Equivalences between Ranked and Unranked Weighted Tree Automata via Binarization

Toni Dietze
Faculty of Computer Science
Technische Universität Dresden
01062 Dresden, Germany
toni.dietze@tu-dresden.de

Abstract

Encoding unranked trees to binary trees, henceforth called binarization, is an important method to deal with unranked trees. For each of three binarizations we show that weighted (ranked) tree automata together with the binarization are equivalent to weighted unranked tree automata; even in the probabilistic case. This allows to easily adapt training methods for weighted (ranked) tree automata to weighted unranked tree automata.

1 Introduction

When dealing with trees, tree grammars, and formal languages of trees, a restriction to binary trees is often beneficial, e.g. for improved generalization when generating grammars from treebanks, or for reduced parsing complexity. A *binarization* maps any (unranked) tree to a binary tree such that the original tree can be reconstructed from the result.

In this paper we investigate three different binarizations inspired by Matsuzaki et al. (2005, Fig. 6). For each of these binarizations we show that a weighted unranked tree language is recognizable by a weighted unranked tree automaton (wuta) iff the binarization of the language is recognizable by a weighted tree automaton (wta). This even holds with restriction to probabilistic automata. To support this result we show that for any $\mathbb{R}_{\geq 0}$-weighted finite state automaton with the sum of weights of all words being 1, there is an equivalent probabilistic finite state automaton. This implies that the class of weighted string languages recognizable by probabilistic finite state automata is closed under reversal.

These results suggest that by adding binarization to training methods for wta we effectively get training methods for wuta. Alterations to the weights and the state behaviour while training then carry over to wuta. This also gives an explanation for why the performance of the training by Matsuzaki et al. (2005) is rather independent from the used binarization.

Related Work. Fülöp and Vogler (2009) give an overview over wta, and Droste and Vogler (2011) introduced wuta. For the unweighted case binarizations (also called encodings) were investigated by, e.g., Comon et al. (2007, Section 8.3). Their first-child-next-sibling encoding is similar to our left-branching binarization. Their extension encoding is also used to define stepwise tree automata (Carme et al., 2004). A stepwise tree automaton is defined like a (ranked) tree automaton. It accepts an unranked tree, if it accepts the extension encoding of the tree while the automaton is interpreted as a (ranked) tree automaton. Högberg et al. (2009, Lemmas 4.2 and 6.2) extend this connection to the weighted case and show that weighted stepwise tree automata and wuta are equally powerful.

Our results for weighted and for probabilistic finite state automata are a special case of *renormalization* of weighted or of probabilistic context-free grammars (Abney et al., 1999; Chi, 1999; Nederhof and Satta, 2003). We provide alternative proofs for this special case.

2 Preliminaries

Let A and B be sets, and $R \subseteq A \times B$ a relation. By $R(a)$ we denote the set $\{b \in B \mid (a, b) \in R\}$ for every $a \in A$. The *inverse relation of* R is defined by $R^{-1} = \{(b, a) \mid (a, b) \in R\}$. Occasionally we identify a function $f\colon A \to B$ with the relation $\{(a, f(a)) \mid a \in A\}$.

An *alphabet* is a finite, non-empty set. Let Σ be an alphabet. The *set of words over* Σ is denoted by Σ^*. The length of a word $w \in \Sigma^*$ is denoted by $|w|$. The *empty word*, denoted by ε, is the word of length 0. The set of *unranked trees over* Σ, denoted by U_Σ, is the smallest set U such that for every $\sigma \in \Sigma$, $k \in \mathbb{N}$, and $t_1, \dots, t_k \in U$ we have $\sigma(t_1, \dots, t_k) \in U$. We abbreviate $\sigma() \in \mathrm{U}_\Sigma$ by σ. Let $t = \sigma(t_1, \dots, t_k) \in \mathrm{U}_\Sigma$. We call σ the *root symbol of* t. The *root rank of* t is defined by $\mathrm{rk}(t) = k$. The *set of positions of* t is recursively defined by $\mathrm{pos}(t) = \varepsilon \cup \bigcup_{i \in \{1,\dots,k\}} \{i\rho \mid \rho \in \mathrm{pos}(t_i)\}$. Let $\rho \in \mathrm{pos}(t)$. The *subtree of* t *at* ρ is recursively defined by $t|_\rho = t$ if $\rho = \varepsilon$ and $t|_\rho = t_i|_{\rho'}$ if $\rho = i\rho'$. The *symbol of* t *at* ρ, denoted by $t(\rho)$, is defined as the root symbol of $t|_\rho$. The *rank of* t *at* ρ is defined as $\mathrm{rk}(t|_\rho)$.

Let S be a non-empty set (of *sorts*). An *S-sorted alphabet* is a family $\Sigma = (\Sigma^{(s)} \mid s \in S)$ of pairwise disjoint sets such that their union is non-empty and finite. For families of sets we often use the same identifier for denoting the family and the union of its members. Let Σ be an $(S \times S^*)$-sorted alphabet. The *family of (S-sorted) trees over* Σ, denoted by $\mathrm{T}_\Sigma = (\mathrm{T}_\Sigma^{(s)} \mid s \in S)$, is defined as the smallest family $(T^{(s)} \mid s \in S)$ such that for every $(s_0, s_1 \dots s_k) \in (S \times S^*)$, for every $\sigma \in \Sigma^{(s_0, s_1 \dots s_k)}$, for every $t_1 \in T^{(s_1)}, \dots, t_k \in T^{(s_k)}$, we have $\sigma(t_1, \dots, t_k) \in T^{(s_0)}$. If S is a singleton, this is equivalent to the usual definition of ranked trees. Note that $\mathrm{T}_\Sigma \subseteq \mathrm{U}_\Sigma$, so all notions for unranked trees are also valid for sorted trees. Also note that, for sorted trees, the symbol at a position determines the rank at this position; therefore we will use rk also for symbols from Σ.

A commutative semiring is an algebraic structure $\Re = (R, +, \cdot, 0, 1)$ such that $(R, +, 0)$ and $(R, \cdot, 1)$ are commutative monoids, $\cdot$ is distributive over $+$, and 0 is annihilating w.r.t. $\cdot$. We often write $\Re$ instead of R. Let A be a set and $f \colon A \to \Re$ be a mapping. The *support of* f is defined by $\mathrm{supp}(f) = \{a \in A \mid f(a) \neq 0\}$.

In the following we will define devices to associate weights with sorted trees, words, or unranked trees. We will use weights from an arbitrary commutative semiring. Therefore, if we use addition, multiplication, 0, and 1, then these are operations from that semiring. We will call two such devices equivalent, if their (yet to define) semantics $[\![\cdot]\!]$ is the same.

Definition 1 (wsta)**.** Let S be a set of sorts and $\Re$ be a commutative semiring. An *$\Re$-weighted S-sorted tree automaton ($\Re$-S-wsta)* is a tuple (Q, Σ, I, Δ) where

- Q is an S-sorted alphabet (of *states*),
- Σ is an $(S \times S^*)$-sorted alphabet (of *terminals*),
- $I \colon Q \to \Re$ is a mapping (*root weights*), and
- $\Delta = (\Delta^{(\sigma)} \colon Q^{(s_0)} \times \dots \times Q^{(s_k)} \to \Re \mid (s_0, s_1 \dots s_k) \in S \times S^*,\ \sigma \in \Sigma^{(s_0, s_1 \dots s_k)})$ is a family of mappings (*transition weights*). □

Let $\mathcal{M} = (Q, \Sigma, I, \Delta)$ be an $\Re$-S-wsta. The *size of* $\mathcal{M}$ is defined by $\mathrm{size}(\mathcal{M}) = \mathrm{supp}(I) + \sum_{\sigma \in \Sigma} \mathrm{supp}(\Delta^{(\sigma)})$. For some $s \in S$, we call $\mathcal{M}$ *s-rooted*, if for every $s' \in S \backslash \{s\}$ and $q \in Q^{(s')}$ we have that $I(q) = 0$. We define the relation $\mathrm{run}_\mathcal{M}$ that contains (t, r), if $t \in \mathrm{T}_\Sigma$, $r \colon \mathrm{pos}(t) \to Q$, and for every $\rho \in \mathrm{pos}(t)$ we have $r(\rho) \in Q^{(s_0)}$, if $t(\rho) \in \Sigma^{(s_0, s_1 \dots s_k)}$, i.e., the sorts of states and terminals at the same position match. We say that *r is a run of $\mathcal{M}$ on t*.

Definition 2 (semantics of wsta)**.** Let $\mathcal{M} = (Q, \Sigma, I, \Delta)$ be an $\Re$-S-wsta and $t \in \mathrm{T}_\Sigma$. The *weight of t under $\mathcal{M}$* is defined by $[\![\mathcal{M}]\!](t) = \sum_{r \in \mathrm{run}_\mathcal{M}(t)} [\![\mathcal{M}]\!](t, r)$ where

$$[\![\mathcal{M}]\!](t, r) = I(r(\varepsilon)) \cdot \prod_{\rho \in \mathrm{pos}(t)} \Delta^{(\sigma)}(r(\rho), r(\rho 1), \dots, r(\rho k))$$

$$\text{and } \sigma = t(\rho) \text{ and } k = \mathrm{rk}(\sigma). \quad \square$$

Note: If S is a singleton set, then S-wsta are equivalent to weighted tree automata over ranked alphabets (Fülöp and Vogler, 2009). The sorts just add syntactic restrictions that will help us in the following.

Definition 3 (wfsa)**.** Let $\Re$ be a commutative semiring. An *$\Re$-weighted finite state automaton ($\Re$-wfsa)* is a tuple (P, Σ, J, Π, F) where

- P is an alphabet (of *states*),
- Σ is an alphabet (of *terminals*),
- $J \colon P \to \Re$ is a mapping (*initial weights*),
- $\Pi \colon P \times \Sigma \times P \to \Re$ is a mapping (*transition weights*), and
- $F \colon P \to \Re$ is a mapping (*final weights*). □

By $\mathrm{wfsa}(\Re, \Sigma)$ we denote the set of all $\Re$-wfsa with terminal alphabet Σ. Let $\mathcal{N} = (P, \Sigma, J, \Pi, F)$ be an $\Re$-wfsa. The *size of* $\mathcal{N}$ is defined by $\mathrm{size}(\mathcal{N}) = \mathrm{supp}(J) + \mathrm{supp}(\Pi) + \mathrm{supp}(F)$. We define the relation $\mathrm{run}_\mathcal{N}$ that contains (w, r), if $w \in \Sigma^*$ and $r \colon \{0, \dots, |w|\} \to P$. We say that *$r$ is a run of $\mathcal{N}$ on t*.

Definition 4 (semantics of wfsa)**.** Let $\mathcal{N} = (P, \Sigma, J, \Pi, F)$ be an $\Re$-wfsa and $w =$

$w_1 \ldots w_k \in \Sigma^*$ for some $k \in \mathbb{N}$. The *weight of w under $\mathcal{N}$* is defined by $[\![\mathcal{N}]\!](w) = \sum_{r \in \mathrm{run}_{\mathcal{N}}(w)} [\![\mathcal{N}]\!](w, r)$ where

$$[\![\mathcal{N}]\!](w,r) = J(r(0)) \cdot \Big(\prod_{i \in \{1,\ldots,|w|\}} \Pi(r(i-1), w_i, r(i))\Big) \cdot F(r(|w|)). \quad \square$$

Definition 5 (wuta)**.** Let $\mathfrak{R}$ be a commutative semiring. An *$\mathfrak{R}$-weighted unranked tree automaton ($\mathfrak{R}$-wuta)* is a tuple (Q, Σ, I, Δ) where

- Q is an alphabet (of *states*),
- Σ is an alphabet (of *terminals*),
- $I\colon Q \to \mathfrak{R}$ is a mapping (*root weights*), and
- $\Delta\colon Q \times \Sigma \to \mathrm{wfsa}(\mathfrak{R}, Q)$ is a mapping. $\square$

Let $\mathcal{M} = (Q, \Sigma, I, \Delta)$ be an $\mathfrak{R}$-wuta. The *size of $\mathcal{M}$* is defined by $\mathrm{size}(\mathcal{M}) = \mathrm{supp}(I) + \sum_{q \in Q, \sigma \in \Sigma} \mathrm{size}(\Delta(q,\sigma))$. The *number of states of $\mathcal{M}$* is defined as $|Q|$ plus the numbers of states of all wfsa in the image of Δ. We define the relation $\mathrm{run}_{\mathcal{M}}$ that contains (t, r), if $t \in \mathrm{U}_\Sigma$ and $r\colon \mathrm{pos}(t) \to Q$. We say that *$r$ is a run of $\mathcal{M}$ on t*.

Definition 6 (semantics of wuta)**.** Let $\mathcal{M} = (Q, \Sigma, I, \Delta)$ be an $\mathfrak{R}$-wuta. Let $t \in \mathrm{U}_\Sigma$. The *weight of t under $\mathcal{M}$* is defined by $[\![\mathcal{M}]\!](t) = \sum_{r \in \mathrm{run}_{\mathcal{M}}(t)} [\![\mathcal{M}]\!](t, r)$ where

$$[\![\mathcal{M}]\!](t,r) = I(r(\varepsilon)) \cdot \prod_{\rho \in \mathrm{pos}(t)} [\![\Delta(r(\rho), \sigma)]\!](r(\rho 1) \ldots r(\rho k))$$

$$\text{and } \sigma = t(\rho) \text{ and } k = \mathrm{rk}(t|_\rho). \quad \square$$

By exploiting distributivity it is easy to find the following equivalent formulation. Let $P_{\mathcal{M}}$ be the set of all states of all wfsa in the image of Δ. We define the relation $\text{ex-run}_{\mathcal{M}}$ that contains (t, e), if $t \in \mathrm{U}_\Sigma$ and $e = (r, s)$ where $r \in \mathrm{run}_{\mathcal{M}}(t)$ and $s\colon \mathrm{pos}(t) \to \bigcup_{n \in \mathbb{N}} {P_{\mathcal{M}}}^{\{0,\ldots,n\}}$ such that $s(\rho) \in \mathrm{run}_{\Delta(r(\rho), t(\rho))}(r(\rho 1) \ldots r(\rho\, \mathrm{rk}(t|_\rho)))$ for every $\rho \in \mathrm{pos}(t)$. We say that *e is an extended run of $\mathcal{M}$ on t*. We have $[\![\mathcal{M}]\!](t) = \sum_{e \in \text{ex-run}_{\mathcal{M}}(t)} [\![\mathcal{M}]\!](t, e)$ where

$$[\![\mathcal{M}]\!](t,(r,s)) = I(r(\varepsilon)) \cdot \prod_{\rho \in \mathrm{pos}(t)} J_\rho(s(\rho)(0)) \cdot \Big(\prod_{i=1}^{\mathrm{rk}(t|_\rho)} \Pi_\rho\big(s(\rho)(i-1), r(\rho i), s(\rho)(i)\big)\Big) \cdot F_\rho\big(s(\rho)(\mathrm{rk}(t|_\rho))\big)\Big)$$

and $(P_\rho, Q, J_\rho, \Pi_\rho, F_\rho) = \Delta(r(\rho), t(\rho))$. A similar result was stated by Droste and Vogler (2011, Def. 6.7 and Obs. 6.8).

3 Equivalences via Binarizations

In this section we will present three different surjective mappings $h\colon \mathrm{T}_\Gamma \to \mathrm{U}_\Sigma$ where Σ is an alphabet and Γ is a sorted alphabet with the maximum rank of a symbol being 2. We call h a *binarization* and we *binarize* a tree by using h backwards. Note that this allows several representations for a single unranked tree. Occasionally we also say that *t' is a binarization of t* if $t' \in \mathrm{T}_\Gamma$ and $t = h(t')$.

We show that wsta together with any of the presented binarizations and wuta are equally powerful; more formally for every wuta $\mathcal{M}$ there is a wsta $\mathcal{M}'$ and vice versa such that $[\![\mathcal{M}]\!](t) = \sum_{t' \in h^{-1}(t)} [\![\mathcal{M}']\!](t')$ for every $t \in \mathrm{T}_\Sigma$.

3.1 Left-branching Binarization

Our first binarization is inspired by the LEFT binarization of Matsuzaki et al. (2005, Fig. 6). It is similar to first-child-next-sibling encoding (Comon et al., 2007, Sec. 8.3.1). It transforms an unranked branch into a sequence of branches growing rightwards (cf. Figure 1).

Let Σ be an alphabet and let $S = \{\mathrm{T}, \mathrm{H}\}$ be a set of sorts. Intuitively, T will be the sort for trees and H will be the sort for hedges (sequences of trees). Based on Σ and assuming $\textsc{Cons}, \textsc{Null} \notin \Sigma$, we define the $(S \times S^*)$-sorted alphabet Γ by $\Gamma^{(\mathrm{T},\mathrm{H})} = \Sigma$, $\Gamma^{(\mathrm{H},\mathrm{TH})} = \{\textsc{Cons}\}$, $\Gamma^{(\mathrm{H},\varepsilon)} = \{\textsc{Null}\}$.

There is a unique homomorphism h from the S-sorted term algebra over Γ into the S-sorted algebra $((A^{(s)} \mid s \in S), (\theta_\sigma \mid \sigma \in \Gamma))$ where $A^{(\mathrm{T})} = \mathrm{U}_\Sigma$, $A^{(\mathrm{H})} = (\mathrm{U}_\Sigma)^*$,

$$\forall \sigma \in \Sigma\colon \theta_\sigma(t_1 \ldots t_k) = \sigma(t_1, \ldots, t_k),$$
$$\theta_{\textsc{Cons}}(t_0, t_1 \ldots t_k) = t_0 t_1 \ldots t_k,$$
$$\theta_{\textsc{Null}}() = \varepsilon.$$

Figure 1 (ignoring the states for now) illustrates h. Note that h is a bijection, where $h(\xi')(\rho) = \xi'(12^{\rho_1 - 1}1 \cdots 12^{\rho_n - 1}1)$ for every $\xi' \in \mathrm{T}_\Gamma^{(\mathrm{T})}$ and $\rho = \rho_1 \cdots \rho_n \in \mathrm{pos}(h(\xi'))$.

Now let $\mathcal{M}' = (Q', \Gamma, I', \Delta')$ be a T-rooted $\mathfrak{R}$-S-wsta and $\mathcal{M} = (Q, \Sigma, I, \Delta)$ be an $\mathfrak{R}$-wuta such that Q and the state sets of the wfsa defined by Δ are pairwise disjoint. We say that *$\mathcal{M}$ and $\mathcal{M}'$ are*

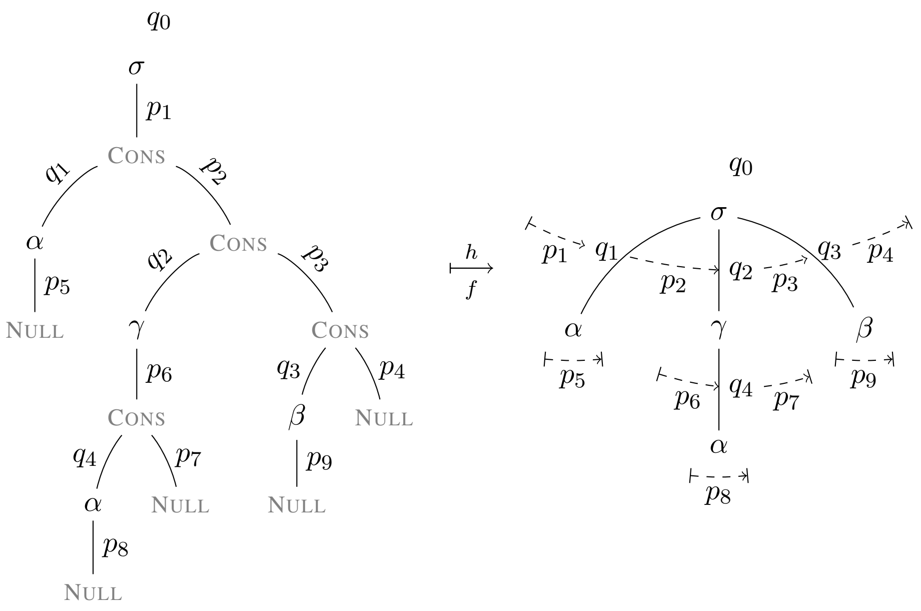

Figure 1: Trees with runs of related wsta and wuta.

$$
\begin{aligned}
&[\![\mathcal{M}]\!](t,e)\\
&= I(r(\varepsilon)) \cdot \prod_{\rho\in\mathrm{pos}(t)} J_\rho(s(\rho)(0))\\
&\qquad \cdot \Big(\prod_{i=1}^{\mathrm{rk}(t|_\rho)} \mathit{\Pi}_\rho\big(s(\rho)(i-1), r(\rho i), s(\rho)(i)\big)\Big) \cdot F_\rho\big(s(\rho)(\mathrm{rk}(t|_\rho))\big)\\
&= I'(r(\varepsilon)) \cdot \prod_{\rho\in\mathrm{pos}(t)} \mathit{\Delta}'^{(t(\rho))}(r(\rho), s(\rho)(0))\\
&\qquad \cdot \Big(\prod_{i=1}^{\mathrm{rk}(t|_\rho)} \mathit{\Delta}'^{(\textsc{Cons})}(s(\rho)(i-1), r(\rho i), s(\rho)(i))\Big) && \text{(by definition of related)}\\
&\qquad \cdot \mathit{\Delta}'^{(\textsc{Null})}(s(\rho)(\mathrm{rk}(t|_\rho))))\\
&= I'(r'(\varepsilon)) \cdot \prod_{\substack{\rho\in\mathrm{pos}(t),\\ \rho'=12^{\rho_1-1}1\cdots12^{\rho_{|\rho|}-1}1,\\ k=\mathrm{rk}(t|_\rho)}} \mathit{\Delta}'^{(t'(\rho'))}(r'(\rho'), r'(\rho'1))\\
&&& \text{(by definition of } h \text{ and } f\text{)}\\
&\qquad \cdot \Big(\prod_{i=1}^{k} \mathit{\Delta}'^{(t'(\rho'12^{i-1}))}(r'(\rho'12^{i-1}), r'(\rho'12^{i-1}1), r'(\rho'12^{i}))\Big)\\
&\qquad \cdot \mathit{\Delta}'^{(t'(\rho'12^{k}))}(r'(\rho'12^{k}))\\
&= I'(r'(\varepsilon)) \cdot \prod_{\rho\in\mathrm{pos}(t')} \mathit{\Delta}'^{(t'(\rho))}(r'(\rho), r'(\rho 1), \ldots, r'(\rho\,\mathrm{rk}(t'|_\rho)))\\
&&& \text{(by commutativity of } \cdot \text{ and definition of } h\text{)}\\
&= [\![\mathcal{M}']\!](t', r')
\end{aligned}
$$

Figure 2: Showing that $[\![\mathcal{M}]\!](t,e) = [\![\mathcal{M}']\!](t',r')$ for proof of Theorem 7, where $(t,e) = f(t',r')$, $(r,s) = e$, and $(P_\rho, Q, J_\rho, \mathit{\Pi}_\rho, F_\rho) = \mathit{\Delta}(r(\rho), t(\rho))$ for every $\rho \in \mathrm{pos}(t)$.

related, if $Q'^{(\mathrm{T})} = Q$, $Q'^{(\mathrm{H})}$ is the union of the state sets of all wfsa defined by Δ, and for every $\sigma \in \Sigma$, $q_0, q \in Q$, and $p, p' \in P$ we have

$$I'(q) = \begin{cases} I(q) & \text{if } q \in Q, \\ 0 & \text{otherwise,} \end{cases}$$
$$\Delta'^{(\sigma)}(q_0, p) = J(p),$$
$$\Delta'^{(\textsc{Cons})}(p, q, p') = \Pi(p, q, p'),$$
$$\Delta'^{(\textsc{Null})}(p) = F(p),$$

where $(P, Q, J, \Pi, F) = \Delta(q_0, \sigma)$. Note that $\mathcal{M}'$ is T-rooted.

Theorem 7. *If $\mathcal{M}$ and $\mathcal{M}'$ are related, then $[\![\mathcal{M}']\!](t) = [\![\mathcal{M}]\!](h(t))$ for every $t \in \mathrm{T}_\Gamma$.*

Proof. Assume that $\mathcal{M}$ and $\mathcal{M}'$ are related. Figure 1 shows an example tree and its image under h. Moreover it shows a run and its image under the function $f\colon \mathrm{run}_{\mathcal{M}'} \to \text{ex-run}_{\mathcal{M}}$ that is defined as follows: For every $(t', r') \in \mathrm{run}_{\mathcal{M}'}$ and $\rho \in \mathrm{pos}(h(t))$ we let $\rho' = 12^{\rho_1 - 1}1 \cdots 12^{\rho_{|\rho|}-1}1$ and define $f(t', r') = (h(t'), (r, s))$ where $r(\rho) = r'(\rho')$ and $s(\rho)(i) = r'(\rho' 12^i)$ for every $i \in \{0, \ldots, \mathrm{rk}(t|_\rho)\}$. Note that f is a bijection.

Let $(t', r') \in \mathrm{run}_{\mathcal{M}'}$ and $(t, e) = f(t', r')$. In Figure 2 we show that $[\![\mathcal{M}]\!](t, e) = [\![\mathcal{M}']\!](t', r')$. This immediately implies that $[\![\mathcal{M}]\!](t) = [\![\mathcal{M}']\!](t')$. q.e.d.

It is easy to see that a wuta and a wsta have the same size and number of states, if they are related.

3.2 Right-branching Binarization

Our second binarization is based on the RIGHT binarization of Matsuzaki et al. (2005, Fig. 6).

In comparison to left-branching binarization we make the following changes. We define Γ by $\Gamma^{(\mathrm{T,H})} = \Sigma$, $\Gamma^{(\mathrm{H,HT})} = \{\textsc{Snoc}\}$, and $\Gamma^{(\mathrm{H},\varepsilon)} = \{\textsc{Null}\}$. To define h we replace the definition of $\theta_{\textsc{Cons}}$ by $\theta_{\textsc{Snoc}}(t_1 \ldots t_k, t_{k+1}) = t_1 \ldots t_k t_{k+1}$. In the definition for *related*, we just replace $\textsc{Cons}$ by $\textsc{Snoc}$ and reverse the wfsa, i.e. we interchange J and F and swap the states in transitions. Theorem 7 still holds with these changes; the proof works analogously.

3.3 Mixed Binarization

We now have a look at a binarization where the direction of growth may be flipped at arbitrary positions from rightwards to leftwards; cf. CENTER-PARENT and CENTER-HEAD binarization of Matsuzaki et al. (2005, Fig. 6). For this purpose, let $S = \{\mathrm{T}, \mathrm{H}, \overline{\mathrm{H}}\}$. Based on Σ and assuming $\textsc{Flip}, \textsc{Cons}, \textsc{Null}, \textsc{Snoc}, \overline{\textsc{Null}} \notin \Sigma$, we define the $(S \times S^*)$-sorted alphabet Γ by

$$\Gamma^{(\mathrm{T,H})} = \Sigma, \qquad \Gamma^{(\mathrm{H},\overline{\mathrm{H}}\mathrm{T})} = \{\textsc{Flip}\},$$
$$\Gamma^{(\mathrm{H},\varepsilon)} = \{\textsc{Null}\}, \qquad \Gamma^{(\mathrm{H,TH})} = \{\textsc{Cons}\},$$
$$\Gamma^{(\overline{\mathrm{H}},\varepsilon)} = \{\overline{\textsc{Null}}\}, \qquad \Gamma^{(\overline{\mathrm{H}},\overline{\mathrm{H}}\mathrm{T})} = \{\textsc{Snoc}\}.$$

There is a unique homomorphism h from the S-sorted term algebra over Γ into the S-sorted algebra $((A^{(s)} \mid s \in S), (\theta_\sigma \mid \sigma \in \Gamma))$ where $A^{(\mathrm{T})} = \mathrm{U}_\Sigma$, $A^{(\mathrm{H})} = A^{(\overline{\mathrm{H}})} = (\mathrm{U}_\Sigma)^*$, and

$$\forall \sigma \in \Sigma\colon \theta_\sigma(t_1 \ldots t_k) = \sigma(t_1, \ldots, t_k),$$
$$\theta_{\textsc{Cons}}(t_0, t_1 \ldots t_k) = t_0 t_1 \ldots t_k,$$
$$\theta_{\textsc{Flip}}(t_1 \ldots t_k, t_{k+1}) = t_1 \ldots t_k t_{k+1},$$
$$\theta_{\textsc{Snoc}}(t_1 \ldots t_k, t_{k+1}) = t_1 \ldots t_k t_{k+1},$$
$$\theta_{\textsc{Null}}() = \theta_{\overline{\textsc{Null}}}() = \varepsilon.$$

Unfortunately, this homomorphism is just surjective, but not bijective. That means, there may be several possible binarizations of an unranked tree. Given a wsta $\mathcal{M}'$ we will construct a wuta $\mathcal{M}$, such that the weight of an unranked tree t under $\mathcal{M}$ is the sum of weights of all binarizations $h^{-1}(t)$ under $\mathcal{M}'$.

Figure 4 shows an unranked node with the three subtrees t_1, t_2, t_3 and Figure 3 shows one possible binarization with the binarized subtrees t'_1, t'_2, t'_3. Note that in Figure 3 the rightmost subtree t'_3 is attached to the node labeled $\textsc{Flip}$, yet this node is located in the middle of the tree. Therefore, if we follow the path from the root to the leaf labeled $\overline{\textsc{Null}}$, we find t'_1, t'_3, and t'_2 in this order. For the indicated run, we find the states in the order p_0, p_1, p_3, p_2.

Conversely, in the unranked case we find the subtrees in the order t_1, t_2, t_3. This is indicated by the arrow in Figure 3. Therefore, in a run of a wfsa from the constructed wuta, we have to pass along the information that we visited the state p_1, because we need it at the rightmost subtree t_3. Additionally, since each transition of the wfsa deals with one subtree, but the $\overline{\textsc{Null}}$ node has no subtrees, we have to guess a child and pass on this guess in the state. The constructed run is depicted in Figure 4.

Construction 8. For this construction, we use Σ, Γ, and S as defined above.

Let $\mathcal{M}' = (Q', \Gamma, I', \Delta')$ be a T-rooted $\mathfrak{R}$-S-wsta. We construct the $\mathfrak{R}$-wuta $\mathcal{M} =$

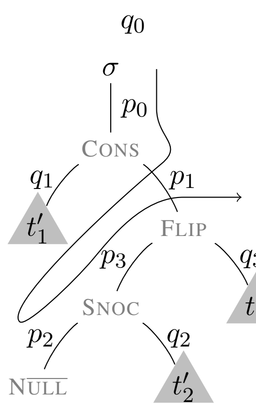

Figure 3: A binarization of Figure 4. The arrow indicates the processing order of the constructed wuta (cf. Construction 8). Note that the arrow touches p_1 twice.

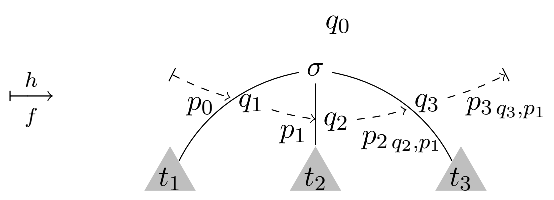

Figure 4: Unranked node with three subtrees.

$(Q'^{(\mathrm{T})}, \Sigma, I, \Delta)$ where, for every $q_0 \in Q'^{(\mathrm{T})}$ and $\sigma \in \Sigma$, we set $I(q_0) = I'(q_0)$ and $\Delta(q_0, \sigma) = (P, Q'^{(\mathrm{T})}, J, \Pi, F)$ where $P = Q'^{(\mathrm{H})} \cup \{\bar{p}_{q,s} \mid \bar{p} \in Q'^{(\overline{\mathrm{H}})}, q \in Q'^{(\mathrm{T})}, s \in Q'^{(\mathrm{H})}\}$, and for every $p, r, s \in Q'^{(\mathrm{H})}$, $\bar{p}, \bar{r} \in Q'^{(\overline{\mathrm{H}})}$, and $q, q' \in Q'^{(\mathrm{T})}$ we set

$$\begin{aligned}
J(p) &= \Delta'^{(\sigma)}(q_0, p),\\
\Pi(p, q, r) &= \Delta'^{(\textsc{Cons})}(p, q, r),\\
\Pi(s, q, \bar{r}_{q,s}) &= \Delta'^{(\overline{\textsc{Null}})}(\bar{r}),\\
\Pi(\bar{p}_{q',s}, q, \bar{r}_{q,s}) &= \Delta'^{(\textsc{Snoc})}(\bar{r}, \bar{p}, q'),\\
F(\bar{p}_{q,s}) &= \Delta'^{(\textsc{Flip})}(s, \bar{p}, q),\\
F(p) &= \Delta'^{(\textsc{Null})}(p).
\end{aligned}$$

Every other weight is set to 0. □

Theorem 8. *For $\mathcal{M}$ and $\mathcal{M}'$ from Construction 8, we have $[\![\mathcal{M}]\!](t) = \sum_{t' \in h^{-1}(t)} [\![\mathcal{M}']\!](t')$ for every $t \in \mathrm{T}_\Sigma$.*

Proof. Analogously to the proof of Theorem 7, we define a function $f\colon \mathrm{run}_{\mathcal{M}'} \to \text{ex-run}_{\mathcal{M}}$. Figures 3 and 4 visualize this mapping. Note in contrast to h that f is injective.

By construction of $\mathcal{M}$, we have that $[\![\mathcal{M}']\!](t', r') = [\![\mathcal{M}]\!](f(t', r'))$ for every $t' \in \mathrm{T}_\Gamma$ and $r' \in \mathrm{run}_{\mathcal{M}'}(t')$. For every (t, e) not in the image of f we have $[\![\mathcal{M}]\!](t, e) = 0$.

All in all we have

$$\begin{aligned}
[\![\mathcal{M}]\!](t) &= \sum_{e \in \text{ex-run}_{\mathcal{M}}(t)} [\![\mathcal{M}]\!](t, e)\\
&= \sum_{e \in \text{ex-run}_{\mathcal{M}}(t) \cap \mathrm{im}(f)} [\![\mathcal{M}]\!](t, e)\\
&= \sum_{e \in \text{ex-run}_{\mathcal{M}}(t) \cap \mathrm{im}(f)} [\![\mathcal{M}']\!](f^{-1}(t, e))\\
&= \sum_{t' \in h^{-1}(t)} \sum_{r' \in \mathrm{run}_{\mathcal{M}'}(t')} [\![\mathcal{M}']\!](t', r')\\
&= \sum_{t' \in h^{-1}(t)} [\![\mathcal{M}']\!](t'). \qquad \text{q.e.d.}
\end{aligned}$$

In Construction 8, the number of states of a single wfsa of $\mathcal{M}$ is in $\mathcal{O}(|Q'|^3)$ and its size is in $\mathcal{O}(|Q'|^2 \cdot \mathrm{size}(\mathcal{M}'))$. Since there is a wfsa for every state and terminal of $\mathcal{M}$, the number of states of $\mathcal{M}$ is in $\mathcal{O}(|Q'|^4 \cdot |\Sigma|)$ and its size is in $\mathcal{O}(|Q'|^3 \cdot |\Sigma| \cdot \mathrm{size}(\mathcal{M}'))$. Note that Π and F are the same for every constructed wfsa, so it might be beneficial to share them in an implementation.

For the other direction, i.e. when constructing a wsta given a wuta such that Theorem 8 holds, note that trees resulting from left-branching and right-branching binarization may also result from mixed binarization (modulo different node labels). Therefore the results from the previous sections can be applied.

4 The Probabilistic Case

For this section, we consider the semiring of nonnegative reals with addition and multiplication. Note that every element different from 0 has an inverse with respect to multiplication. We will use this fact later on. Since our semiring is fixed, we will not mention it anymore in this section.

If we only consider weights between 0 and 1, and some additional conditions are met, these weights can be intuitively interpreted as probabilities. With this idea in mind we start with investigating wfsa since they form the core of wuta.

4.1 Probabilistic Automata

A wfsa $\mathcal{N} = (P, \Sigma, J, \Pi, F)$ is called

- *out-probabilistic*, if for every $p \in P$ we have $F(p) + \sum_{\sigma \in \Sigma, p' \in P} \Pi(p, \sigma, p') = 1$,
- *semi-probabilistic*, if it is out-probabilistic and $\sum_{p \in P} J(p) = 1$,
- *convergent*, if $\sum_{w \in \Sigma^*} [\![\mathcal{N}]\!](w)$ is finite,
- *consistent*, if this sum is 1,
- *probabilistic*, if it is semi-probabilistic and consistent, and
- *reduced*, if for every state $p \in P$ there is a word $w \in \Sigma^*$, a run $r \in \mathrm{run}_{\mathcal{N}}(w)$, and an index $i \in \{0, \ldots, |w|\}$ such that $[\![\mathcal{N}]\!](w, r) > 0$ and $r(i) = p$.

These notions are strongly influenced by Dupont et al. (2005). If a semi-probabilistic wfsa is reduced, then it is consistent and, hence, probabilistic (Dupont et al., 2005, cf. Def. 9 and Prop. 2). Note that you can construct an equivalent reduced wfsa from any wfsa by removing those states p that violate the above condition. These states can be effectively determined.

Related Work For the remainder of this section we investigate a special case of *renormalization* of weighted or of probabilistic context-free grammars (Abney et al., 1999; Chi, 1999; Nederhof and Satta, 2003). We restrict our investigations to wfsa and give alternative proofs.

In the following, we give an alternative view on wfsa by using matrices. Let A be a matrix. The matrix entry in the i-th row and the j-th column is denoted by $(A)_{i,j}$. If A has just a single row or column, we write $(A)_i$ for the entry in the i-th column or row, respectively. A matrix with just a single entry is identified with this entry. A *diagonal matrix* is a matrix that has non-zero entries only on its diagonal. For some vector X, by $\mathrm{diag}(X)$ we denote the diagonal matrix that has the entries of X on its diagonal. The *identity matrix*, denoted by Id, is $\mathrm{diag}(1 \ \ldots \ 1)$; the dimensions of Id will always be clear from the context.

Let (P, Σ, J, Π, F) be a wfsa. We may assume w.l.o.g. that $P = \{1, \ldots, |P|\}$. We will interpret J as a $1 \times |P|$ matrix, F as a $|P| \times 1$ matrix, and we will write $\Pi^{(\sigma)}$ for the $|P| \times |P|$ matrix defined as $(\Pi^{(\sigma)})_{p,p'} = \Pi(p, \sigma, p')$ for every $\sigma \in \Sigma$, $p, p' \in P$.

Observation 9. *Let $\mathcal{N} = (P, \Sigma, J, \Pi, F)$ be a wfsa and $w = w_1 \ldots w_k \in \Sigma^*$. The weight of w under $\mathcal{N}$ can be alternatively calculated by*

$$[\![\mathcal{N}]\!](w) = J \cdot \Big(\prod_{i=1}^{|w|} \Pi^{(w_i)}\Big) \cdot F.$$

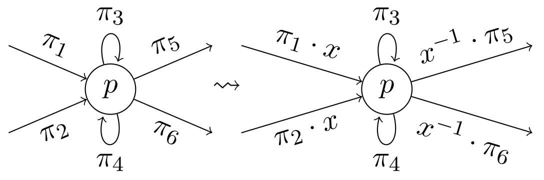

Figure 5: Changing weights $\pi_1, \ldots, \pi_6$ at state p of a wfsa with a positive real x.

The idea of the next lemma is to locally change weights of a wfsa without changing its semantics. For this purpose, the weights of "incoming" transitions (including initial weights) of some state p are scaled by some factor x while the weights of "outgoing" transitions (including final weights) of p are scaled by x^{-1}. Weights of transitions from p to p itself do not change. Figure 5 visualizes this idea. Lemma 10 applies this idea to all states simultaneously.

Lemma 10. *Let $\mathcal{N} = (P, \Sigma, J, \Pi, F)$ be a wfsa and let $X \in (\mathbb{R}_{>0})^{|P|}$. Construct the wfsa*

$$\mathcal{N}' = (P, \Sigma, J \cdot \mathrm{diag}(X), \Pi', \mathrm{diag}(X)^{-1} \cdot F)$$

with $\Pi'^{(\sigma)} = \mathrm{diag}(X)^{-1} \cdot \Pi^{(\sigma)} \cdot \mathrm{diag}(X)$ for every $\sigma \in \Sigma$. The wfsa $\mathcal{N}$ and $\mathcal{N}'$ are equivalent.

Proof. Let $w \in \Sigma^*$. If we calculate $[\![\mathcal{N}']\!](w)$ as presented in Observation 9, it is easy to see that for every factor $\mathrm{diag}(X)$ there is the adjacent factor $\mathrm{diag}(X)^{-1}$ and vice versa. q.e.d.

Construction 11. Let $\mathcal{N} = (P, \Sigma, J, \Pi, F)$ be a convergent and w.l.o.g. reduced wfsa, and let $A = \sum_{\sigma \in \Sigma} \Pi^{(\sigma)}$. Then $\mathrm{Id} - A$ is invertible and an out-probabilistic wfsa $\mathcal{N}'$ equivalent to $\mathcal{N}$ can be constructed by applying Lemma 10 to $\mathcal{N}$ with $X = (\mathrm{Id} - A)^{-1} \cdot F$. □

Theorem 11. *For every convergent wfsa there is an equivalent out-probabilistic wfsa.*

Proof. Note that $\sum_{w \in \Sigma^*} [\![\mathcal{N}]\!](w) = \sum_{j \in \mathbb{N}} J \cdot A^j \cdot F$. Hence, for every $p, p' \in P$ and $i, k \in \mathbb{N}$ we have

$$\begin{aligned}\infty &> \sum_{j\in\mathbb{N}} J\cdot A^j\cdot F \qquad \text{(by convergence)}\\ &\geq \sum_{j\geq i+k} J\cdot A^j\cdot F = \sum_{j\in\mathbb{N}} J\cdot A^i\cdot A^j\cdot A^k\cdot F\\ &\geq \sum_{j\in\mathbb{N}} (J\cdot A^i)_p\cdot (A^j)_{p,p'}\cdot (A^k\cdot F)_{p'}\\ &= (J\cdot A^i)_p\cdot\Big(\sum_{j\in\mathbb{N}} (A^j)_{p,p'}\Big)\cdot (A^k\cdot F)_{p'}.\end{aligned}$$

Since $\mathcal{N}$ is reduced, there are $i,k\in\mathbb{N}$ such that $(J\cdot A^i)_p > 0$ and $(A^k\cdot F)_{p'} > 0$; therefore $\sum_{j\in\mathbb{N}}(A^j)_{p,p'} < \infty$ for every $p,p'\in P$. Hence, $\sum_{j\in\mathbb{N}} A^j$ is a converging Neumann series. This implies that the inverse of $\mathrm{Id}-A$ exists and is equal to this sum; this seems to be a well known result in the field of functional analysis; e.g. cf. Heuser (2006, Thm. 12.4).

Now we need a vector X to apply Lemma 10 such that the resulting wfsa is out-probabilistic, i.e., $\mathrm{diag}(X)^{-1}\cdot A\cdot\mathrm{diag}(X)\cdot(1\ \dots\ 1)^{\mathrm{T}} + \mathrm{diag}(X)^{-1}\cdot F = (1\ \dots\ 1)^{\mathrm{T}}$. This equation can easily be transformed into $(\mathrm{Id}-A)\cdot X = F$. Since $\mathrm{Id}-A$ is invertible, the equation is solved by $X = (\mathrm{Id}-A)^{-1}\cdot F$.

It remains to be shown that every entry of X is strictly positive. Recall that every entry of A is non-negative and that for every $p\in P$ there is a $j\in\mathbb{N}$ such that $(A^j\cdot F)_p > 0$. This implies that every entry of $(\mathrm{Id}-A)^{-1} = \sum_{j\in\mathbb{N}} A^j$ is non-negative and that every entry of $X = (\sum_{j\in\mathbb{N}} A^j)\cdot F$ is strictly positive. q.e.d.

Theorem 12. *For every consistent wfsa there is an equivalent probabilistic wfsa.*

Proof. Since consistency implies convergence we can apply Construction 11 and obtain $\mathcal{N}' = (P,\Sigma,J',\Pi',F')$. By Lemma 10 $\mathcal{N}'$ is consistent. It remains to be shown that $\mathcal{N}'$ is also semi-probabilistic. By Construction 11 $\mathcal{N}'$ is already out-probabilistic, so we just have to show that $\sum_{p\in P} J'(p) = 1$.

For $p\in P$, let $\mathcal{N}_p = (P,\Sigma,J_p,\Pi',F')$ where J_p is the $1\times|P|$ matrix where $(J_p)_p = 1$ and every other entry is 0. Obviously $\mathcal{N}_p$ is semi-probabilistic. Let $\mathcal{N}'_p$ be the equivalent reduced wfsa that is created from $\mathcal{N}_p$ just by removing states. It is easy to see that $\mathcal{N}'_p$ is also semi-probabilistic. Since $\mathcal{N}'_p$ is semi-probabilistic and reduced, it is also consistent, hence, by equivalency, also $\mathcal{N}_p$ is consistent. That means

$$\begin{aligned}1 = \sum_{w\in\Sigma^*}[\![\mathcal{N}_p]\!](w) &= \sum_{w\in\Sigma^*} J_p\cdot\Big(\prod_{i=1}^{|w|}\Pi'^{(w_i)}\Big)\cdot F'\\ &= \sum_{w\in\Sigma^*}\Big(\big(\prod_{i=1}^{|w|}\Pi'^{(w_i)}\big)\cdot F'\Big)_p.\end{aligned}$$

Since p was chosen arbitrarily, we have $1 = \sum_{w\in\Sigma^*}[\![\mathcal{N}']\!](w) = J'\cdot(1\ \dots\ 1)^{\mathrm{T}}$. q.e.d.

Corollary 13. *The class of weighted languages recognizable by probabilistic wfsa is closed under reversal.*

Proof. A wfsa can easily be reversed by transposing the transition matrices and interchanging initial and final weights. The corollary follows by Theorem 12. q.e.d.

Related Work. Paz (1971, Chapter III, Section A, Theorem 1.8) presents the same result for his probabilistic automata, which are slightly different from our probabilistic wfsa. His construction requires an exponential number of states in comparison to the given automaton. Our definition of probabilistic wfsa allows the presented construction, which does not change the state set at all. Paz' construction can be easily adapted to our case, yet it is unclear if our construction can also be adapted to his case.

4.2 Probabilities and Tree Automata

The notions probabilistic, semi-probabilistic, convergent, consistent, and reduced can be easily generalized to wsta and wuta. Note that in the tree case semi-probabilistic and reduced do not generally imply consistent. We now investigate what happens to these properties when constructing a wuta given a wsta or vice versa as presented in Section 3.

If the input is reduced, so is the output, except in case of mixed binarization when going from wsta to wuta; yet the output can easily be reduced. In case of left-branching binarization, if the input is semi-probabilistic, the output is also semi-probabilistic. Note that in any case, if the input is semi-probabilistic and reduced, then the wfsa in the wuta are consistent. Hence, in case of right-branching binarization, if the input is semi-probabilistic and reduced, the output is reduced and by Theorem 12 we can find an equivalent automaton that is also semi-probabilistic. In case of mixed binarization the direction from wuta to wsta is subsumed by the previous cases. Starting with a semi-probabilistic and reduced wsta, we can apply Construction 8, reduce the wfsa in the wuta without breaking consistency, and apply Theorem 12, ending up with a semi-probabilistic and reduced wuta.

Theorems 7 and 8 imply that if the input is convergent or even consistent, so is the output. Hence in the previous paragraph we may replace semi-probabilistic by probabilistic and the statements still hold.

5 Outlook: Implications on Training Methods

For each of three binarizations we have shown that wsta together with a binarization are equally powerful to wuta.

Our results suggest training methods for wuta: By binarizing a wuta, training the resulting wsta, and undoing the binarization, it is possible to use training algorithms designed for wsta also on wuta. The training may even alter the state behaviour, e.g. by splitting or merging states; cf. e.g. Matsuzaki et al. (2005) and Petrov et al. (2006). *Merging* (for example two) states q_1 and q_2 to a new state q means to replace every occurrence of q_1 and q_2 by q; the state set, the initial weights and the transition weights have to be adapted appropriately. The opposite direction is the *splitting* of a state q into (for example two) new states q_1 and q_2, which means to replace every occurrence of q by q_1 or q_2 in every possible combination, e.g., if there is a transition with two occurrences of q before splitting, then there are four transitions with different occurrences of q_1 and q_2 after splitting. Note that q, q_1, and q_2 need to have the same sort, otherwise there would be incompatibilities with the sorts of the (unchanged) terminals after splitting or merging.

These results formally explain why the performance of the training by Matsuzaki et al. (2005) is rather independent from the used binarization. Note that they used probabilistic context-free grammars with latent annotations (pcfg-la) while we used wsta, but it is easy to see that both formalisms are equally powerful. Additionally our binarizations use different node labels and introduced additional unary nodes as well as NULL and $\overline{\text{NULL}}$; but again this does not change the power of the formalism. Note that while changing latent annotations for pcfg-la it is not necessary to deal with sorts, because a latent annotation is always considered together with the terminal[1] it is attached to.

References

Steven Abney, David McAllester, and Fernando Pereira. 1999. Relating probabilistic grammars and automata. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, page 542–549, Stroudsburg, PA, USA. Association for Computational Linguistics. DOI: `10.3115/1034678.1034759`.

Julien Carme, Joachim Niehren, and Marc Tommasi. 2004. Querying unranked trees with stepwise tree automata. In Vincent van Oostrom, editor, *Rewriting Techniques and Applications*, volume 3091 of *Lecture Notes in Computer Science*, page 105–118. Springer Berlin Heidelberg. DOI: `10.1007/978-3-540-25979-4_8`.

Zhiyi Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160, March. URL: `http://dl.acm.org/citation.cfm?id=973215.973219`.

Hubert Comon, Max Dauchet, Remi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. 2007. Tree automata techniques and applications, October. URL: `http://tata.gforge.inria.fr/`.

Manfred Droste and Heiko Vogler. 2011. Weighted logics for unranked tree automata. *Theory of Computing Systems*, 48(1):23–47. DOI: `10.1007/s00224-009-9224-4`.

Pierre Dupont, François Denis, and Yann Esposito. 2005. Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms. *Pattern Recognition*, 38(9):1349–1371. Grammatical Inference. DOI: `10.1016/j.patcog.2004.03.020`.

Zoltán Fülöp and Heiko Vogler, 2009. *Handbook of Weighted Automata*, chapter Weighted Tree Automata and Tree Transducers, pages 313–403. Springer Berlin Heidelberg, Berlin, Heidelberg. DOI: `10.1007/978-3-642-01492-5_9`.

Harro Heuser. 2006. *Funktionalanalysis / Theorie und Anwendung*. Teubner, fourth edition. URL: `https://www.springer.com/9783835100268`.

[1]We stick to the notion of terminal as it is used in this paper. In the context of pcfg-la, nullary terminals are called terminal symbols while the other terminals are called nonterminal symbols.

Johanna Högberg, Andreas Maletti, and Heiko Vogler. 2009. Bisimulation minimisation of weighted automata on unranked trees. *Fundamenta Informaticae*, 92(1-2):103–130, January. DOI: `10.3233/FI-2009-0068`.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 75–82, Stroudsburg, PA, USA. Association for Computational Linguistics. DOI: `10.3115/1219840.1219850`.

Mark-Jan Nederhof and Giorgio Satta. 2003. Probabilistic parsing as intersection. In *8th International Workshop on Parsing Technologies*, page 137–148.

Azaria Paz. 1971. *Introduction to Probabilistic Automata*. Academic Press. URL: `http://www.sciencedirect.com/science/book/9780125476508`.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 433–440, Stroudsburg, PA, USA. Association for Computational Linguistics. DOI: `10.3115/1220175.1220230`.