supposing a suitable representation of M, the construction of $c(M)$ can be performed in constant time.

Now let us relate the semantics of $c(M)$ to the one of M. Clearly, the two devices run on trees over different alphabets, so we again first define an input tree translation. Let $c\colon U_\Sigma \to U_{\Sigma'}$ be recursively defined by $c(f[t_1, \ldots, t_k]) = f[c(t_1), \ldots, c(t_k), \Diamond]$ for every $f \in \Sigma$ and $t_1, \ldots, t_k \in U_\Sigma$. Clearly, c is injective. This mapping is lifted to tree series as follows: $c\colon K\langle\langle U_\Sigma\rangle\rangle \to K\langle\langle U_{\Sigma'}\rangle\rangle$ by $(c(\varphi), u) = \sum_{t\in c^{-1}(u)} (\varphi, t)$ for every $u \in U_{\Sigma'}$. Note that the extended mapping c is still injective.

Lemma 5.5 $c(\|M\|) = \|c(M)\|$.

Proof Let $u \in U_{\Sigma'}$. If $c^{-1}(u) = \emptyset$, then it is easily seen that $(c(\|M\|), u) = 0 = (\|c(M)\|, u)$. In the remaining case, let $t \in c^{-1}(u)$; note that t is unique because c is injective.

$$(c(\|M\|), u) = (\|M\|, t) = \sum_{r\in run_M(t)} weight_M(r) \cdot \omega(r(\varepsilon))$$

$$= \sum_{r \in run_M(t)} \Big(\prod_{\substack{w \in Pos(t) \\ k = rank_t(w)}} (\|A_{t(w)}\|, r(w1) \cdots r(wk))_{r(w)} \Big) \cdot \omega(r(\varepsilon))$$

$= \{$see the part of wsa with P-output on page 7$\}$

$$\sum_{r \in run_M(t)} \Big(\prod_{\substack{w \in Pos(t) \\ k = rank_t(w)}} (\|(sim(A_{t(w)})^{r(w)}\|, r(w1) \cdots r(wk)\Diamond) \Big) \cdot \omega(r(\varepsilon))$$

$= \{$because of the definition of $A_\Diamond\}$

$$\sum_{r \in run_{c(M)}(u)} \Big(\prod_{\substack{w \in Pos(u) \\ k = rank_u(w)}} (\|c(M)^{r(w)}_{t(w)}\|, r(w1) \cdots r(wk)) \Big) \cdot \omega(r(\varepsilon))$$

$$= \sum_{r \in run_{c(M)}(u)} weight_{c(M)}(r) \cdot \omega(r(\varepsilon)) = (\|c(M)\|, u) \qquad \blacksquare$$

Let us first consider backward bisimulation. We define the notion of backward bisimulation for M and then relate it to the corresponding notion for $c(M)$. Fortunately, the relation will be close enough to allow us to reduce the problem of backward bisimulation minimisation of M to the corresponding problem of $c(M)$. In the sequel, let $\mathcal{R}' = \mathcal{R} \cup \bigcup_{f \in \Sigma'} \mathcal{R}_f$, where $\mathcal{R}$ is an equivalence relation on Q and $\mathcal{R}_f$ is an equivalence relation on Q_f for every $f \in \Sigma'$.

Definition 5.6 We say that $\mathcal{R}'$ is a *backward bisimulation on* M if $\mathcal{R} \cup \mathcal{R}_f$ is a backward bisimulation on $sim(A_f)^{\mathcal{R}}$ for every $f \in \Sigma'$. □

Lemma 5.7 The relation $\mathcal{R}'$ is a backward bisimulation on M if and only if $\mathcal{R}'$ is a backward bisimulation on $c(M)$.

Proof Let $c(M) = (Q', \Sigma', K, (\lambda'_f)_{f \in \Sigma'}, \mu', \omega')$ and $(q_1, q_2) \in \mathcal{R}'$. First suppose $\mathcal{R}'$ is a backward bisimulation on M. For every $B_1 \in (Q \cup \{\Diamond\})/\mathcal{R}'$, $f \in \Sigma'$, and $B_2 \in Q_f/\mathcal{R}_f$ we have

$$\sum_{\substack{q \in B_1 \\ p \in B_2}} \mu'(q)_{p,q_1} = \sum_{\substack{q \in B_1 \\ p \in B_2}} \mu'(q)_{p,q_2}$$

by the construction of $c(M)$ and because $\mathcal{R} \cup \mathcal{R}_f$ is a backward bisimulation on $sim(A_f)^{\mathcal{R}}$. For other blocks B_1 and B_2 we have that both sums are 0. Now, let $(q_1, q_2) \in \mathcal{R}$. Then $\lambda'_f(q_1) = 0 = \lambda'_f(q_2)$ for every $f \in \Sigma'$. On the other hand, if $(q_1, q_2) \in \mathcal{R}_f$ for some $f \in \Sigma'$, then $\lambda'_f(q_1) = \lambda_f(q_1) = \lambda_f(q_2) = \lambda'_f(q_2)$ since $\mathcal{R} \cup \mathcal{R}_f$ is a backward bisimulation on $sim(A_f)^{\mathcal{R}}$. Moreover, $\lambda'_{f'}(q_1) = 0 = \lambda'_{f'}(q_2)$ for every $f' \neq f$. It follows that $\mathcal{R}'$ is a backward bisimulation on $c(M)$. The converse is proved in the same manner. ∎

By definition every backward bisimulation $\mathcal{R}'$ on M is a refinement of the equivalence induced by the partition $\Pi = \{Q\} \cup \{Q_f \mid f \in \Sigma'\}$. Thus one direction of

Lemma 5.7 is readily applicable. If we run the backward bisimulation minimisation algorithm initialised with the partition Π on $c(M)$, then we obtain the coarsest backward bisimulation $\mathcal{P}'$ on $c(M)$ that is a refinement of Π (by [22, Lemma 4.13] and a minor modification of [22, Lemma 4.14]). By Lemma 5.7, $\mathcal{P}'$ is also a backward bisimulation on M. In fact, it must be the coarsest backward bisimulation on M because any coarser backward bisimulation $\mathcal{P}''$ on M is naturally a refinement of Π and a backward bisimulation on $c(M)$ by Lemma 5.7. However, the algorithm returned the coarsest backward bisimulation on $c(M)$ that is a refinement of Π, which yields that $\mathcal{P}' = \mathcal{P}''$. Thus, we can efficiently compute the coarsest backward bisimulation on M using the backward bisimulation minimisation algorithm for $c(M)$.

It remains to present how to collapse M with respect to a backward bisimulation $\mathcal{R}'$. Moreover, the collapsed wputa should clearly recognise the same unranked tree series as M.

Definition 5.8 If $\mathcal{R}'$ is a backward bisimulation on M, then the *backward-aggregated wputa* is $M/\mathcal{R}' = (Q/\mathcal{R}, \Sigma, K, (A'_f)_{f\in\Sigma}, \omega')$ with

(i) $A'_f = (Q_f/\mathcal{R}_f, Q/\mathcal{R}, K, \lambda'_f, \mu'_f, \nu'_f)$ where
- $\lambda'_f([p]) = \lambda_f(p)$ for every $p \in Q_f$;
- $\mu'_f(D)_{B,[p']} = \sum_{q\in D, p\in B} \mu_f(q)_{p,p'}$ for every $f \in \Sigma$, $D \in Q/\mathcal{R}$, $B \in Q_f/\mathcal{R}_f$, and $p' \in Q_f$; and
- $\nu'_f(B)_{[q]} = \sum_{p\in B} \nu_f(p)_q$ for every $f \in \Sigma$, $B \in Q_f/\mathcal{R}_f$, and $q \in Q$;

(ii) $\omega'(D) = \sum_{q\in D} \omega(q)$ for every $D \in Q/\mathcal{R}$. □

Finally, let us prove that collapsing preserves the semantics. We achieve this in an indirect manner using $c(M)$.

Lemma 5.9 If $\mathcal{R}'$ is a backward bisimulation on M, then $c(M/\mathcal{R}')$ is isomorphic to $c(M)/\mathcal{R}'$.

Proof Let $M/\mathcal{R}' = (Q', \Sigma, K, (A'_f)_{f\in\Sigma}, \omega')$ [see Definition 5.8]. We easily observe that $sim(A'_f)$ is isomorphic to $sim(A_f)^{\mathcal{R}}/(\mathcal{R} \cup \mathcal{R}_f)$. Hence $c(M/\mathcal{R}')$, which is essentially the union of the various $sim(A'_f)$ and $A_\Diamond$, can be seen as the union of the various $sim(A_f)^{\mathcal{R}}/(\mathcal{R} \cup \mathcal{R}_f)$ and $A_\Diamond$. This union is isomorphic to $c(M)/\mathcal{R}'$. Let us formally verify this to convince the reader.

The wsuta $c(M)/\mathcal{R}'$ is $(Q'', \Sigma', K, (\lambda''_f)_{f\in\Sigma'}, \mu'', \omega'')$ where
- $Q'' = (Q \cup \bigcup_{f\in\Sigma'} Q_f)/\mathcal{R}' = Q/\mathcal{R} \cup \bigcup_{f\in\Sigma'} Q_f/\mathcal{R}_f$;
- $\lambda''_f([p]) = \lambda_f(p)$ for every $f \in \Sigma'$ and $p \in Q_f$;
- $\mu''(D)_{B,[p']} = \sum_{q\in D, p\in B} \mu_f(q)_{p,p'}$ for every $D \in Q/\mathcal{R} \cup \{\{\Diamond\}\}$, $f \in \Sigma'$, $B \in Q_f/\mathcal{R}_f$, and $p' \in Q \cup Q_f$;
- $\omega''(D) = \sum_{q\in D} \omega(q)$ for every $D \in Q/\mathcal{R}$; and
- all remaining entries in λ''_f, μ'', and ω'' are 0.

A straightforward check now shows that $c(M)/\mathcal{R}'$ is isomorphic to the wsuta $c(M/\mathcal{R}')$. In fact, the isomorphism identifies $\{\Diamond\}$ and $\Diamond$. ∎

As usual, we say that M is backward bisimulation minimal if it allows only the identity as backward bisimulation. Now we can exploit the established relation and formulate our main theorem of this section.

Theorem 5.10 A backward bisimulation minimal wputa that is equivalent to M can be computed in time $O(mn)$ where $n = |Q| + \sum_{f\in\Sigma} |Q_f|$ and m is the sum over all $f \in \Sigma$ of the number of nonzero entries in μ_f. If K is additively cancellative or the BOOLEAN semiring, then we can achieve it in time $O(m \log n)$.

Proof We first construct the wsuta $c(M)$. By a slight modification of the method used to prove Theorem 4.9, we can compute the coarsest backward bisimulation $\mathcal{R}'$ on $c(M)$ that refines Π in time $O(mn)$ (resp., in time $O(m \log n)$ if K is additively cancellative or the BOOLEAN semiring). We already argued that $\mathcal{R}'$ is then the coarsest backward bisimulation on M. Similar to [21, Lemma 26], we can compute $M/\mathcal{R}'$ in time $O(m)$. So we constructed a backward bisimulation minimal wputa $M/\mathcal{R}'$ in the given time bound. It remains to show that $M/\mathcal{R}'$ and M are equivalent. We know that $c(\|M\|) = \|c(M)\|$ by Lemma 5.5. Moreover, we know that $c(M)/\mathcal{R}'$ and $c(M)$ are equivalent by Theorem 4.9. By Lemma 5.9, $c(M/\mathcal{R}')$ is isomorphic and hence equivalent to $c(M)/\mathcal{R}'$. Thus, we obtain

$$c(\|M/\mathcal{R}'\|) = \|c(M/\mathcal{R}')\| = \|c(M)/\mathcal{R}'\| = \|c(M)\| = c(\|M\|)$$

with the help of Lemma 5.5. Since c is injective, we conclude that $M/\mathcal{R}'$ and M are equivalent. ∎

Let us now investigate forward bisimulation. We follow the same approach as in the backward case and first define forward bisimulation such that every forward bisimulation on M is also a forward bisimulation on $c(M)$. Recall that $\mathcal{R}' = \mathcal{R} \cup \bigcup_{f\in\Sigma'} \mathcal{R}_f$, where $\mathcal{R}$ is an equivalence relation on Q and $\mathcal{R}_f$ is an equivalence relation on Q_f for every $f \in \Sigma'$.

Definition 5.11 We say that $\mathcal{R}'$ is a *forward bisimulation on* M if

- $\omega(q_1) = \omega(q_2)$ for every $(q_1, q_2) \in \mathcal{R}$;
- $\mathcal{R} \cup \mathcal{R}_f$ is a forward bisimulation on $sim(A_f)$ for every $f \in \Sigma'$; and
- for every $(q_1, q_2) \in \mathcal{R}$, $f \in \Sigma'$, $p \in Q_f$, and block $B \in Q_f/\mathcal{R}_f$

$$\sum_{p'\in B} \mu_f(q_1)_{p,p'} = \sum_{p'\in B} \mu_f(q_2)_{p,p'} \quad .$$ □

Lemma 5.12 The relation $\mathcal{R}'$ is a forward bisimulation on M if and only if $\mathcal{R}'$ is a forward bisimulation on $c(M)$.

Proof We leave the proof of this fact as an exercise. ∎

The argument that we used to show that we can compute the coarsest backward bisimulation on M can also be used for the forward case (this time using variations of [22, Lemmata 3.17 and 3.18]), so that we can compute the coarsest forward bisimulation on M by computing the coarsest forward bisimulation on $c(M)$ that is a refinement of Π.

Definition 5.13 Suppose that $\mathcal{R}'$ is a forward bisimulation on M. Then the *forward-aggregated wputa* is $M/\mathcal{R}' = (Q/\mathcal{R}, \Sigma, K, (A'_f)_{f\in\Sigma}, \omega')$ where

(i) $A'_f = (Q_f/\mathcal{R}_f, Q/\mathcal{R}, K, \lambda'_f, \mu'_f, \nu'_f)$ with
- $\lambda'_f(B) = \sum_{p\in B} \lambda_f(p)$ for every $B \in Q_f/\mathcal{R}_f$;
- $\mu'_f([q])_{[p],B} = \sum_{p'\in B} \mu_f(q)_{p,p'}$ for every $q \in Q$, $p \in Q_f$, and $B \in Q_f/\mathcal{R}_f$; and
- $\nu'_f([p])_D = \sum_{q\in D} \nu_f(p)_q$ for every $p \in Q_f$ and $D \in Q/\mathcal{R}$;

(ii) $\omega'([q]) = \omega(q)$ for every $q \in Q$. □

Lemma 5.14 If $\mathcal{R}'$ is a forward bisimulation on M, then $c(M/\mathcal{R}')$ and $c(M)/\mathcal{R}'$ are isomorphic.

Proof The wsuta $c(M)/\mathcal{R}' = (Q', \Sigma', K, (\lambda'_f)_{f\in\Sigma'}, \mu', \omega')$ is given by

- $Q' = (Q \cup \bigcup_{f\in\Sigma'} Q_f)/\mathcal{R}' = Q/\mathcal{R} \cup \bigcup_{f\in\Sigma'} Q_f/\mathcal{R}_f$;
- $\lambda'_f(B) = \sum_{p\in B} \lambda_f(p)$ for every $f \in \Sigma'$ and $B \in Q_f/\mathcal{R}_f$;
- $\mu'([q])_{[p],B} = \sum_{p'\in B} \mu_f(q)_{p,p'}$ for every $f \in \Sigma'$, $q \in Q \cup \{\Diamond\}$, $p \in Q_f$, and $B \in (Q \cup Q_f)/\mathcal{R}'$;
- $\omega'([q]) = \omega(q)$ for every $q \in Q$; and
- all remaining entries in λ'_f, μ', and ω' are 0.

We leave it to the reader to confirm that it is isomorphic to $c(M/\mathcal{R}')$. ∎

The wputa is forward bisimulation minimal if it admits only the identity as forward bisimulation.

Theorem 5.15 For every wputa M we can compute a forward bisimulation minimal equivalent wputa in time $O(mn)$ where $n = |Q| + \sum_{f\in\Sigma} |Q_f|$ and m is the sum over all $f \in \Sigma$ of the number of nonzero entries in μ_f. If K is additively cancellative or the BOOLEAN semiring, then we can achieve it in time $O(m \log n)$.

Proof The proof mimics the one of Theorem 5.10. ∎

Let us finally look at determinism. By definition, M is deterministic if $c(M)$ is deterministic. Let M be deterministic and $\mathcal{R}'$ be a forward bisimulation on M. Clearly, $c(M)$ and $c(M)/\mathcal{R}'$ are then deterministic. Since $c(M/\mathcal{R}')$ and $c(M)/\mathcal{R}'$ are isomorphic by Lemma 5.14, we obtain that $c(M/\mathcal{R}')$ and consequently also $M/\mathcal{R}'$ are deterministic. Thus we arrive at the following theorem, which shows that minimisation of deterministic puta can be achieved in time $O(m \log n)$. This improves [9, Theorem 4] where a quadratic time-bound was shown.

Theorem 5.16 For every deterministic puta M we can compute a minimal equivalent deterministic puta in time $O(m \log n)$.

Proof Without loss of generality, suppose that M is total and has no unreachable states (these are the requirements for the algorithm of [9]; see there for details). We already proved in Theorem 5.15 that we can compute $M/\mathcal{R}'$ where $\mathcal{R}'$ is the coarsest forward bisimulation on M in time $O(m \log n)$. Moreover, we know that $M/\mathcal{R}'$ is deterministic and equivalent to M. It remains to show that it is minimal. To this end, consider Algorithm EQUIVALENT-STATES of [9]. Let $\mathcal{P}$ be the output of this algorithm when run on M, and let $\mathcal{P}' = \mathcal{P} \cup \{(\Diamond, \Diamond)\}$. We show that $\mathcal{P}'$ is a forward bisimulation on M. It is immediately clear by line 7 that $\mathcal{P}'$ is a refinement of Π. Let $(q_1, q_2) \in \mathcal{P}'$. Let us list the consequences of each line (using also the fact that M is deterministic):

1. If $q_1, q_2 \in Q$, then $\omega(q_1) = \omega(q_2)$.
2. If $q_1, q_2 \in Q_f$ for some $f \in \Sigma$, then $\bigvee_{q \in D} \mu_f(\Diamond)_{q_1,q} = \bigvee_{q \in D} \mu_f(\Diamond)_{q_2,q}$ for every $D \in Q/\mathcal{P}'$.
3. If $q_1, q_2 \in Q_f$ for some $f \in \Sigma$, then $\bigvee_{p \in B} \mu_f(q)_{q_1,p} = \bigvee_{p \in B} \mu_f(q)_{q_2,p}$ for every $q \in Q$ and $B \in Q_f/\mathcal{P}'$.
4. If $q_1, q_2 \in Q$, then $\bigvee_{p' \in B} \mu_f(q_1)_{p,p'} = \bigvee_{p' \in B} \mu_f(q_2)_{p,p'}$ for every $f \in \Sigma$, $p \in Q_f$, and $B \in Q_f/\mathcal{P}'$.

Obviously, $\mathcal{P}'$ is a forward bisimulation on M (see Definition 5.11). However, $\mathcal{R}'$ is the coarsest forward bisimulation on M that refines Π, so consequently, $\mathcal{P}' = \mathcal{R}'$. Finally, note that $(\Diamond, \Diamond) \in \mathcal{P}'$ does not influence $M/\mathcal{P}'$. Thus, $M/\mathcal{R}'$ is a minimal equivalent puta. ∎

6 Comparison of computational power

The last automaton model for unranked trees that we consider is the *weighted unranked tree automaton* (wuta). This generalisation of fta on unranked trees [6] is obtained by augmenting transitions with semiring weights [12]. For every state q and input symbol f, a wuta has a recognisable power series $\delta(q, f)$ on its state space. It assigns to a state q at an f-labelled position w the weight $(\delta(q, f), q_1 \cdots q_k)$, where $q_1, \ldots, q_k$ are the states that are assigned to the positions immediately below w. Let us formalise this idea.

Definition 6.1 A *weighted unranked tree automaton* (abbreviated wuta) is a tuple $(Q, \Sigma, K, \delta, \omega)$ where

- Q is an alphabet (of *states*);
- Σ is an alphabet (of *input symbols*);
- K is a semiring;

- $\delta\colon \Sigma \times Q \to Rec(K, Q)$ is a *transition mapping*; and
- $\omega \in K^Q$ is a *root weight vector*. □

Definition 6.2 Let $M = (Q, \Sigma, K, \delta, \omega)$ be a wuta. The set of all *runs* of M on $t \in U_\Sigma$, denoted by $run_M(t)$, is the set of all mappings $Pos(t) \to Q$. The *weight* of a run $r \in run_M(t)$ is given by

$$weight_M(r) = \prod_{\substack{w \in Pos(t) \\ k = rank_t(w)}} (\delta(t(w), r(w)), r(w1) \cdots r(wk)) \ .$$

The *unranked tree series* $\|M\|$ recognised by M is defined for every $t \in U_\Sigma$ by

$$(\|M\|, t) = \sum_{r \in run_M(t)} weight_M(r) \cdot \omega(r(\varepsilon)) \ . \qquad \square$$

Lemma 6.3 For every wuta M there is a wputa M' such that $\|M'\| = \|M\|$.

Proof Let $M = (Q, \Sigma, K, \delta, \omega)$ be a wuta. Clearly, for every $f \in \Sigma$ there exists a wsa $A_f = (Q_f, Q, K, \lambda_f, \mu_f, \nu_f)$ such that $\|A_f^q\| = \delta(q, f)$ for every $q \in Q$ (see [13]). We construct the wputa $M' = (Q, \Sigma, K, (A'_f)_{f \in \Sigma}, \omega)$ where for every $f \in \Sigma$ we have $A'_f = (Q_f, Q, K, \lambda_f, \mu_f, \nu'_f)$ with $(\nu'_f)_{q,q} = \nu_f(q)$ for every $q \in Q$ and $(\nu'_f)_{q,p} = 0$ for all $p, q \in Q$ such that $p \neq q$. It is clear that $(\|A_f^q\|, w) = (\|A'_f\|, w)_q$ for every $f \in \Sigma$, $w \in Q^*$, and $q \in Q$. Now let $t \in U_\Sigma$. Clearly, $run_M(t) = run_{M'}(t)$, so it remains to prove that $weight_M(r) = weight_{M'}(r)$ for every $r \in run_M(t)$.

$$\begin{aligned} weight_M(r) &= \prod_{\substack{w \in Pos(t) \\ k = rank_t(w)}} (\delta(t(w), r(w)), r(w1) \cdots r(wk)) \\ &= \prod_{\substack{w \in Pos(t) \\ k = rank_t(w)}} (\|A'_{t(w)}\|, r(w1) \cdots r(wk))_{r(w)} = weight_{M'}(r) \end{aligned}$$ ■

Lemma 6.4 For every wsuta M there is a wuta M' such that $\|M'\| = \|M\|$.

Proof Let $M = (Q, \Sigma, K, (\lambda_f)_{f \in \Sigma}, \mu, \omega)$ be a wsuta. Consider the wuta $M' = (Q, \Sigma, K, \delta, \omega)$ such that $\delta(f, q) = \|(M)_f^q\|$ for every $f \in \Sigma$ and $q \in Q$. The statement $\|M'\| = \|M\|$ is obvious. ■

Lemma 6.5 For every wputa M there is a wsuta M' such that $\|M'\| = \|M\|$.

Proof Let $M = (Q, \Sigma, K, (A_f)_{f \in \Sigma}, \omega)$ with $A_f = (Q_f, Q, K, \lambda_f, \mu_f, \nu_f)$ for every $f \in \Sigma$. Without loss of generality, suppose that Q and all Q_f with $f \in \Sigma$ are pairwise disjoint. We construct the wsuta $M' = (Q', \Sigma, K, (\lambda'_f)_{f \in \Sigma}, \mu', \omega')$ such that

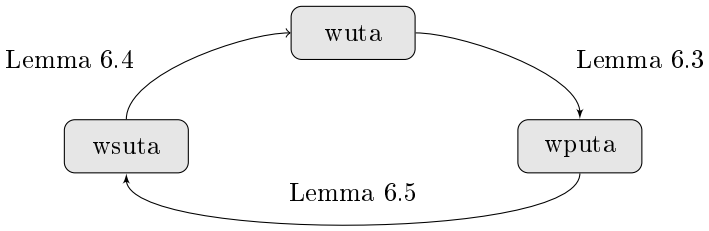


Figure 4: The automata models wsuta, wputa, and wuta have the same computational power.

- $Q' = Q \cup \bigcup_{f \in \Sigma} Q_f$;
- for every $f \in \Sigma$ and $q \in Q \cup Q_f$ let

$$\lambda'_f(q) = \begin{cases} \lambda_f(q) & \text{if } q \in Q_f \\ (\lambda_f \cdot \nu_f)_q & \text{if } q \in Q \end{cases}$$

- for every $f \in \Sigma$, $p_1 \in Q_f$, $p_2 \in Q \cup Q_f$, and $q \in Q$ let

$$\mu'(q)_{p_1,p_2} = \begin{cases} \mu_f(q)_{p_1,p_2} & \text{if } p_2 \in Q_f \\ (\mu_f(q) \cdot \nu_f)_{p_1,p_2} & \text{if } p_2 \in Q \end{cases}$$

- $\omega'(q) = \omega(q)$ for every $q \in Q$.
- All remaining entries in λ_f, μ, and ω are 0.

We first prove that $(\|A_f\|, w)_q = (\|(M')^q_f\|, w)$ for every $f \in \Sigma$, $w \in Q^*$, and $q \in Q$. Let $w = q_1 \cdots q_n$ for some $q_1, \ldots, q_n \in Q$. If $w = \varepsilon$, then

$$(\|A_f\|, \varepsilon)_q = (\lambda_f \cdot \nu_f)_q = \lambda'_f(q) = \sum_{p \in Q'} \lambda'_f(p) \cdot \overline{\mu}'(\varepsilon)_{p,q} = (\|(M')^q_f\|, \varepsilon) \ .$$

Otherwise, $w = vq_n$ and

$$(\|A_f\|, w)_q = \left(\lambda_f \cdot \overline{\mu}_f(v) \cdot \mu_f(q_n) \cdot \nu_f\right)_q = \left(\lambda_f \cdot \overline{\mu}'(v) \cdot \mu'(q_n)\right)_q = (\|(M')^q_f\|, w).$$

Let $t \in U_\Sigma$. Note that $weight_{M'}(r) = 0$ for every $r \in run_{M'}(t) \setminus run_M(t)$. Thus it remains to prove that $weight_{M'}(r) = weight_M(r)$ for every $r \in run_M(t)$.

$$\begin{aligned} weight_{M'}(r) &= \prod_{\substack{w \in Pos(t) \\ k = rank_t(w)}} (\|(M')^{r(w)}_{t(w)}\|, r(w1) \cdots r(wk)) \\ &= \prod_{\substack{w \in Pos(t) \\ k = rank_t(w)}} (\|A_{t(w)}\|, r(w1) \cdots r(wk))_{r(w)} = weight_M(r) \ , \end{aligned}$$

which proves the statement. ∎

We summarise Lemmata 6.5, 6.4, and 6.3 in Theorem 6.6.

Theorem 6.6 Let $\mathcal{S} \in K\langle\langle U_\Sigma \rangle\rangle$. The following statements are equivalent:

- $\mathcal{S}$ is recognisable by a wputa.
- $\mathcal{S}$ is recognisable by a wsuta.
- $\mathcal{S}$ is recognisable by a wuta.

References

[1] P. A. Abdulla, J. Högberg, and L. Kaati. Bisimulation minimization of tree automata. *Int. J. Foundations of Computer Science*, 18(4):699–713, 2007.

[2] S. Abiteboul, P. Bunemann, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[3] A. Alexandrakis and S. Bozapalidis. Weighted grammars and Kleene's theorem. *Information Processing Letters*, 24(1):1–4, 1987.

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

[5] B. Borchardt. *The theory of recognizable tree series*. Akademische Abhandlungen zur Informatik. Verlag für Wissenschaft und Forschung, 2005.

[6] A. Brüggemann-Klein, M. Murata, and D. Wood. Regular tree and regular hedge languages over unranked alphabets. Technical Report HKUST-TCSC-2001-0, The Hongkong University of Science and Technology, 2001.

[7] P. Buchholz. Bisimulation relations for weighted automata. *Theoretical Computer Science*, 2008. to appear.

[8] J. Carme, J. Niehren, and M. Tommasi. Querying unranked trees with stepwise tree automata. In *Proc. 15th Int. Conf. Rewriting Techniques and Applications*, volume 3091 of *LNCS*, pages 105–118. Springer, 2004.

[9] J. Cristau, C. Löding, and W. Thomas. Deterministic automata on unranked trees. In *Proc. 15th Int. Symp. Fundamentals of Computation Theory*, volume 3623 of *LNCS*, pages 68–79. Springer, 2005.

[10] M. Droste, W. Kuich, and H. Vogler, editors. *Handbook of Weighted Automata*. Springer, 2008. to appear.

[11] M. Droste, C. Pech, and H. Vogler. A Kleene theorem for weighted tree automata. *Theory of Computing Systems*, 38(1):1–38, 2005.

[12] M. Droste and H. Vogler. Weighted logics for XML. manuscript, 2008.

[13] S. Eilenberg. *Automata, Languages, and Machines – Volume A*, volume 59 of *Pure and Applied Mathematics*. Academic Press, 1974.

[14] Z. Ésik and W. Kuich. Formal tree series. *J. Automata, Languages and Combinatorics*, 8(2):219–285, 2003.

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

[16] F. Gécseg and M. Steinby. Tree languages. In *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer, 1997.

[17] J. Golan. *Semirings and their applications*. Kluwer Academic, 1999.

[18] 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, 2005.

[19] 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.

[20] J. Högberg, A. Maletti, and J. May. Backward and forward bisimulation minimisation of tree automata. In *Proc. 12th Int. Conf. Implementation and Application of Automata*, volume 4783 of LNCS, pages 109–121. Springer, 2007.

[21] J. Högberg, A. Maletti, and J. May. Bisimulation minimisation for weighted tree automata. In *Proc. 11th Int. Conf. Developments in Language Theory*, volume 4588 of LNCS, pages 229–241. Springer, 2007.

[22] J. Högberg, A. Maletti, and J. May. Bisimulation minimisation for weighted tree automata. Technical Report ISI-TR-634, University of Southern California, 2007.

[23] T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.

[24] N. Klarlund, Th. Schwentick, and D. Suciu. XML: Models, Schemas, Types, Logics, and Queries. In *Proc. Dagstuhl Seminar: Logics for Emerging Applications on Databases*, pages 1–41. Springer, 2003.

[25] A. Malcher. Minimizing finite automata is computationally hard. *Theoretical Computer Science*, 327(3):375–390, 2004.

[26] W. Martens and J. Niehren. Minimizing tree automata for unranked trees. In *Proc. 10th Int. Symp. Database Programming Languages*, volume 3774 of *LNCS*, pages 232–246. Springer, 2005.

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

[28] R. Milner. *A Calculus of Communicating Systems*. Springer, 1982.

[29] F. Neven. Automata, logic, and XML. In *Proc. 16th Int. Workshop Computer Science Logic*, volume 2471 of *LNCS*, pages 2–26. Springer, 2002.

[30] M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4:245–270, 1961.