Parsing Algorithms based on Tree Automata

Andreas Maletti
Departament de Filologies Romàniques
Universitat Rovira i Virgili, Tarragona, Spain
andreas.maletti@urv.cat

Giorgio Satta
Department of Information Engineering
University of Padua, Italy
satta@dei.unipd.it

Abstract

We investigate several algorithms related to the parsing problem for weighted automata, under the assumption that the input is a string rather than a tree. This assumption is motivated by several natural language processing applications. We provide algorithms for the computation of parse-forests, best tree probability, inside probability (called partition function), and prefix probability. Our algorithms are obtained by extending to weighted tree automata the Bar-Hillel technique, as defined for context-free grammars.

1 Introduction

Tree automata are finite-state devices that recognize tree languages, that is, sets of trees. There is a growing interest nowadays in the natural language parsing community, and especially in the area of syntax-based machine translation, for probabilistic tree automata (PTA) viewed as suitable representations of grammar models. In fact, probabilistic tree automata are generatively more powerful than probabilistic context-free grammars (PCFGs), when we consider the latter as devices that generate tree languages. This difference can be intuitively understood if we consider that a computation by a PTA uses hidden states, drawn from a finite set, that can be used to transfer information within the tree structure being recognized.

As an example, in written English we can empirically observe different distributions in the expansion of so-called noun phrase (NP) nodes, in the contexts of subject and direct-object positions, respectively. This can be easily captured using some states of a PTA that keep a record of the different contexts. In contrast, PCFGs are unable to model these effects, because NP node expansion should be independent of the context in the derivation. This problem for PCFGs is usually solved by resorting to so-called parental annotations (Johnson, 1998), but this, of course, results in a different tree language, since these annotations will appear in the derived tree.

Most of the theoretical work on parsing and estimation based on PTA has assumed that the input is a tree (Graehl et al., 2008), in accordance with the very definition of these devices. However, both in parsing as well as in machine translation, the input is most often represented as a string rather than a tree. When the input is a string, some trick is applied to map the problem back to the case of an input tree. As an example in the context of machine translation, assume a probabilistic tree transducer T as a translation model, and an input string w to be translated. One can then intermediately construct a tree automaton M_w that recognizes the set of all possible trees that have w as yield, with internal nodes from the input alphabet of T. This automaton M_w is further transformed into a tree transducer implementing a partial identity translation, and such a transducer is composed with T (relation composition). This is usually called the 'cascaded' approach. Such an approach can be easily applied also to parsing problems.

In contrast with the cascaded approach above, which may be rather inefficient, in this paper we investigate a more direct technique for parsing strings based on weighted and probabilistic tree automata. We do this by extending to weighted tree automata the well-known Bar-Hillel construction defined for context-free grammars (Bar-Hillel et al., 1964) and for weighted context-free grammars (Nederhof and Satta, 2003). This provides an abstract framework under which several parsing algorithms can be directly derived, based on weighted tree automata. We discuss several applications of our results, including algorithms for the computation of parse-forests, best tree probability, inside probability (called partition function), and prefix probability.

Proceedings of the 11th International Conference on Parsing Technologies (IWPT), pages 1–12,
Paris, France, October, 2009.

2 Preliminary definitions

Let S be a nonempty set and $\cdot$ be an associative binary operation on S. If S contains an element 1 such that $1 \cdot s = s = s \cdot 1$ for every $s \in S$, then $(S, \cdot, 1)$ is a monoid. A monoid $(S, \cdot, 1)$ is commutative if the equation $s_1 \cdot s_2 = s_2 \cdot s_1$ holds for every $s_1, s_2 \in S$. A **commutative semiring** $(S, +, \cdot, 0, 1)$ is a nonempty set S on which a binary addition $+$ and a binary multiplication $\cdot$ have been defined such that the following conditions are satisfied:

- $(S, +, 0)$ and $(S, \cdot, 1)$ are commutative monoids,
- $\cdot$ distributes over $+$ from both sides, and
- $s \cdot 0 = 0 = 0 \cdot s$ for every $s \in S$.

A **weighted string automaton**, abbreviated WSA, (Schützenberger, 1961; Eilenberg, 1974) is a system $M = (Q, \Sigma, \mathcal{S}, I, \nu, F)$ where

- Q is a finite alphabet of states,
- Σ is a finite alphabet of input symbols,
- $\mathcal{S} = (S, +, \cdot, 0, 1)$ is a semiring,
- $I \colon Q \to S$ assigns initial weights,
- $\nu \colon Q \times \Sigma \times Q \to S$ assigns a weight to each transition, and
- $F \colon Q \to S$ assigns final weights.

We now proceed with the semantics of M. Let $w \in \Sigma^*$ be an input string of length n. For each integer i with $1 \leq i \leq n$, we write $w(i)$ to denote the i-th character of w. The set $\mathrm{Pos}(w)$ of **positions** of w is $\{i \mid 0 \leq i \leq n\}$. A **run** of M on w is a mapping $r \colon \mathrm{Pos}(w) \to Q$. We denote the set of all such runs by $\mathrm{Run}_M(w)$. The weight of a run $r \in \mathrm{Run}_M(w)$ is

$$\mathrm{wt}_M(r) = \prod_{i=1}^{n} \nu(r(i-1), w(i), r(i)) \ .$$

We assume the right-hand side of the above equation evaluates to 1 in case $n = 0$. The WSA M recognizes the mapping $M \colon \Sigma^* \to S$, which is defined for every $w \in \Sigma^*$ of length n by[1]

$$M(w) = \sum_{r \in \mathrm{Run}_M(w)} I(r(0)) \cdot \mathrm{wt}_M(r) \cdot F(r(n)) \ .$$

In order to define weighted tree automata (Berstel and Reutenauer, 1982; Ésik and Kuich, 2003; Borchardt, 2005), we need to introduce some additional notation. Let Σ be a **ranked alphabet**, that is, an alphabet whose symbols have an associated arity. We write Σ_k to denote the set of all k-ary symbols in Σ. We use a special symbol $e \in \Sigma_0$ to syntactically represent the empty string ε. The set of Σ-**trees**, denoted by T_Σ, is the smallest set satisfying both of the following conditions

- for every $\alpha \in \Sigma_0$, the single node labeled α, written $\alpha()$, is a tree of T_Σ,
- for every $\sigma \in \Sigma_k$ with $k \geq 1$ and for every $t_1, \ldots, t_k \in T_\Sigma$, the tree with a root node labeled σ and trees $t_1, \ldots, t_k$ as its k children, written $\sigma(t_1, \ldots, t_k)$, belongs to T_Σ.

As a convention, throughout this paper we assume that $\sigma(t_1, \ldots, t_k)$ denotes $\sigma()$ if $k = 0$. The size of the tree $t \in T_\Sigma$, written $|t|$, is defined as the number of occurrences of symbols from Σ in t.

Let $t = \sigma(t_1, \ldots, t_k)$. The yield of t is recursively defined by

$$\mathrm{yd}(t) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_0 \setminus \{e\} \\ \varepsilon & \text{if } \sigma = e \\ \mathrm{yd}(t_1) \cdots \mathrm{yd}(t_k) & \text{otherwise.} \end{cases}$$

The set of **positions** of t, denoted by $\mathrm{Pos}(t)$, is recursively defined by

$$\mathrm{Pos}(\sigma(t_1, \ldots, t_k)) = \{\varepsilon\} \cup \{iw \mid 1 \leq i \leq k, w \in \mathrm{Pos}(t_i)\} \ .$$

Note that $|t| = |\mathrm{Pos}(t)|$ and, according to our convention, when $k = 0$ the above definition provides $\mathrm{Pos}(\sigma()) = \{\varepsilon\}$. We denote the symbol of t at position w by $t(w)$ and its rank by $\mathrm{rk}_t(w)$.

A **weighted tree automaton** (WTA) is a system $M = (Q, \Sigma, \mathcal{S}, \mu, F)$ where

- Q is a finite alphabet of states,
- Σ is a finite ranked alphabet of input symbols,
- $\mathcal{S} = (S, +, \cdot, 0, 1)$ is a semiring,
- μ is an indexed family $(\mu_k)_{k \in \mathbb{N}}$ of mappings $\mu_k : \Sigma_k \to S^{Q \times Q^k}$, and
- $F \colon Q \to S$ assigns final weights.

In the above definition, Q^k is the set of all strings over Q having length k, with $Q^0 = \{\varepsilon\}$. Further note that $S^{Q \times Q^k}$ is the set of all matrices with elements in S, row index set Q, and column index set Q^k. Correspondingly, we will use the common matrix notation and write instances of μ in the form $\mu_k(\sigma)_{q_0, q_1 \cdots q_k}$. Finally, we assume $q_1 \cdots q_k = \varepsilon$ if $k = 0$.

We define the semantics also in terms of runs. Let $t \in T_\Sigma$. A **run** of M on t is a mapping $r \colon \mathrm{Pos}(t) \to Q$. We denote the set of all such runs

[1] We overload the symbol M to denote both an automaton and its recognized mapping. However, the intended meaning will always be clear from the context.

by $\mathrm{Run}_M(t)$. The weight of a run $r \in \mathrm{Run}_M(t)$ is

$$\mathrm{wt}_M(r) = \prod_{\substack{w \in \mathrm{Pos}(t) \\ \mathrm{rk}_t(w)=k}} \mu_k(t(w))_{r(w),r(w1)\cdots r(wk)} \ .$$

Note that, according to our convention, the string $r(w1)\cdots r(wk)$ denotes ε when $k = 0$. The WTA M recognizes the mapping $M\colon T_\Sigma \to S$, which is defined by

$$M(t) = \sum_{r \in \mathrm{Run}_M(t)} \mathrm{wt}_M(r) \cdot F(r(\varepsilon))$$

for every $t \in T_\Sigma$. We say that t is **recognized** by M if $M(t) \neq 0$.

In our complexity analyses, we use the following measures. The size of a transition (p, α, q) in (the domain of ν in) a WSA is $|p\alpha q| = 3$. The size of a transition in a WTA, viewed as an instance $(\sigma, q_0, q_1 \cdots q_k)$ of some mapping μ_k, is defined as $|\sigma q_0 \cdots q_k|$, that is, the rank of the input symbol occurring in the transition plus two. Finally, the size $|M|$ of an automaton M (WSA or WTA) is defined as the sum of the sizes of its nonzero transitions. Note that this does not take into account the size of the representation of the weights.

3 Binarization

We introduce in this section a specific transformation of WTA, called **binarization**, that reduces the transitions of the automaton to some normal form in which no more than three states are involved. This transformation maps the set of recognized trees into a special binary form, in such a way that the yields of corresponding trees and their weights are both preserved. We use this transformation in the next section in order to guarantee the computational efficiency of the parsing algorithm we develop. The standard 'first-child, next-sibling' binary encoding for trees (Knuth, 1997) would eventually result in a transformed WTA of quadratic size. To obtain instead a linear size transformation, we introduce a slightly modified encoding (Högberg et al., 2009, Section 4), which is inspired by (Carme et al., 2004) and the classical currying operation.

Let Σ be a ranked alphabet and assume a fresh symbol $@ \notin \Sigma$ (corresponding to the basic list concatenation operator). Moreover, let $\Delta = \Delta_2 \cup \Delta_1 \cup \Delta_0$ be the ranked alphabet such that $\Delta_2 = \{@\}$, $\Delta_1 = \bigcup_{k \geq 1} \Sigma_k$, and $\Delta_0 = \Sigma_0$. In words, all the original non-nullary symbols from Σ are now unary, @ is binary, and the original nullary symbols from Σ have their rank preserved. We encode each tree of T_Σ as a tree of T_Δ as follows:

- $\mathrm{enc}(\alpha) = \alpha()$ for every $\alpha \in \Sigma_0$,
- $\mathrm{enc}(\gamma(t)) = \gamma(\mathrm{enc}(t))$ for every $\gamma \in \Sigma_1$ and $t \in T_\Sigma$, and
- for $k \geq 2$, $\sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in T_\Sigma$

$$\mathrm{enc}(\sigma(t_1, \ldots, t_k)) = \sigma(@(\mathrm{enc}(t_1), \ldots @(\mathrm{enc}(t_{k-1}), \mathrm{enc}(t_k)) \cdots)).$$

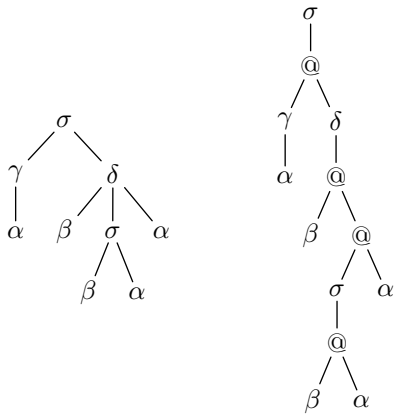

Figure 1: Input tree t and encoded tree $\mathrm{enc}(t)$.

An example of the above encoding is illustrated in Figure 1. Note that $|\mathrm{enc}(t)| \in \mathcal{O}(|t|)$ for every $t \in T_\Sigma$. Furthermore, t can be easily reconstructed from $\mathrm{enc}(t)$ in linear time.

Definition 1 Let $M = (Q, \Sigma, \mathcal{S}, \mu, F)$ be a WTA. The encoded WTA $\mathrm{enc}(M)$ is $(P, \Delta, \mathcal{S}, \mu', F')$ where

$$P = \{[q] \mid q \in Q\} \cup \cup \{[w] \mid \mu_k(\sigma)_{q,uw} \neq 0, u \in Q^*,\ w \in Q^+\},$$

$F'([q]) = F(q)$ for every $q \in Q$, and the transitions are constructed as follows:

(i) $\mu'_0(\alpha)_{[q],\varepsilon} = \mu_0(\alpha)_{q,\varepsilon}$ for every $\alpha \in \Sigma_0$,
(ii) $\mu'_1(\sigma)_{[q],[w]} = \mu_k(\sigma)_{q,w}$ for every $\sigma \in \Sigma_k$, $k \geq 1$, $q \in Q$, and $w \in Q^k$, and
(iii) $\mu'_2(@)_{[qw],[q][w]} = 1$ for every $[qw] \in P$ with $|w| \geq 1$ and $q \in Q$.

All remaining entries in F' and μ' are 0. □

Notice that each transition of $\mathrm{enc}(M)$ involves no more than three states from P. Furthermore, we have $|\mathrm{enc}(M)| \in \mathcal{O}(|M|)$. The following result is rather intuitive (Högberg et al., 2009, Lemma 4.2); its proof is therefore omitted.

Theorem 1 *Let $M = (Q, \Sigma, \mathcal{S}, \mu, F)$ be a* WTA*, and let $M' = \mathrm{enc}(M)$. Then $M(t) = M'(\mathrm{enc}(t))$ for every $t \in T_\Sigma$.* □

4 Bar-Hillel construction

The so-called Bar-Hillel construction was proposed in (Bar-Hillel et al., 1964) to show that the intersection of a context-free language and a regular language is still a context-free language. The proof of the result consisted in an effective construction of a context-free grammar $\mathrm{Prod}(G, N)$ from a context-free grammar G and a finite automaton N, such that $\mathrm{Prod}(G, N)$ generates the intersection of the languages generated by G and N.

It was later recognized that the Bar-Hillel construction constitutes one of the foundations of the theory of tabular parsing based on context-free grammars. More precisely, by taking the finite automaton N to be of some special kind, accepting only a single string, the Bar-Hillel construction provides a framework under which several well-known tabular parsing algorithms can easily be derived, that were proposed much later in the literature.

In this section we extend the Bar-Hillel construction to WTA, with a similar purpose of establishing an abstract framework under which one could easily derive parsing algorithms based on these devices. In order to guarantee computational efficiency, we avoid here stating the Bar-Hillel construction for WTA with alphabets of arbitrary rank. The next result therefore refers to WTA with alphabet symbols of rank at most 2. These may, but need not, be automata obtained through the binary encoding discussed in Section 3.

Definition 2 Let $M = (Q, \Sigma, \mathcal{S}, \mu, F)$ be a WTA such that the maximum rank of a symbol in Σ is 2, and let $N = (P, \Sigma_0 \setminus \{e\}, \mathcal{S}, I, \nu, G)$ be a WSA over the same semiring. We construct the WTA

$$\mathrm{Prod}(M, N) = (P \times Q \times P, \Sigma, \mathcal{S}, \mu', F')$$

as follows:

(i) For every $\sigma \in \Sigma_2$, states $p_0, p_1, p_2 \in P$, and states $q_0, q_1, q_2 \in Q$ let

$$\mu'_2(\sigma)_{(p_0,q_0,p_2),(p_0,q_1,p_1)(p_1,q_2,p_2)} = \mu_2(\sigma)_{q_0,q_1 q_2} \ .$$

(ii) For every symbol $\gamma \in \Sigma_1$, states $p_0, p_1 \in P$, and states $q_0, q_1 \in Q$ let

$$\mu'_1(\gamma)_{(p_0,q_0,p_1),(p_0,q_1,p_1)} = \mu_1(\gamma)_{q_0,q_1} \ .$$

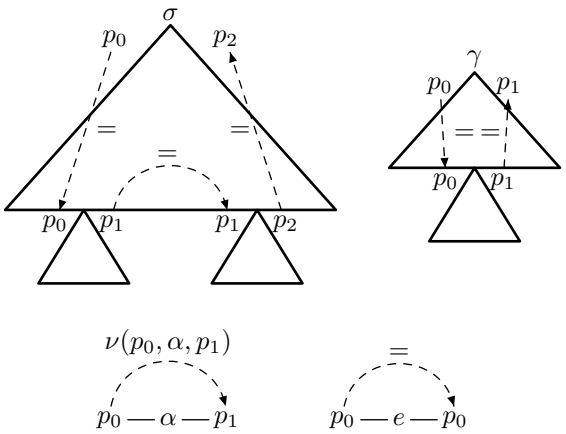

Figure 2: Information transport in the first and third components of the states in our Bar-Hillel construction.

(iii) For every symbol $\alpha \in \Sigma_0$, states $p_0, p_1 \in P$, and $q \in Q$ let

$$\mu'_0(\alpha)_{(p_0,q,p_1),\varepsilon} = \mu_0(\alpha)_{q,\varepsilon} \cdot s$$

where

$$s = \begin{cases} \nu(p_0, \alpha, p_1) & \text{if } \alpha \neq e \\ 1 & \text{if } \alpha = e \text{ and } p_0 = p_1 \end{cases} .$$

(iv) $F'(p_0, q, p_1) = I(p_0) \cdot F(q) \cdot G(p_1)$ for every $p_0, p_1 \in P$ and $q \in Q$.

All remaining entries in μ' are 0. □

Theorem 2 *Let M and N be as in Definition 2, and let $M' = \mathrm{Prod}(M, N)$. If $\mathcal{S}$ is commutative, then $M'(t) = M(t) \cdot N(\mathrm{yd}(t))$ for every $t \in T_\Sigma$.* □

PROOF For a state $q \in P \times Q \times P$, we write q_i to denote its i-th component with $i \in \{1, 2, 3\}$. Let $t \in T_\Sigma$ and $r \in \mathrm{Run}_{M'}(t)$ be a run of M' on t. We call the run r **well-formed** if for every $w \in \mathrm{Pos}(t)$:

(i) if $t(w) = e$, then $r(w)_1 = r(w)_3$,

(ii) if $t(w) \notin \Sigma_0$, then:

(a) $r(w)_1 = r(w1)_1$,

(b) $r(w\,\mathrm{rk}_t(w))_3 = r(w)_3$, and

(c) if $\mathrm{rk}_t(w) = 2$, then $r(w1)_3 = r(w2)_1$.

Note that no conditions are placed on the second components of the states in r. We try to illustrate the conditions in Figure 2.

A standard proof shows that $\mathrm{wt}_{M'}(r) = 0$ for all runs $r \in \mathrm{Run}_{M'}(t)$ that are not well-formed. We now need to map runs of M' back into 'corresponding' runs for M and N. Let us fix some $t \in T_\Sigma$ and some well-formed run $r \in \mathrm{Run}_{M'}(t)$.

We define the run $\pi_M(r) \in \mathrm{Run}_M(t)$ by letting

$$\pi_M(r)(w) = r(w)_2,$$

for every $w \in \mathrm{Pos}(t)$. Let $\{w_1, \ldots, w_n\} = \{w' \mid w' \in \mathrm{Pos}(t),\ t(w') \in \Sigma_0 \setminus \{e\}\}$, with $w_1 < \cdots < w_n$ according to the lexicographic order on $\mathrm{Pos}(t)$. We also define the run $\pi_N(r) \in \mathrm{Run}_N(\mathrm{yd}(t))$ by letting

$$\pi_N(r)(i-1) = r(w_i)_1,$$

for every $1 \leq i < n$, and

$$\pi_N(r)(n) = r(w_n)_3 \ .$$

Note that conversely every run of M on t and every run of N on $\mathrm{yd}(t)$ yield a unique run of M' on t.

Now, we claim that

$$\mathrm{wt}_{M'}(r) = \mathrm{wt}_M(\pi_M(r)) \cdot \mathrm{wt}_N(\pi_N(r))$$

for every well-formed run $r \in \mathrm{Run}_{M'}(t)$. To prove the claim, let $t = \sigma(t_1, \ldots, t_k)$ for some $\sigma \in \Sigma_k$, $k \leq 2$, and $t_1, \ldots, t_k \in T_\Sigma$. Moreover, for every $1 \leq i \leq k$ let $r_i(w) = r(iw)$ for every $w \in \mathrm{Pos}(t_i)$. Note that $r_i \in \mathrm{Run}_{M'}(t_i)$ and that r_i is well-formed for every $1 \leq i \leq k$.

For the induction base, let $\sigma \in \Sigma_0$; we can write

$$\begin{aligned}
&\mathrm{wt}_{M'}(r) \\
&= \mu'_0(\sigma)_{r(\varepsilon),\varepsilon} \\
&= \begin{cases} \mu_0(\sigma)_{r(\varepsilon)_2,\varepsilon} \cdot \nu(r(\varepsilon)_1, \sigma, r(\varepsilon)_3) & \text{if } \sigma \neq e \\ \mu_0(\sigma)_{r(\varepsilon)_2,\varepsilon} & \text{if } \sigma = e \end{cases} \\
&= \mathrm{wt}_M(\pi_M(r)) \cdot \mathrm{wt}_N(\pi_N(r)) \ .
\end{aligned}$$

In the induction step (i.e., $k > 0$) we have

$$\begin{aligned}
&\mathrm{wt}_{M'}(r) \\
&= \prod_{\substack{w \in \mathrm{Pos}(t) \\ \mathrm{rk}_t(w) = n}} \mu'_n(t(w))_{r(w), r(w1) \cdots r(wn)} \\
&= \mu'_k(\sigma)_{r(\varepsilon), r(1) \cdots r(k)} \cdot \prod_{i=1}^{k} \mathrm{wt}_{M'}(r_i) \ .
\end{aligned}$$

Using the fact that r is well-formed, commutativity, and the induction hypothesis, we obtain

$$\begin{aligned}
&= \mu_k(\sigma)_{r(\varepsilon)_2, r(1)_2 \cdots r(k)_2} \cdot \\
&\quad \cdot \prod_{i=1}^{k} \Big(\mathrm{wt}_M(\pi_M(r_i)) \cdot \mathrm{wt}_N(\pi_N(r_i)) \Big) \\
&= \mathrm{wt}_M(\pi_2(r)) \cdot \mathrm{wt}_N(\pi_N(r)) \ ,
\end{aligned}$$

where in the last step we have again used the fact that r is well-formed. Using the auxiliary statement $\mathrm{wt}_{M'}(r) = \mathrm{wt}_M(\pi_M(r)) \cdot \mathrm{wt}_N(\pi_N(r))$, the main proof now is easy.

$$\begin{aligned}
&M'(t) \\
&= \sum_{r \in \mathrm{Run}_{M'}(t)} \mathrm{wt}_{M'}(r) \cdot F'(r(\varepsilon)) \\
&= \sum_{\substack{r \in \mathrm{Run}_{M'}(t) \\ r \text{ well-formed}}} \mathrm{wt}_M(\pi_M(r)) \cdot \mathrm{wt}_N(\pi_N(r)) \cdot \\
&\qquad \cdot I(r(\varepsilon)_1) \cdot F(r(\varepsilon)_2) \cdot G(r(\varepsilon)_3) \\
&= \Big(\sum_{r \in \mathrm{Run}_M(t)} \mathrm{wt}_M(r) \cdot F(r(\varepsilon)) \Big) \cdot \\
&\qquad \cdot \Big(\sum_{\substack{w = \mathrm{yd}(t) \\ r \in \mathrm{Run}_N(w)}} I(r(0)) \cdot \mathrm{wt}_N(r) \cdot G(r(|w|)) \Big) \\
&= M(t) \cdot N(\mathrm{yd}(t))
\end{aligned}$$

■

Let us analyze now the computational complexity of a possible implementation of the construction in Definition 2. In step (i), we could restrict the computation by considering only those transitions in M satisfying $\mu_2(\sigma)_{q_0,q_1q_2} \neq 0$, which provides a number of choices in $\mathcal{O}(|M|)$. Combined with the choices for the states p_0, p_1, p_2 of N, this provides $\mathcal{O}(|M| \cdot |P|^3)$ non-zero transitions in $\mathrm{Prod}(M, N)$. This is also a bound on the overall running time of step (i). Since we additionally assume that weights can be multiplied in constant time, it is not difficult to see that all of the remaining steps can be accommodated within such a time bound. We thus conclude that the construction in Definition 2 can be implemented to run in time and space $\mathcal{O}(|M| \cdot |P|^3)$.

5 Parsing applications

In this section we discuss several applications of the construction presented in Definition 2 that are relevant for parsing based on WTA models.

5.1 Parse forest

Parsing is usually defined as the problem of constructing a suitable representation for the set of all possible parse trees that are assigned to a given input string w by some grammar model. The set of all such parse trees is called **parse forest**. The extension of the Bar-Hillel construction that we have

presented in Section 4 can be easily adapted to obtain a parsing algorithm for WTA models. This is described in what follows.

First, we should represent the input string w in a WSA that recognizes the language $\{w\}$. Such an automaton has a state set $P = \{p_0, \ldots, p_{|w|}\}$ and transition weights $\nu(p_{i-1}, w(i), p_i) = 1$ for each i with $1 \leq i \leq |w|$. We also set $I(p_0) = 1$ and $F(p_{|w|}) = 1$. Setting all the weights to 1 for a WSA N amounts to ignoring the weights, i.e., those weights will not contribute in any way when applying the Bar-Hillel construction.

Assume now that M is our grammar model, represented as a WTA. The WTA $\mathrm{Prod}(M, N)$ constructed as in Definition 2 is not necessarily **trim**, meaning that it might contain transitions with non-zero weight that are never used in the recognition. Techniques for eliminating such useless transitions are well-known, see for instance (Gécseg and Steinby, 1984, Section II.6), and can be easily implemented to run in linear time. Once $\mathrm{Prod}(M, N)$ is trim, we have a device that recognizes all and only those trees that are assigned by M to the input string w, and the weights of those trees are preserved, as seen in Theorem 2. The WTA $\mathrm{Prod}(M, N)$ can then be seen as a representation of a parse forest for the input string w, and we conclude that the construction in Definition 2, combined with some WTA reduction algorithm, represents a parsing algorithm for WTA models working in cubic time on the length of the input string and in linear time on the size of the grammar model.

More interestingly, from the framework developed in Section 4, one can also design more efficient parsing algorithms based on WTA. Borrowing from standard ideas developed in the literature for parsing based on context-free grammars, one can specialize the construction in Definition 2 in such a way that the number of useless transitions generated for $\mathrm{Prod}(M, N)$ is considerably reduced, resulting in a more efficient construction. This can be done by adopting some search strategy that guides the construction of $\mathrm{Prod}(M, N)$ using knowledge of the input string w as well as knowledge about the source model M.

As an example, we can apply step (i) only on demand, that is, we process a transition $\mu'_2(\sigma)_{q_0,q_1q_2}$ in $\mathrm{Prod}(M, N)$ only if we have already computed non-zero transitions of the form $\mu'_{k_1}(\sigma_1)_{q_1,w_1}$ and $\mu'_{k_2}(\sigma_2)_{q_2,w_2}$, for some $\sigma_1 \in \Sigma_{k_1}$, $w_1 \in Q^{k_1}$ and $\sigma_2 \in \Sigma_{k_2}$, $w_2 \in Q^{k_2}$ where Q is the state set of $\mathrm{Prod}(M, N)$. The above amounts to a bottom-up strategy that is also used in the Cocke-Kasami-Younger recognition algorithm for context-free grammars (Younger, 1967).

More sophisticated strategies are also possible. For instance, one could adopt the Earley strategy developed for context-free grammar parsing (Earley, 1970). In this case, parsing is carried out in a top-down left-to-right fashion, and the binarization construction of Section 3 is carried out on the flight. This has the additional advantage that it would be possible to use WTA models that are not restricted to the special normal form of Section 3, still maintaining the cubic time complexity in the length of the input string. We do not pursue this idea any further in this paper, since our main goal here is to outline an abstract framework for parsing based on WTA models.

5.2 Probabilistic tree automata

Let us now look into specific semirings that are relevant for statistical natural language processing. The semiring of non-negative real numbers is $\mathbb{R}_{\geq 0} = (\mathbb{R}_{\geq 0}, +, \cdot, 0, 1)$. For the remainder of the section, let $M = (Q, \Sigma, \mathbb{R}_{\geq 0}, \mu, F)$ be a WTA over $\mathbb{R}_{\geq 0}$. M is **convergent** if

$$\sum_{t \in T_\Sigma} M(t) \;<\; \infty.$$

We say that M is a **probabilistic** tree automaton (Ellis, 1971; Magidor and Moran, 1970), or PTA for short, if $\mu_k(\sigma)_{q,q_1 \cdots q_k} \in [0, 1]$ and $F(q) \in [0, 1]$, for every $\sigma \in \Sigma_k$ and $q, q_1, \ldots, q_k \in Q$. In other words, in a PTA all weights are in the range $[0, 1]$ and can be interpreted as probabilities. For a PTA M we therefore write $p_M(r) = \mathrm{wt}(r)$ and $p_M(t) = M(t)$, for each $t \in T_\Sigma$ and $r \in \mathrm{Run}_M(t)$.

A PTA is **proper** if $\sum_{q \in Q} F(q) = 1$ and

$$\sum_{\sigma \in \Sigma_k, k \geq 0, w \in Q^k} \mu_k(\sigma)_{q,w} = 1$$

for every $q \in Q$. Since the set of symbols is finite, we could have only required that the sum over all weights as shown with $w \in Q^k$ equals 1 for every $q \in Q$ and $\sigma \in \Sigma_k$. A simple rescaling would then be sufficient to arrive at our notion. Furthermore, a PTA is **consistent** if $\sum_{t \in T_\Sigma} p_M(t) = 1$. If a PTA is consistent, then p_M is a probability distribution over the set T_Σ.

The WTA M is **unambiguous** if for every input tree $t \in T_\Sigma$, there exists at most one $r \in \mathrm{Run}_M(t)$ such that $r(\varepsilon) \in F$ and $\mathrm{wt}_M(r) \neq 0$. In other words, in an unambiguous WTA, there exists at most one successful run for each input tree. Finally, M is in **final-state normal form** if there exists a state $q_S \in Q$ such that

- $F(q_S) = 1$,
- $F(q) = 0$ for every $q \in Q \setminus \{q_S\}$, and
- $\mu_k(\sigma)_{q,w} = 0$ if $w(i) = q_S$ for some $1 \leq i \leq k$.

We commonly denote the unique final state by q_S. For the following result we refer the reader to (Droste et al., 2005, Lemma 4.8) and (Bozapalidis, 1999, Lemma 22). The additional properties mentioned in the items of it are easily seen.

Theorem 3 *For every* WTA M *there exists an equivalent* WTA M' *in final-state normal form.*

- *If M is convergent (respectively, proper, consistent), then M' is such, too.*
- *If M is unambiguous, then M' is also unambiguous and for every $t \in T_\Sigma$ and $r \in \mathrm{Run}_M(t)$ we have $\mathrm{wt}_{M'}(r') = \mathrm{wt}_M(r) \cdot F(r(\varepsilon))$ where $r'(\varepsilon) = q_S$ and $r'(w) = r(w)$ for every $w \in \mathrm{Pos}(t) \setminus \{\varepsilon\}$.* □

It is not difficult to see that a proper PTA in final-state normal form is always convergent.

In statistical parsing applications we use grammar models that induce a probability distribution on the set of parse trees. In these applications, there is often the need to visit a parse tree with highest probability, among those in the parse forest obtained from the input sentence. This implements a form of disambiguation, where the most likely tree under the given model is selected, pretending that it provides the most likely syntactic analysis of the input string. In our setting, the above approach reduces to the problem of 'unfolding' a tree from a PTA $\mathrm{Prod}(M, N)$, that is assigned the highest probability.

In order to find efficient solutions for the above problem, we make the following two assumptions.

- M is in final-state normal form. By Theorem 3 this can be achieved without loss of generality.
- M is unambiguous. This restrictive assumption avoids the so-called 'spurious' ambiguity, that would result in several computations in the model for an individual parse tree.

It is not difficult to see that PTA satisfying these two properties are still more powerful than the probabilistic context-free grammar models that are commonly used in statistical natural language processing.

1: **Function** BESTPARSE(M)
2: $\mathcal{E} \leftarrow \emptyset$
3: **repeat**
4: $\quad \mathcal{A} \leftarrow \{q \mid \mu_k(\sigma)_{q,q_1\cdots q_k} > 0,\ q \notin \mathcal{E},\ q_1, \ldots, q_k \in \mathcal{E}\}$
5: $\quad$ **for all** $q \in \mathcal{A}$ **do**
6: $\quad\quad \delta(q) \leftarrow \max_{\substack{\sigma \in \Sigma_k, k \geq 0 \\ q_1, \ldots, q_k \in \mathcal{E}}} \mu_k(\sigma)_{q,q_1\cdots q_k} \cdot \prod_{i=1}^{k} \delta(q_i)$
7: $\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{\operatorname{argmax}_{q \in \mathcal{A}}\ \delta(q)\}$
8: **until** $q_S \in \mathcal{E}$
9: **return** $\delta(q_S)$

Figure 3: Search algorithm for the most probable parse in an unambiguous PTA M in final-state normal form.

Once more, we borrow from the literature on parsing for context-free grammars, and adapt a search algorithm developed by Knuth (1977); see also (Nederhof, 2003). The basic idea here is to generalize Dijkstra's algorithm to compute the shortest path in a weighted graph. The search algorithm is presented in Figure 3.

The algorithm takes as input a trim PTA M that recognizes at least one parse tree. We do not impose any bound on the rank of the alphabet symbols for M. Furthermore, M needs not be a proper PTA. In order to simplify the presentation, we provide the algorithm in a form that returns the largest probability assigned to some tree by M.

The algorithm records into the $\delta(q)$ variables the largest probability found so far for a run that brings M into state q, and stores these states into an agenda $\mathcal{A}$. States for which $\delta(q)$ becomes optimal are popped from $\mathcal{A}$ and stored into a set $\mathcal{E}$. Choices are made on a greedy base. Note that when a run has been found leading to an optimal probability $\delta(q)$, from our assumption we know that the associated tree has only one run that ends up in state q.

Since $\mathcal{E}$ is initially empty (line 2), only weights satisfying $\mu_0(\sigma)_{q,\varepsilon} > 0$ are considered when line 4 is executed for the first time. Later on (line 7) the largest probability is selected among all those that can be computed at this time, and the set $\mathcal{E}$ is populated. As a consequence, more states become

available in the agenda in the next iteration, and new transitions can now be considered. The algorithm ends when the largest probability has been calculated for the unique final state q_S.

We now analyze the computational complexity of the algorithm in Figure 3. The 'repeat-until' loop runs at most $|Q|$ times. Entirely reprocessing set $\mathcal{A}$ at each iteration would be too expensive. We instead implement $\mathcal{A}$ as a priority heap and maintain a clock for each weight $\mu_k(\sigma)_{q,q_1\cdots q_k}$, initially set to k. Whenever a new optimal probability $\delta(q)$ becomes available through $\mathcal{E}$, we decrement the clock associated with each $\mu_k(\sigma)_{q,q_1\cdots q_k}$ by d, in case $d > 0$ occurrences of q are found in the string $q_1 \cdots q_k$. In this way, at each iteration of the 'repeat-until' loop, we can consider only those weights $\mu_k(\sigma)_{q,q_1\cdots q_k}$ with associated clock of zero, compute new values $\delta(q)$, and update the heap. For each $\mu_k(\sigma)_{q,q_1\cdots q_k} > 0$, all clock updates and the computation of quantity $\mu_k(\sigma)_{q,q_1\cdots q_k} \cdot \prod_{i=1}^{k} \delta(q_i)$ (when the associated clock becomes zero) both take an amount of time proportional to the length of the transition itself. The overall time to execute these operations is therefore linear in $|M|$. Accounting for the heap, the algorithm has overall running time in $\mathcal{O}(|M| + |Q| \log |Q|)$.

The algorithm can be easily adapted to return a tree having probability $\delta(q_S)$, if we keep a record of all transitions selected in the computation along with links from a selected transition and all of the previously selected transitions that have caused its selection. If we drop the unambiguity assumption for the PTA, then the problem of computing the best parse tree becomes NP-hard, through a reduction from similar problems for finite automata (Casacuberta and de la Higuera, 2000). In contrast, the problem of computing the probability of all parse trees of a string, also called the inside probability, can be solved in polynomial time in most practical cases and will be addressed in Subsection 5.4.

5.3 Normalization

Consider the WTA $\mathrm{Prod}(M, N)$ obtained as in Definition 2. If N is a WSA encoding an input string w as in Subsection 5.1 and if M is a proper and consistent PTA, then $\mathrm{Prod}(M, N)$ is a PTA as well. However, in general $\mathrm{Prod}(M, N)$ will not be proper, nor consistent. Properness and consistency of $\mathrm{Prod}(M, N)$ are convenient in all those applications where a statistical parsing module needs to be coupled with other statistical modules, in such a way that the composition of the probability spaces still induces a probability distribution. In this subsection we deal with the more general problem of how to transform a WTA that is convergent into a PTA that is proper and consistent. This process is called **normalization**. The normalization technique we propose here has been previously explored, in the context of probabilistic context-free grammars, in (Abney et al., 1999; Chi, 1999; Nederhof and Satta, 2003).

We start by introducing some new notions. Let us assume that M is a convergent WTA. For every $q \in Q$, we define

$$\mathrm{wt}_M(q) = \sum_{\substack{t \in T_\Sigma, r \in \mathrm{Run}_M(t) \\ r(\varepsilon) = q}} \mathrm{wt}_M(r) \ .$$

Note that quantity $\mathrm{wt}_M(q)$ equals the sum of the weights of all trees in T_Σ that would be recognized by M if we set $F(q) = 1$ and $F(p) = 0$ for each $p \in Q \setminus \{q\}$, that is, if q is the unique final state of M. It is not difficult to show that, since M is convergent, the sum in the definition of $\mathrm{wt}_M(q)$ converges for each $q \in Q$. We will show in Subsection 5.4 that the quantities $\mathrm{wt}_M(q)$ can be approximated to any desired precision.

To simplify the presentation, and without any loss of generality, throughout this subsection we assume that our WTA are in final-state normal form. We can now introduce the normalization technique.

Definition 3 Let $M = (Q, \Sigma, \mathbb{R}_{\geq 0}, \mu, F)$ be a convergent WTA in final-state normal form. We construct the WTA

$$\mathrm{Norm}(M) = (Q, \Sigma, \mathbb{R}_{\geq 0}, \mu', F) \ ,$$

where for every $\sigma \in \Sigma_k$, $k \geq 0$, and $q, q_1, \ldots, q_k \in Q$

$$\mu'_k(\sigma)_{q,q_1\cdots q_k} = \mu_k(\sigma)_{q,q_1\cdots q_k} \cdot \cdot \frac{\mathrm{wt}_M(q_1) \cdot \ldots \cdot \mathrm{wt}_M(q_k)}{\mathrm{wt}_M(q)} \ . \ \square$$

We now show the claimed property for our transformation.

Theorem 4 *Let M be as in Definition 3, and let $M' = \mathrm{Norm}(M)$. Then M' is a proper and consistent* PTA*, and for every $t \in T_\Sigma$ we have $M'(t) = \frac{M(t)}{\mathrm{wt}_M(q_S)}$.* $\square$

PROOF Clearly, M' is again in final-state normal form. An easy derivation shows that

$$\mathrm{wt}_M(q) = \sum_{\substack{\sigma\in\Sigma_k \\ q_1,\ldots,q_k\in Q}} \mu_k(\sigma)_{q,q_1\cdots q_k} \cdot \prod_{i=1}^{k} \mathrm{wt}_M(q_i)$$

for every $q \in Q$. Using the previous remark, we obtain

$$\begin{aligned}
&\sum_{\sigma\in\Sigma_k, q_1,\ldots,q_k\in Q} \mu'_k(\sigma)_{q,q_1\cdots q_k} \\
&= \sum_{\sigma\in\Sigma_k, q_1,\ldots,q_k\in Q} \mu_k(\sigma)_{q,q_1\cdots q_k} \cdot \\
&\quad \cdot \frac{\mathrm{wt}_M(q_1)\cdot\ldots\cdot\mathrm{wt}_M(q_{k'})}{\mathrm{wt}_M(q)} \\
&= \frac{\displaystyle\sum_{\substack{\sigma\in\Sigma_k, \\ q_1,\ldots,q_k\in Q}} \mu_k(\sigma)_{q,q_1\cdots q_k} \cdot \prod_{i=1}^{k} \mathrm{wt}_M(q_i)}{\displaystyle\sum_{\substack{\sigma\in\Sigma_k, \\ p_1,\ldots,p_k\in Q}} \mu_k(\sigma)_{q,p_1\cdots p_k} \cdot \prod_{i=1}^{k} \mathrm{wt}_M(p_i)} \\
&= 1 \ ,
\end{aligned}$$

which proves that M' is a proper PTA.

Next, we prove an auxiliary statement. Let $t = \sigma(t_1,\ldots,t_k)$ for some $\sigma \in \Sigma_k$, $k \geq 0$, and $t_1,\ldots,t_k \in T_\Sigma$. We claim that

$$\mathrm{wt}_{M'}(r) = \frac{\mathrm{wt}_M(r)}{\mathrm{wt}_M(r(\varepsilon))}$$

for every $r \in \mathrm{Run}_M(t) = \mathrm{Run}_{M'}(t)$. For every $1 \leq i \leq k$, let $r_i \in \mathrm{Run}_M(t_i)$ be such that $r_i(w) = r(iw)$ for every $w \in \mathrm{Pos}(t_i)$. Then

$$\begin{aligned}
\mathrm{wt}_{M'}(r) &= \prod_{\substack{w\in\mathrm{Pos}(t) \\ \mathrm{rk}_t(w)=n}} \mu'_n(t(w))_{r(w),r(w1)\cdots r(wn)} \\
&= \mu'_k(\sigma)_{r(\varepsilon),r(1)\cdots r(k)} \cdot \prod_{i=1}^{k} \mathrm{wt}_{M'}(r_i) \\
&= \mu'_k(\sigma)_{r(\varepsilon),r_1(\varepsilon)\cdots r_k(\varepsilon)} \cdot \prod_{i=1}^{k} \frac{\mathrm{wt}_M(r_i)}{\mathrm{wt}_M(r_i(\varepsilon))} \\
&= \mu_k(\sigma)_{r(\varepsilon),r(1)\cdots r(k)} \cdot \frac{\mathrm{wt}_M(r_1)\cdot\ldots\cdot\mathrm{wt}_M(r_k)}{\mathrm{wt}_M(r(\varepsilon))} \\
&= \frac{\mathrm{wt}_M(r)}{\mathrm{wt}_M(r(\varepsilon))} \ .
\end{aligned}$$

Consequently,

$$\begin{aligned}
M'(t) &= \sum_{\substack{r\in\mathrm{Run}_{M'}(t) \\ r(\varepsilon)=q_S}} \mathrm{wt}_{M'}(r) \\
&= \sum_{\substack{r\in\mathrm{Run}_M(t) \\ r(\varepsilon)=q_S}} \frac{\mathrm{wt}_M(r)}{\mathrm{wt}_M(q_S)} = \frac{M(t)}{\mathrm{wt}_M(q_S)}
\end{aligned}$$

and

$$\begin{aligned}
\sum_{t\in T_\Sigma} M'(t) &= \sum_{\substack{t\in T_\Sigma, r\in\mathrm{Run}_{M'}(t) \\ r(\varepsilon)=q_S}} \mathrm{wt}_{M'}(r) \\
&= \sum_{\substack{t\in T_\Sigma, r\in\mathrm{Run}_M(t) \\ r(\varepsilon)=q_S}} \frac{\mathrm{wt}_M(r)}{\mathrm{wt}_M(q_S)} \\
&= \frac{\mathrm{wt}_M(q_S)}{\mathrm{wt}_M(q_S)} = 1 \ ,
\end{aligned}$$

which prove the main statement and the consistency of M', respectively. ∎

5.4 Probability mass of a state

Assume M is a convergent WTA. We have defined quantities $\mathrm{wt}_M(q)$ for each $q \in Q$. Note that when M is a proper PTA in final-state normal form, then $\mathrm{wt}_M(q)$ can be seen as the probability mass that 'rests' on state q. When dealing with such PTA, we use the notation $Z_M(q)$ in place of $\mathrm{wt}_M(q)$, and call Z_M the **partition function** of M. This terminology is borrowed from the literature on exponential or Gibbs probabilistic models.

In the context of probabilistic context-free grammars, the computation of the partition function has several applications, including the elimination of epsilon rules (Abney et al., 1999) and the computation of probabilistic distances between probability distributions realized by these formalisms (Nederhof and Satta, 2008). Besides what we have seen in Subsection 5.3, we will provide one more application of partition functions for the computations of so-called prefix probabilities in Subsection 5.5 We also add that, when computed on the Bar-Hillel automata of Section 4, the partition function provides the so-called inside probabilities of (Graehl et al., 2008) for the given states and substrings.

Let $|Q| = n$ and let us assume an arbitrary ordering $q_1,\ldots,q_n$ for the states in Q. We can then rewrite the definition of $\mathrm{wt}_M(q)$ as

$$\mathrm{wt}_M(q) = \sum_{\substack{\sigma\in\Sigma_k, k\geq 0 \\ q_{i_1},\ldots,q_{i_k}\in Q}} \mu_k(\sigma)_{q,q_{i_1}\cdots q_{i_k}} \cdot \prod_{j=1}^{k} \mathrm{wt}_M(q_{i_j})$$

(see proof of Theorem 4). We rename $\mathrm{wt}_M(q_i)$ with the unknown X_{q_i}, $1 \leq i \leq n$, and derive a

system of n nonlinear polynomial equations of the form

$$X_{q_i} = \sum_{\substack{\sigma \in \Sigma_k, k \geq 0 \\ q_{i_1}, \ldots, q_{i_k} \in Q}} \mu_k(\sigma)_{q, q_{i_1} \cdots q_{i_k}} \cdot X_{q_{i_1}} \cdot \ldots \cdot X_{q_{i_k}} = f_{q_i}(X_{q_1}, \ldots, X_{q_n}) \ , \quad (1)$$

for each i with $1 \leq i \leq n$.

Throughout this subsection, we will consider solutions of the above system in the extended nonnegative real number semiring

$$\mathbb{R}^{\infty}_{\geq 0} = (\mathbb{R}_{\geq 0} \cup \{\infty\}, +, \cdot, 0, 1)$$

with the usual operations extended to ∞. We can write the system in (1) in the compact form $\boldsymbol{X} = F(\boldsymbol{X})$, where we represent the unknowns as a vector $\boldsymbol{X} = (X_{q_1}, \ldots, X_{q_n})$ and F is a mapping of type $(\mathbb{R}^{\infty}_{\geq 0})^n \to (\mathbb{R}^{\infty}_{\geq 0})^n$ consisting of the polynomials $f_{q_i}(\boldsymbol{X})$.

We denote the vector $(0, \ldots, 0) \in (\mathbb{R}^{\infty}_{\geq 0})^n$ as $\boldsymbol{X}^0$. Let $\boldsymbol{X}, \boldsymbol{X'} \in (\mathbb{R}^{\infty}_{\geq 0})^n$. We write $\boldsymbol{X} \leq \boldsymbol{X'}$ if $X_{q_i} \leq X'_{q_i}$ for every $1 \leq i \leq n$. Since each polynomial $f_{q_i}(\boldsymbol{X})$ has coefficients represented by positive real numbers, it is not difficult to see that, for each $\boldsymbol{X}, \boldsymbol{X'} \in (\mathbb{R}^{\infty}_{\geq 0})^n$, we have $F(\boldsymbol{X}) \leq F(\boldsymbol{X'})$ whenever $\boldsymbol{X}^0 \leq \boldsymbol{X} \leq \boldsymbol{X'}$. This means that F is an order preserving, or **monotone**, mapping.

We observe that $((\mathbb{R}^{\infty}_{\geq 0})^n, \leq)$ is a complete lattice with least element $\boldsymbol{X}^0$ and greatest element $(\infty, \ldots, \infty)$. Since F is monotone on a complete lattice, by the Knaster-Tarski theorem (Knaster, 1928; Tarski, 1955) there exists a least and a greatest fixed-point of F that are solutions of $\boldsymbol{X} = F(\boldsymbol{X})$.

The Kleene theorem states that the least fixed-point solution of $\boldsymbol{X} = F(\boldsymbol{X})$ can be obtained by iterating F starting with the least element $\boldsymbol{X}^0$. In other words, the sequence $\boldsymbol{X}^k = F(\boldsymbol{X}^{k-1})$, $k = 1, 2, \ldots$ converges to the least fixed-point solution. Notice that each $\boldsymbol{X}^k$ provides an approximation for the partition function of M where only trees of depth not larger than k are considered. This means that $\lim_{k \to \infty} \boldsymbol{X}^k$ converges to the partition function of M, and the least fixed-point solution is also the sought solution. Thus, we can approximate $\mathrm{wt}_M(q)$ with $q \in Q$ to any degree by iterating F a sufficiently large number of times.

The fixed-point iteration method discussed above is also well-known in the numerical calculus literature, and is frequently applied to systems of nonlinear equations in general, because it can be easily implemented. When a number of standard conditions are met, each iteration of the algorithm (corresponding to the value of k above) adds a fixed number of bits to the precision of the approximated solution; see (Kelley, 1995) for further discussion.

Systems of the form $\boldsymbol{X} = F(\boldsymbol{X})$ where all $f_{q_i}(\boldsymbol{X})$ are polynomials with nonnegative real coefficients are called **monotone system of polynomials**. Monotone systems of polynomials associated with proper PTA have been specifically investigated in (Etessami and Yannakakis, 2005) and (Kiefer et al., 2007), where worst case results on exponential rate of convergence are reported for the fixed-point method.

5.5 Prefix probability

In this subsection we deal with one more application of the Bar-Hillel technique presented in Section 4. We show how to compute the so-called prefix probabilities, that is, the probability that a tree recognized by a PTA generates a string starting with a given prefix. Such probabilities have several applications in language modeling. As an example, prefix probabilities can be used to compute the probability distribution on the terminal symbol that follows a given prefix (under the given model).

For probabilistic context-free grammars, the problem of the computation of prefix probabilities has been solved in (Jelinek et al., 1992); see also (Persoon and Fu, 1975). The approach we propose here, originally formulated for probabilistic context-free grammars in (Nederhof and Satta, 2003; Nederhof and Satta, 2009), is more abstract than the previous ones, since it entirely rests on properties of the Bar-Hillel construction that we have already proved in Section 4.

Let $M = (Q, \Sigma, \mathbb{R}_{\geq 0}, \mu, F)$ be a proper and consistent PTA in final-state normal form, $\Delta = \Sigma_0 \setminus \{e\}$, and let $u \in \Delta^+$ be some string. We assume here that M is in the binary form discussed in Section 3. In addition, we assume that M has been preprocessed in order to remove from its recognized trees all of the unary branches as well as those branches that generate the null string ε. Although we do not discuss this construction at length in this paper, the result follows from a transformation casting weighted context-free grammars into Chomsky Normal Form (Fu

and Huang, 1972; Abney et al., 1999).

We define

$$\mathrm{Pref}(M,u) = \{t \,|\, t \in T_\Sigma,\ M(t) > 0,\ \mathrm{yd}(t) = uv,\ v \in \Delta^*\}\ .$$

The **prefix probability** of u under M is defined as

$$\sum_{t \in \mathrm{Pref}(M,u)} p_M(t)\ .$$

Let $|u| = n$. We define a WSA N_u with state set $P = \{p_0, \ldots, p_n\}$ and transition weights $\nu(p_{i-1}, u(i), p_i) = 1$ for each i with $1 \leq i \leq n$, and $\nu(p_n, \sigma, p_n) = 1$ for each $\sigma \in \Delta$. We also set $I(p_0) = 1$ and $F(p_n) = 1$. It is easy to see that N_u recognizes the language $\{uv \,|\, v \in \Delta^*\}$. Furthermore, the PTA $M_p = \mathrm{Prod}(M, N_u)$ specified as in Definition 2 recognizes the desired tree set $\mathrm{Pref}(M,u)$, and it preserves the weights of those trees with respect to M. We therefore conclude that $Z_{M_p}(q_S)$ is the prefix probability of u under M. Prefix probabilities can then be approximated using the fixed-point iteration method of Subsection 5.4. Rather than using an approximation method, we discuss in what follows how the prefix probabilities can be exactly computed.

Let us consider more closely the product automaton M_p, assuming that it is trim. Each state of M_p has the form $\pi = (p_i, q, p_j)$, $p_i, p_j \in P$ and $q \in Q$, with $i \leq j$. We distinguish three, mutually exclusive cases.

(i) $j < n$: From our assumption that M (and thus M_p) does not have unary or ε branches, it is not difficult to see that all $Z_{M_p}(\pi)$ can be exactly computed in time $\mathcal{O}((j-i)^3)$.

(ii) $i = j = n$: We have $\pi = (p_n, q, p_n)$. Then the equations for $Z_{M_p}(\pi)$ exactly mirror the equations for $Z_M(q)$, and $Z_{M_p}(\pi) = Z_{M_p}(q)$. Because M is proper and consistent, this means that $Z_{M_p}(\pi) = 1$.

(iii) $i < j = n$: A close inspection of Definition 2 reveals that in this case the equations (1) are all linear, assuming that we have already replaced the solutions from (i) and (ii) above into the system. This is because any weight $\mu_2(\sigma)_{\pi_0,\pi_1\pi} > 0$ in M_p with $\pi = (p_i, q, p_n)$ and $i < n$ must have $(\pi_1)_3 < n$. Quantities $Z_{M_p}(\pi)$ can then be exactly computed as the solution of a linear system of equations in time $\mathcal{O}(n^3)$.

Putting together all of the observations above, we obtain that for a proper and consistent PTA that has been preprocessed, the prefix probability of u can be computed in cubic time in the length of the prefix itself.

6 Concluding remarks

In this paper we have extended the Bar-Hillel construction to WTA, closely following the methodology proposed in (Nederhof and Satta, 2003) for weighted context-free grammars. Based on the obtained framework, we have derived several parsing algorithms for WTA, under the assumption that the input is a string rather than a tree.

As already remarked in the introduction, WTA are richer models than weighted context-free grammar, since the formers use hidden states in the recognition of trees. This feature makes it possible to define a product automaton in Definition 2 that generates exactly those trees of interest for the input string. In contrast, in the context-free grammar case the Bar-Hillel technique provides trees that must be mapped to the tree of interest using some homomorphism. For the same reason, one cannot directly convert WTA into weighted context-free grammars and then apply existing parsing algorithms for the latter formalism, unless the alphabet of nonterminal symbols is changed. Finally, our main motivation in developing a framework specifically based on WTA is that this can be extended to classes of weighted tree transducers, in order to deal with computational problems that arise in machine translation applications. We leave this for future work.

Acknowledgments

The first author has been supported by the Ministerio de Educación y Ciencia (MEC) under grant JDCI-2007-760. The second author has been partially supported by MIUR under project PRIN No. 2007TJNZRE_002.

References

S. Abney, D. McAllester, and F. Pereira. 1999. Relating probabilistic grammars and automata. In *37th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 542–549, Maryland, USA, June.

Y. Bar-Hillel, M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Y. Bar-Hillel, editor, *Language and Information: Selected Essays on their Theory and Application*, chapter 9, pages 116–150. Addison-Wesley, Reading, Massachusetts.

J. Berstel and C. Reutenauer. 1982. Recognizable formal power series on trees. *Theoret. Comput. Sci.*, 18(2):115–148.

B. Borchardt. 2005. *The Theory of Recognizable Tree Series*. Ph.D. thesis, Technische Universität Dresden.

S. Bozapalidis. 1999. Equational elements in additive algebras. *Theory Comput. Systems*, 32(1):1–33.

J. Carme, J. Niehren, and M. Tommasi. 2004. Querying unranked trees with stepwise tree automata. In *Proc. RTA*, volume 3091 of LNCS, pages 105–118. Springer.

F. Casacuberta and C. de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications; 5th International Colloquium, ICGI 2000*, pages 15–24. Springer.

Z. Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

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

J. Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, February.

S. Eilenberg. 1974. *Automata, Languages, and Machines*, volume 59 of *Pure and Applied Math.* Academic Press.

C. A. Ellis. 1971. Probabilistic tree automata. *Information and Control*, 19(5):401–416.

Z. Ésik and W. Kuich. 2003. Formal tree series. *J. Autom. Lang. Combin.*, 8(2):219–285.

K. Etessami and M. Yannakakis. 2005. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. In *22nd International Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *Lecture Notes in Computer Science*, pages 340–352, Stuttgart, Germany. Springer-Verlag.

K.S. Fu and T. Huang. 1972. Stochastic grammars and languages. *International Journal of Computer and Information Sciences*, 1(2):135–170.

F. Gécseg and M. Steinby. 1984. *Tree Automata*. Akadémiai Kiadó, Budapest.

J. Graehl, K. Knight, and J. May. 2008. Training tree transducers. *Comput. Linguist.*, 34(3):391–427.

J. Högberg, A. Maletti, and H. Vogler. 2009. Bisimulation minimisation of weighted automata on unranked trees. *Fundam. Inform.* to appear.

F. Jelinek, J.D. Lafferty, and R.L. Mercer. 1992. Basic methods of probabilistic context free grammars. In P. Laface and R. De Mori, editors, *Speech Recognition and Understanding — Recent Advances, Trends and Applications*, pages 345–360. Springer-Verlag.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

C. T. Kelley. 1995. *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

S. Kiefer, M. Luttenberger, and J. Esparza. 2007. On the convergence of Newton's method for monotone systems of polynomial equations. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, pages 217–266.

B. Knaster. 1928. Un théorème sur les fonctions d'ensembles. *Ann. Soc. Polon. Math.*, 6:133–134.

D. E. Knuth. 1977. A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5, February.

D. E. Knuth. 1997. *Fundamental Algorithms*. The Art of Computer Programming. Addison Wesley, 3rd edition.

M. Magidor and G. Moran. 1970. Probabilistic tree automata and context free languages. *Israel Journal of Mathematics*, 8(4):340–348.

M.-J. Nederhof and G. Satta. 2003. Probabilistic parsing as intersection. In *8th International Workshop on Parsing Technologies*, pages 137–148, LORIA, Nancy, France, April.

M.-J. Nederhof and G. Satta. 2008. Computation of distances for regular and context-free probabilistic languages. *Theoretical Computer Science*, 395(2-3):235–254.

M.-J. Nederhof and G. Satta. 2009. Computing partition functions of PCFGs. *Research on Language & Computation*, 6(2):139–162.

M.-J. Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143.

E. Persoon and K.S. Fu. 1975. Sequential classification of strings generated by SCFG's. *International Journal of Computer and Information Sciences*, 4(3):205–217.

M. P. Schützenberger. 1961. On the definition of a family of automata. *Information and Control*, 4(2–3):245–270.

A. Tarski. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5(2):285–309.

D. H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208.