Preservation of Recognizability for Weighted Linear Extended Top-Down Tree Transducers

Nina Seemann and Daniel Quernheim and Fabienne Braune and Andreas Maletti
University of Stuttgart, Institute for Natural Language Processing
{seemanna, daniel, braunefe, maletti}@ims.uni-stuttgart.de

Abstract

An open question in [FÜLÖP, MALETTI, VOGLER: Weighted extended tree transducers. Fundamenta Informaticae 111(2), 2011] asks whether weighted linear extended tree transducers preserve recognizability in countably complete commutative semirings. In this contribution, the question is answered positively, which is achieved with a construction that utilizes inside weights. Due to the completeness of the semiring, the inside weights always exist, but the construction is only effective if they can be effectively determined. It is demonstrated how to achieve this in a number of important cases.

1 Introduction

Syntax-based statistical machine translation (Knight, 2007) created renewed interest in tree automata and tree transducer theory (Fülöp and Vogler, 2009). In particular, it sparked research on extended top-down tree transducers (Graehl et al., 2009), which are top-down tree transducers (Rounds, 1970; Thatcher, 1970) in which the left-hand sides can contain several (or no) input symbols. A recent contribution by Fülöp et al. (2011) investigates the theoretical properties of weighted extended tree transducers over countably complete and commutative semirings (Hebisch and Weinert, 1998; Golan, 1999). Such semirings permit sums of countably many summands, which still obey the usual associativity, commutativity, and distributivity laws. We will use the same class of semirings.

Figure 1: Syntax-based machine translation pipeline.

Extended top-down tree transducers are used as translation models (TM) in syntax-based machine translation. In the standard pipeline (see Figure 1; LM is short for language model) the translation model is applied to the parses of the input sentence, which can be represented as a recognizable weighted forest (Fülöp and Vogler, 2009). In practice, only the best or the n-best parses are used, but in principle, we can use the recognizable weighted forest of all parses. In either case, the translation model transforms the input trees into a weighted forest of translated output trees. A class of transducers preserves recognizability if for every transducer of the class and each recognizable weighted forest, the weighted forest of translated output trees is again recognizable. Fülöp et al. (2011) investigates which extended top-down tree transducers preserve recognizability under forward (i.e., the setting previously described) and backward application (i.e., the setting in which we start with the output trees and apply the inverse of the translation model), but the question remained open for forward application of weighted linear extended top-down tree transducers [see Table 1 for an overview of the existing results for forward application due to Engelfriet (1975) in the unweighted case and Fülöp et al. (2010) and Fülöp et al. (2011) for the weighted case]. In conclusion, Fülöp et al. (2011) ask: “Are there a commutative semiring S that is countably complete wrt. \( \sum \), a linear wxtt \( \mathcal{M} \) [weighted extended top-down tree transducer with regular look-ahead; see Section 4], and a recognizable...
weighted tree language $\varphi$ such that $\mathcal{M}(\varphi)$ [forward application] is not recognizable? Or even harder, are there $S$ and $\mathcal{M}$ with the same properties such that $\mathcal{M}(1)$ [1 is the weighted forest in which each tree has weight 1] is not recognizable?"

In this contribution, we thus investigate preservation of recognizability (under forward application) for linear extended top-down tree transducers with regular look-ahead (Engelfriet, 1977), which are equivalent to linear weighted extended tree transducers by Fülöp et al. (2011). We show that they always preserve recognizability, thus confirming the implicit hypothesis of Fülöp et al. (2011). The essential tool for our construction is the inside weight (Lari and Young, 1990; Graehl et al., 2008) of the states of the weighted tree grammar (Alexandrakis and Bozapalidis, 1987) representing the parses. The inside weight of a state $q$ is the sum of all weights of trees accepted in this state. In our main construction (see Section 5) we first compose the input weighted tree grammar with the transducer (input restriction). This is particularly simple since we just abuse the look-ahead of the initial rules. In a second step, we normalize the obtained transducer, which yields the standard product construction typically used for input restriction. Finally, we project to the output by basically eliminating the left-hand sides. In this step, the inside weights of states belonging to deleted subtrees are multiplied to the production weight. Due to the completeness of the semiring, the inside weights always exist, but the infinite sums have to be computed effectively for the final step of the construction to be effective. This problem is addressed in Section 6, where we show several methods to effectively compute or approximate the inside weights for all states of a weighted tree grammar.

## 2 Notation

Our weights will be taken from a commutative semiring $(A, +, \cdot, 0, 1)$, which is an algebraic structure of two commutative monoids $(A, +, 0)$ and $(A, \cdot, 1)$ such that $\cdot$ distributes over $+$ and $0 \cdot a = 0$ for all $a \in A$. An infinitary sum operation $\sum$ is a family $(\sum_I)_I$ where $I$ is a countable index set and $\sum_I: A^I \to A$. Given $f: I \to A$, we write $\sum_{i \in I} f(i)$ instead of $\sum_I f$. The semiring together with the infinitary sum operation $\sum$ is countably complete (Eilenberg, 1974; Heibisch and Weinert, 1998; Golan, 1999; Karner, 2004) if for all countable sets $I$ and $a_i \in A$ with $i \in I$

- $\sum_{i \in I} a_i = a_m + a_n$ if $I = \{m, n\}$,
- $\sum_{i \in I} a_i = \sum_{j \in J} (\sum_{i \in I_j} a_i)$ if $I = \bigcup_{j \in J} I_j$ for countable sets $J$ and $I_j$ with $j \in J$ such that $I_j \cap I_{j'} = \emptyset$ for all different $j, j' \in J$, and
- $a \cdot (\sum_{i \in I} a_i) = \sum_{i \in I} (a \cdot a_i)$ for all $a \in A$.

For such a semiring, we let $a^* = \sum_{i \in \mathbb{N}} a^i$ for every $a \in A$. In the following, we assume that $(A, +, \cdot, 0, 1)$ is a commutative semiring that is countably complete with respect to $\sum$.

Our trees have node labels taken from an alphabet $\Sigma$ and leaves might also be labeled by elements of a set $V$. Given a set $T$, we write $\Sigma(T)$ for the set

$$\{\sigma(t_1, \ldots, t_k) \mid k \in \mathbb{N}, \sigma \in \Sigma, t_1, \ldots, t_k \in T\}.$$ 

The set $T_\Sigma(V)$ of $\Sigma$-trees with $V$-leaves is defined as the smallest set $T$ such that $V \cup \Sigma(T) \subseteq T$. We write $T_\Sigma$ for $T_\Sigma(\emptyset)$. For each tree $t \in T_\Sigma(V)$ we identify nodes by positions. The root of $t$ has position $\varepsilon$ and the position $iw$ with $i \in \mathbb{N}$ and $w \in \mathbb{N}^*$ addresses the position $w$ in the $i$-th direct subtree at the root. The set of all positions in $t$ is $\text{pos}(t)$. We write $t(w)$ for the label (taken from $\Sigma \cup V$) of $t$ at position $w \in \text{pos}(t)$. Similarly, we use $t|_w$ to address the subtree of $t$ that is rooted in position $w$, and $t[u]|_w$ to represent the tree that is obtained from replacing the subtree $t|_w$ at $w$ by $u \in T_\Sigma$. For a given set $L \subseteq \Sigma \cup V$ of labels, we let

$$\text{pos}_L(t) = \{w \in \text{pos}(t) \mid t(w) \in L\}$$

| model     | preserves regularity |
|-----------|----------------------|
| unweighted|                      |
| ln-XTOP   | ✓                    |
| 1-XTOP    | ✓                    |
| 1-XTOPR   | ✓                    |
| XTOP      | x                    |
| weighted  |                      |
| ln-XTOP   | ✓                    |
| 1-XTOP    | ✓                    |
| 1-XTOPR   | ✓                    |
| XTOP      | x                    |

Table 1: Overview of the known results due to Engelfriet (1975) and Fülöp et al. (2011) and our results in boxes.
be the set of all positions whose label belongs to \( L \). We also write \( \text{pos}_L(t) \) instead of \( \text{pos}^{\{l\}}(t) \).

We often use the set \( X = \{x_1, x_2, \ldots \} \) of variables and its finite subsets \( X_k = \{x_1, \ldots, x_k\} \) for every \( k \in \mathbb{N} \) to label leaves. Let \( V \) be a set potentially containing some variables of \( X \). The tree \( t \in T_\Sigma(V) \) is linear if \( |\text{pos}_x(t)| \leq 1 \) for every \( x \in X \). Moreover, \( \text{var}(t) = \{x \in X \mid \text{pos}_x(t) \neq \emptyset\} \) collects all variables that occur in \( t \). Given a finite set \( Q \) and \( T \subseteq T_\Sigma(V) \), we let

\[
Q[T] = \{q(t) \mid q \in Q, t \in T\}.
\]

We will treat elements of \( Q[T] \) (in which elements of \( Q \) are always used as unary symbols) as special trees of \( T_{\Sigma,Q}(V) \). A substitution \( \theta \) is a mapping \( \theta \colon X \to T_\Sigma(V) \). When applied to \( t \in T_\Sigma(V) \), it returns the tree \( t\theta \) which is obtained from \( t \) by replacing all occurrences of \( x \in X \) (in parallel) by \( \theta(x) \). This can be defined recursively by \( x\theta = \theta(x) \) for all \( x \in X \), \( v\theta = v \) for all \( v \in V \setminus X \), and \( \sigma(t_1, \ldots, t_k)\theta = \sigma(t_1\theta, \ldots, t_k\theta) \) for all \( \sigma \in \Sigma \) and \( t_1, \ldots, t_k \in T_\Sigma(V) \).

### 3 Weighted Tree Grammars

In this section, we will recall weighted tree grammars (Alexandrakis and Bozapalidis, 1987) [see (Fülöp and Vogler, 2009) for a modern treatment and a complete historical account]. In general, weighted tree grammars (WTGs) offer an efficient representation of weighted forests, which are sets of trees such that each individual tree is equipped with a weight. The representation is even more efficient than packed forests (Mi et al., 2008) and moreover can represent an infinite number of weighted trees. To avoid confusion between the nonterminals of a parser, which produces the forests considered here, and our WTGs, we will refer to the nonterminals of our WTG as states.

**Definition 1.** A weighted tree grammar (WTG) is a system \((Q, \Sigma, q_0, P)\) where

- \( Q \) is a finite set of states (nonterminals),
- \( \Sigma \) is the alphabet of symbols,
- \( q_0 \in Q \) is the starting state, and
- \( P \) is a finite set of productions \( q \xrightarrow{a} t \), where \( q \in Q, a \in A, \) and \( t \in T_\Sigma(Q) \).

**Example 2.** We illustrate our notation on the WTG \( G_{\text{ex1}} = (Q, \Sigma, q_s, P) \) where

- \( Q = \{q_s, q_{np}, q_{prp}, q_n, q_{adj}\} \),
- \( \Sigma \) contains “S”, “NP”, “VP”, “PP”, “DT”, “NN”, “N”, “VBD”, “PRP”, “ADJ”, “man”, “hill”, “telescope”, “laughs”, “the”, “on”, “with”, “old”, and “young”, and
- \( P \) contains the productions

\[
q_s \xrightarrow{1.0} \text{S}(q_{np}, \text{VP(VBD(laughs)))} \quad (\rho_1)
\]
\[
q_{np} \xrightarrow{0.4} \text{NP}(q_{np}, \text{PP}(q_{prp}, q_{np}))
\]
\[
q_{np} \xrightarrow{0.6} \text{NP(DT(the), } q_n) \quad (\rho_2)
\]
\[
q_{prp} \xrightarrow{0.5} \text{PRP(on)}
\]
\[
q_{prp} \xrightarrow{0.5} \text{PRP(with)}
\]
\[
q_n \xrightarrow{0.3} \text{N}(q_{adj}, q_n)
\]
\[
q_n \xrightarrow{0.3} \text{NN}(\text{man}) \quad (\rho_3)
\]
\[
q_n \xrightarrow{0.3} \text{NN(hill)}
\]
\[
q_n \xrightarrow{0.3} \text{NN(telescope)}
\]
\[
q_{adj} \xrightarrow{0.5} \text{ADJ(old)}
\]
\[
q_{adj} \xrightarrow{0.5} \text{ADJ(young)}
\]

It produces a weighted forest representing sentences about young and old men with telescopes on hills.

In the following, let \( G = (Q, \Sigma, q_0, P) \) be a WTG. For every production \( \rho = q \xrightarrow{a} t \) in \( P \), we let \( \text{wt}_G(\rho) = a \). The semantics of \( G \) is defined with the help of derivations. Let \( \xi \in T_\Sigma(Q) \) be a senetional form, and let \( w \in \text{pos}_Q(\xi) \) be such that \( w \) is the lexicographically smallest \( Q \)-labeled position in \( \xi \). Then \( \xi \xrightarrow{q_0} \xi[t]_w \) if \( \xi(w) = q \). For a sequence \( \rho_1, \ldots, \rho_n \in P \) of productions, we let \( \text{wt}_G(\rho_1 \cdots \rho_n) = \prod_{i=1}^n \text{wt}_G(\rho_i) \). For every \( q \in Q \) and \( t \in T_\Sigma(Q) \), we let

\[
\text{wt}_G(q, t) = \sum_{\rho_1, \ldots, \rho_n \in P \atop q\xrightarrow{\rho_1} T_\Sigma(q) \xrightarrow{\rho_2} \cdots \xrightarrow{\rho_n} t} \text{wt}_G(\rho_1 \cdots \rho_n).
\]

The WTG \( G \) computes the weighted forest \( L_G \colon T_\Sigma \to A \) such that \( L_G(t) = \text{wt}_G(q_0, t) \) for every \( t \in T_\Sigma \). Two WTGs are equivalent if they compute the same weighted forest. Since productions of weight 0 are useless, we often omit them.

**Example 3.** For the WTG \( G_{\text{ex2}} \) of Example 2 we display a derivation with weight 0.18 for the sentence “the man laughs” in Figure 2.

The notion of inside weights (Lari and Young, 1990) is well-established, and Maletti and Satta
(2009) consider them for WTGs. Let us recall the definition.

**Definition 4.** The inside weight of state \( q \in Q \) is
\[
\text{in}_G(q) = \sum_{t \in T_\Sigma} \text{wt}_G(q, t) .
\]

In Section 6 we demonstrate how to compute inside weights. Finally, let us introduce WTGs in normal form. The WTG \( G \) is in normal form if \( t \in \Sigma(Q) \) for all its productions \( q \xrightarrow{a} t \in P \). The following theorem was proven by Alexandrakis and Bozapalidis (1987) as Proposition 1.2.

**Theorem 5.** For every WTG there exists an equivalent WTG in normal form.

**Example 6.** The WTG \( G_{ex} \) of Example 2 is not normalized. To illustrate the normalization step, we show the normalization of the production \( \rho_2 \), which is replaced by the following three productions:

\[
q_{np} \xrightarrow{0.6} \text{NP}(q_{dt}, q_t), \quad q_{dt} \xrightarrow{1.0} \text{DT}(q_t) \quad \text{and} \quad q_t \xrightarrow{1.0} \text{the} .
\]

## 4 Weighted linear extended tree transducers

The model discussed in this contribution is an extension of the classical **top-down tree transducer**, which was introduced by Rounds (1970) and Thatcher (1970). Here we consider a weighted and extended variant that additionally has regular look-ahead. The weighted top-down tree transducer is discussed in (Fülöp and Vogler, 2009), and extended top-down tree transducers were studied in (Arnold and Dauchet, 1982; Knight and Graehl, 2005; Knight, 2007; Graehl et al., 2008; Graehl et al., 2009). The combination (weighted extended top-down tree transducer) was recently investigated by Fülöp et al. (2011), who also considered (weighted) regular look-ahead, which was first introduced by Engelfriet (1977) in the unweighted setting.

**Definition 7.** A linear extended top-down tree transducer with full regular look-ahead (1-XTOP\(_R^F\)) is a system \((S, \Sigma, \Delta, s_0, G, R)\) where
- \( S \) is a finite set of states,
- \( \Sigma \) and \( \Delta \) are alphabets of input and output symbols, respectively,
- \( s_0 \in S \) is an initial state,
- \( G = (Q, \Sigma, q_0, P) \) is a WTG, and
- \( R \) is a finite set of weighted rules of the form \( \ell \xrightarrow{\mu} r \) where
  - \( \mu \colon \text{var}(\ell) \to Q \) is the look-ahead, and
  - \( r \in T_\Delta(S[\text{var}(\ell)]) \) is the linear right-hand side.

In the following, let \( M = (S, \Sigma, \Delta, s_0, G, R) \) be an 1-XTOP\(_R^F\). We assume that the WTG \( G \) contains a state \( \top \) such that \( \text{wt}_G(\top, t) = 1 \) for every \( t \in T_\Sigma \). In essence, this state represents the trivial look-ahead. If \( \mu(x) = \top \) for every rule \( \ell \xrightarrow{\mu} r \in R \) and \( x \in \text{var}(\ell) \) (respectively, \( x \in \text{var}(\ell) \)), then \( M \) is an 1-XTOP\(_R^F\) (respectively, 1-XTOP). 1-XTOP\(_R^F\) and I-XTOP coincide exactly with the models of Fülöp et al. (2011), and in the latter model we drop the look-ahead component \( \mu \) and the WTG \( G \) completely.

**Example 8.** The rules of our running example 1-XTOP \( M_{ex} \) (over the input and output alphabet \( \Sigma \), which is also used by the WTG \( G_{ex} \) of Example 2) are displayed in Figure 3.

Next, we present the semantics. Without loss of generality, we assume that we can distinguish states from input and output symbols (i.e., \( S \cap (\Sigma \cup \Delta) = \emptyset \)). A **sentential form** of \( M \) is a tree of \( SF(M) = T_\Delta(Q[T_\Sigma]) \). Let \( \rho = \ell \xrightarrow{\mu} r \) be a rule of \( R \). Moreover, let \( \xi, \zeta \in SF(M) \) be sentential forms and \( w \in \mathbb{N}^* \) be the lexicographically smallest position in \( \text{pos}_Q(\xi) \). We write \( \xi \xrightarrow{b}_{M,\rho} \zeta \) if there exists a substitution \( \theta \colon X \to T_\Sigma \) such that
- \( \xi = \xi[\ell]_w \),
- \( \zeta = \xi[r]_w \), and
- \( b = a \cdot \prod_{x \in \text{var}(\ell)} \text{wt}_G(\mu(x), \theta(x)) \).
The tree transformation $\tau_M$ computed by $M$ is defined by

$$\tau_M(t, u) = \sum_{\rho_1, \ldots, \rho_n \in R} a_1 \cdot \ldots \cdot a_n$$

for every $t \in T_\Sigma$ and $u \in T_\Delta$.

**Example 9.** A sequence of derivation steps of the l-XTOP $M_{ex}$ is illustrated in Figure 4. The transformation it computes is capable of deleting the PP child of every NP-node with probability 0.4 as well as deleting the ADJ child of every N-node with probability 0.3. \hfill \Box

A detailed exposition to unweighted l-XTOP is presented by Arnold and Dauchet (1982) and Graehl et al. (2009).

5 The construction

In this section, we present the main construction of this contribution, in which we will construct a WTG for the forward application of another WTG via an l-XTOP. Let us first introduce the main notions. Let $L: T_\Sigma \rightarrow A$ be a weighted forest and $\tau: T_\Sigma \times T_\Delta \rightarrow A$ be a weighted tree transformation. Then the forward application of $L$ via $\tau$ yields the weighted forest $\tau(L): T_\Delta \rightarrow A$ such that $(\tau(L))(u) = \sum_{t \in T_\Sigma} L(t) \cdot \tau(t, u)$ for every $u \in T_\Delta$. In other words, to compute the weight of $u$ in $\tau(L)$, we consider all input trees $t$ and multiply their weight in $L$ with their translation weight to $u$. The sum of all those products yields the weight for $u$ in $\tau(L)$. In the particular setting considered in this contribution, the weighted forest $L$ is computed by a WTG and the weighted tree transformation $\tau$ is computed by an l-XTOP. The question is whether the resulting weighted forest $\tau(L)$ can be computed by a WTG.

Our approach to answer this question consists of three steps: (i) composition, (ii) normalization, and (iii) range projection, which we address in separate sections. Our input is
Figure 5: Two derivations (without production and grammar decoration) with weight 0.4 [top] and 0.6 [bottom] of the normalized version of the WTG $G_{\text{ex}}$ (see Example 10).

Let $M' = (S, \Sigma, s_0, G' \times G, R)$ be the l-XTOP constructed in the previous section and they have the same left-hand side $\ell$. It can be determined easily that $wt_{G_{\text{ex}}}(q_s, \ell \theta) \neq 0$ only if
- $\theta(x_1) \theta(x_2) \theta(x_3) = q_{np} q_{pp} q_{v}$ or
- $\theta(x_1) \theta(x_2) \theta(x_3) = q_{dt} q_{n} q_{v}$.

Example 10. Let us illustrate the construction on the WTG $G_{\text{ex}}$ of Example 2 and the l-XTOP $M_{\text{ex}}$ of Example 8. According to our assumptions, $G_{\text{ex}}$ should first be normalized (see Theorem 5).

Theorem 11. For every $t \in T_{\Sigma}$ and $u \in T_{\Delta}$, we have $\tau_{M'}(t, u) = L(t) \cdot \tau(t, u)$.

Proof. We prove an intermediate property for each derivation of $M$. Let

$$s_0(t) \xrightarrow{b_1} \ldots \xrightarrow{b_s} M, \rho_1 \ldots u$$

be a derivation of $M$. Let $\rho_1 = \ell \xrightarrow{a} \mu$ be the first rule, which trivially must be in $R_{s_0}$. Then for every $\theta$: $\var(\ell) \rightarrow Q'$, there exists a derivation

$$s_0(t) \xrightarrow{c_1} M', \rho_1 \xrightarrow{c_2} M', \rho_2 \ldots \xrightarrow{c_b} M', \rho_s \ldots u$$

in $M'$ such that

$$c_1 = b_1 \cdot wt_{G'}(q_0, \ell \theta) \cdot \prod_{x \in \var(\ell)} wt_{G'}(\theta(x), \theta'(x))$$

where $\theta': \var(\ell) \rightarrow T_{\Sigma}$ is such that $t = \ell \theta'$. Since we sum over all such derivations and

$$\sum_{\theta: \var(\ell) \rightarrow Q'} \prod_{x \in \var(\ell)} wt_{G'}(\theta(x), \theta'(x)) = wt_{G'}(q_0, t) = L_G(t)$$

by a straightforward extension of Lemma 4.1.8 of (Borchardt, 2005), we obtain that the derivations in $M'$ sum to $L_G(t) \cdot b_1 \ldots b_s$ as desired. The main property follows trivially from the intermediate result.

5.2 Normalization

Currently, the weights of the input WTG are only on the initial rules and in its look-ahead. Next, we use essentially the same method as in the previous section to remove the look-ahead from all variables that are not deleted. Let $M' = (S, \Sigma, \Delta, s_0, G \times G', R)$ be the l-XTOP constructed in the previous section and

$$\ell \xrightarrow{a_{\mu}} r$$

in $R'$, where $\mu'(x) = (\mu(x), \theta(x))$ for every $x \in \var(\ell)$. The main property follows trivially from the intermediate result.\[\square\]
Example 10. \( \theta \) placing the subtree or (ii) \( \mu \) such that

\[
\begin{align*}
\text{(left)} \quad & \mu(x_1) \mu(x_2) \mu(x_3) = q_{np}q_{pp}q_u \\
\text{(right)} \quad & \mu(x_1) \mu(x_2) \mu(x_3) = q_{da}q_{in}q_u \quad \text{and} \quad c = 0.6 \quad \text{(see Example 10)}.
\end{align*}
\]

Figure 6: 4 new \( \text{XTOP}^R \) rules, where \( \mu \) and \( c \) are either (i) \( \mu(x_1)x(x_2)\mu(x_3) = q_{np}q_{pp}q_u \) and \( c = 0.4 \) or (ii) \( \mu(x_1)x(x_2)\mu(x_3) = q_{da}q_{in}q_u \) and \( c = 0.6 \) (see Example 10).

\[
\begin{align*}
\text{(left)} \quad & \mu(x_1)\mu(x_2)\mu(x_3) = q_{np}q_{pp}q_u \\
\text{(right)} \quad & \mu(x_1)\mu(x_2)\mu(x_3) = q_{da}q_{in}q_u \quad \text{and} \quad c = 0.6 \quad \text{(see Example 10)}.
\end{align*}
\]

Figure 7: New \( \text{XTOP}^R \) rules, where \( \mu(x_2) = q_{pp} \) [left] and \( \mu(x_2) = q_{n} \) [right] (see Figure 6).

\[
\begin{align*}
\rho = \ell \overset{a}{\rightarrow} \mu \in R \quad \text{be a rule with} \quad \mu(x) = \langle \top, q' \rangle \\
\text{for some} \quad q' \in Q' \setminus \{\top\} \quad \text{and} \quad x \in \text{var}(r). \quad \text{Note} \quad \mu(x) = \langle \top, q' \rangle \quad \text{for some} \quad q' \in Q' \quad \text{for all} \quad x \in \text{var}(r) \quad \text{since} \quad M \quad \text{is an \( \text{XTOP}^R \).}
\end{align*}
\]

such that \( R' \) contains the rule \( \ell \overset{a}{\rightarrow} \mu \in R' \), where

\[
\mu'(x') = \begin{cases} 
\langle \top, \top \rangle & \text{if } x = x' \\
\mu(x') & \text{otherwise}
\end{cases}
\]

for all \( x' \in \text{var}(\ell) \) and \( r' \) is obtained from \( r \) by replacing the subtree \( s(x) \) with \( s \in S \) by \( \langle s, q' \rangle(x) \).

Additionally, for every rule \( \ell^{m} \overset{a}{\rightarrow} \mu^{m} \in R_{s} \) and \( \theta : \text{var}(\ell^{m}) \to Q' \), the rule

\[
\ell^{m} \overset{a^{m}}{\rightarrow} \mu^{m} \quad \text{is in} \quad R', \quad \text{where} \quad \mu^{m}(x) = \langle \mu^{m}(x), \theta(x) \rangle \quad \text{for every} \quad x \in \text{var}(\ell). \quad \text{This procedure is iterated until we obtain an \( \text{XTOP}^R \) \( M'' \). Clearly, the iteration must terminate since we do not change the rule shape, which yields that the size of the potential rule set is bounded.}

**Theorem 12.** The \( \text{XTOP}^R \) \( M'' \) and the \( \text{XTOP}^R \) \( M' \) are equivalent.

Proof. It can be proved that the \( \text{XTOP}^R \) constructed after each iteration is equivalent to its input \( \text{XTOP}^R \) in the same fashion as in Theorem 11 with the only difference that the rule replacement now occurs anywhere in the derivation (not necessarily at the beginning) and potentially several times. Consequently, the finally obtained \( \text{XTOP}^R \) \( M'' \) is equivalent to \( M' \).

Example 13. Let us reconsider the \( \text{XTOP}^R \) constructed in the previous section and apply the normalization step. The interesting rules (i.e., those rules \( \ell \overset{a}{\rightarrow} r \) where \( \text{var}(r) \neq \text{var}(l) \)) are displayed in Figures 7 and 8.

**5.3 Range projection**

We now have an \( \text{XTOP}^R \) \( M'' \) with rules \( R'' \) computing \( \tau_{M''}(t, u) = L(t) \cdot \tau(t, u) \). In the final step, we simply disregard the input and project to the output. Formally, we want to construct a WTG \( G'' \) such that

\[
\text{in}_{G''}(q) = \sum_{t \in T} \text{wt}_{G''}(q, t) .
\]

We construct the WTG

\[
G'' = (S \cup S \times Q', \Delta, s_0, P'')
\]

such that \( \ell(\varepsilon) \overset{a}{\rightarrow} r' \) is in \( P'' \) for every rule \( \ell \overset{a}{\rightarrow} r \in R'' \), where

\[
c = a \cdot \prod_{x \in \text{var}(\ell) \setminus \text{var}(r)} \text{in}_{G''}(\mu(x))
\]

and \( r' \) is obtained from \( r \) by removing the variables of \( X \). If the same production is constructed from several rules, then we add the weights. Note that the WTG \( G'' \) can be effectively computed if \( \text{in}_{G''}(q) \) is computable for every state \( q \).
Theorem 14. For every \( u \in T_{\Delta} \), we have
\[
L_G(u) = \sum_{t \in T_{\Sigma}} L(t) \cdot \tau(t, u) = (\tau(L))(u).
\]

Example 15. The WTG productions for the rules of Figures 7 and 8 are:
\[
\begin{align*}
s_0 & \rightarrow^{0.4} s((s_1, q_{np}), \text{VP}((s_3, q_u))) \\
& \rightarrow^{0.6} s((s_1, q_{dt}), \text{VP}((s_3, q_v))) \\
\langle s_2, q_n \rangle & \rightarrow^{0.3} \langle s_2, q_n \rangle \\
\langle s_1, q_{np} \rangle & \rightarrow^{0.5} \langle s_1, q_{np} \rangle \\
\langle s_1, q_{np} \rangle & \rightarrow^{0.5} \langle s_1, q_{dt} \rangle.
\end{align*}
\]

Note that all inside weights are 1 in our example. The first production uses the inside weight of \( q_{pp} \), whereas the second production uses the inside weight of \( q_n \). Note that the third production can be constructed twice.

6 Computation of inside weights

In this section, we address how to effectively compute the inside weight for every state. If the WTG \( G = (Q, \Sigma, q_0, P) \) permits only finitely many derivations, then for every \( q \in Q \), the inside weight \( \text{in}_G(q) \) can be computed according to Definition 4 because \( \text{wt}_G(q, t) = 0 \) for almost all \( t \in T_{\Sigma} \). If \( P \) contains (useful) recursive rules, then this approach does not work anymore. Our WTG \( G_{\text{ex}} \) of Example 2 has the following two recursive rules:
\[
\begin{align*}
q_{np} & \rightarrow^{0.4} \text{NP}(q_{np}, \text{PP}(q_{pp}, q_{np})) \\
q_n & \rightarrow^{0.3} \text{N}(q_{adj}, q_n).
\end{align*}
\]

The dependency graph of \( G_{\text{ex}} \), which is shown in Figure 9, has cycles, which yields that \( G_{\text{ex}} \) permits infinitely many derivations. Due to the completeness of the semiring, even the infinite sum of Definition 4 is well-defined, but we still have to compute it. We will present two simple methods to achieve this: (a) an analytic method and (b) an approximation in the next sections.

6.1 Analytic computation

In simple cases we can compute the inside weight using the stars \( a^* \), which we defined in Section 2. Let us first list some interesting countably complete semirings for NLP applications and their corresponding stars.

- **Probabilities**: \( (\mathbb{R}_{\geq 0}, +, \cdot, 0, 1) \) where \( \mathbb{R}_{\geq 0} \) contains all nonnegative real numbers and \( \infty \), which is bigger than every real number. For every \( a \in \mathbb{R}_{\geq 0} \) we have
  \[
a^* = \begin{cases} 
1/a & \text{if } 0 \leq a < 1 \\
\infty & \text{otherwise}
\end{cases}
\]

- **VITERBI**: \([0, 1], \text{max}, \cdot, 0, 1\) where [0, 1] is the (inclusive) interval of real numbers between 0 and 1. For every \( 0 \leq a \leq 1 \) we have \( a^* = 1 \).

- **Tropical**: \((\mathbb{R}_{\geq 0}, \min, +, \infty, 0)\) where \( a^* = 0 \) for every \( a \in \mathbb{R}_{\geq 0} \).

- **Tree unification**: \( (2^T_{\Sigma}(X_1), \cup, \sqcup, \emptyset, \{x_1\}) \)
  where \( 2^T_{\Sigma}(X_1) = \{ L \mid L \subseteq T_{\Sigma}(X_1) \} \) and \( \sqcup \) is unification (where different occurrences of \( x_1 \) can be replaced differently) extended to sets as usual. For every \( L \subseteq T_{\Sigma}(X_k) \) we have \( L^* = \{ x_1 \} \cup (L \sqcup L) \).

We can always try to develop a regular expression (Fülöp and Vogler, 2009) for the weighted forest recognized by a certain state, in which we can drop the actual trees and only compute with the weights. This is particularly easy if our WTG has only left- or right-recursive productions because in this case we obtain classical regular expressions (for strings). Let us consider production \( \rho_5 \). It is right-recursive. On the string level, we obtain the following unweighted regular expression for the string language generated by \( q_n \):
\[
L(q_{adj})^*(\text{man} \mid \text{hill} \mid \text{telescope})
\]
where \( L(q_{adj}) = \{ \text{old}, \text{young} \} \) is the set of strings generated by \( q_{adj} \). Correspondingly, we can derive the inside weight by replacing the generated string with the weights used to derive them. For example, the production \( \rho_5 \), which generates the state \( q_{adj} \), has weight 0.3. We obtain the expression
\[
\text{in}_G(q_n) = (0.3 \cdot \text{in}_G(q_{adj}))^* \cdot (0.3 + 0.2 + 0.2).
\]
Example 16. If we calculate in the probability semiring and $\text{in}_G(q_{\text{adj}}) = 1$, then
\[ \text{in}_G(q_n) = \frac{1}{1 - 0.3} \cdot (0.3 + 0.2 + 0.2) = 1 , \]
as expected (since our productions induce a probability distribution on all trees generated from each state).

Example 17. If we calculate in the tropical semiring, then we obtain
\[ \text{in}_G(q_n) = \min(0.3, 0.2, 0.2) = 0.2 . \]

It should be stressed that this method only allows us to compute $\text{in}_G(q)$ in very simple cases (e.g., WTG containing only left- or right-recursive productions). The production $\rho_4$ has a more complicated recursion, so this simple method cannot be used for our full example WTG.

However, for extremal semirings the inside weight always coincides with a particular derivation. Let us also recall this result. The semiring is extremal if $a + a' \in \{a, a'\}$ for all $a, a' \in A$. The VITERBI and the tropical semiring are extremal. Recall that
\[
\text{in}_G(q) = \sum_{t \in T_S} \text{wt}_G(q, t) = \sum_{t \in T_S} \sum_{\rho_1, \ldots, \rho_n \in P, q = \rho_1 \cdots \rho_n \cdot t} \text{wt}_G(\rho_1 \cdots \rho_n) ,
\]
which yields that $\text{in}_G(q)$ coincides with the derivation weight $\text{wt}_G(\rho_1 \cdots \rho_n)$ of some derivation $q \Rightarrow^* \rho_1 \cdots \rho_n \cdot t$ for some $t \in T_G$. In the VITERBI semiring this is the highest scoring derivation and in the tropical semiring it is the lowest scoring derivation (mind that in the VITERBI semiring the production weights are multiplied in a derivation, whereas they are added in the tropical semiring). There are efficient algorithms (Viterbi, 1967) that compute those derivations and their weights.

6.2 Numerical Approximation

Next, we show how to obtain a numerical approximation of the inside weights (up to any desired precision) in the probability semiring, which is the most important of all semirings discussed here. A similar approach was used by Stolcke (1995) for context-free grammars. To keep the presentation simple, let us suppose that $G = (Q, \Sigma, q_0, P)$ is in normal form (see Theorem 5). The method works just as well in the general case.

We first observe an important property of the inside weights. For every state $q \in Q$
\[ \text{in}_G(q) = \sum_{q^a \sigma(q_1, \ldots, q_n) \in P} a \cdot \text{in}_G(q_1) \cdot \ldots \cdot \text{in}_G(q_n) , \]
which can trivially be understood as a system of equations (where each $\text{in}_G(q)$ with $q \in Q$ is a variable). Since there is one such equation for each variable $\text{in}_G(q)$ with $q \in Q$, we have a system of $|Q|$ non-linear polynomial equations in $|Q|$ variables.

Several methods to solve non-linear systems of equations are known in the numerical calculus literature. For example, the Newton-Raphson method allows us to iteratively compute the roots of any differentiable real-valued function, which can be used to solve our system of equations because we can compute the Jacobi matrix for our system of equations easily. Given a good starting point, the Newton-Raphson method assures quadratic convergence to a root. A good starting point can be obtained, for example, by bisection (Corliss, 1977). Another popular root-finding approximation is described by Brent (1973).

Example 18. For the WTG of Example 2 we obtain the following system of equations:
\[
\begin{align*}
\text{in}_G(q_s) &= 1.0 \cdot \text{in}_G(q_{np}) \\
\text{in}_G(q_{np}) &= 0.4 \cdot \text{in}_G(q_{np}) \cdot \text{in}_G(q_{prp}) \cdot \text{in}_G(q_{np}) + 0.6 \cdot \text{in}_G(q_n) \\
\text{in}_G(q_n) &= 0.3 \cdot \text{in}_G(q_{adj}) \cdot \text{in}_G(q_n) + 0.3 + 0.2 + 0.2 \\
\text{in}_G(q_{adj}) &= 0.5 + 0.5 \\
\text{in}_G(q_{prp}) &= 0.5 + 0.5 .
\end{align*}
\]
Together with $\text{in}_G(q_n) = 1$, which we already calculated in Example 16, the only interesting value is
\[ \text{in}_G(q_s) = \text{in}_G(q_{np}) = 0.4 \cdot \text{in}_G(q_{np})^2 + 0.6 , \]
which yields the roots $\text{in}_G(q_{np}) = 1$ and $\text{in}_G(q_{np}) = 1.5$. The former is the desired solution. As before, this is the expected solution.  \( \Box \)
References

Athanasios Alexandrakis and Symeon Bozapalidis.
1987. Weighted grammars and Kleene’s theorem.
Inf. Process. Lett., 24(1):1–4.

André Arnold and Max Dauchet. 1982. Morphismes
et et bimorphismes d’arbres. Theoret. Comput. Sci.,
20(1):33–93.

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

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

Richard P. Brent. 1973. Algorithms for Minimization
without Derivatives. Series in Automatic Computation.
Prentice Hall, Englewood Cliffs, NJ, USA.

George Corliss. 1977. Which root does the bisection
algorithm find? SIAM Review, 19(2):325–327.

Samuel Eilenberg. 1974. Automata, Languages, and
Machines — Volume A, volume 59 of Pure and Ap-
plied Math. Academic Press.

Joost Engelfriet. 1975. Bottom-up and top-down tree
transformations — a comparison. Math. Systems
Theory, 9(3):198–231.

Joost Engelfriet. 1977. Top-down tree transducers
with regular look-ahead. Math. Systems Theory,
10(1):289–303.

Zoltán Fülöp and Heiko Vogler. 2009. Weighted tree
automata and tree transducers. In Manfred Droste,
Werner Kuich, and Heiko Vogler, editors, Hand-
book of Weighted Automata, EATCS Monographs
on Theoret. Comput. Sci., chapter 9, pages 313–
403. Springer.

Zoltán Fülöp, Andreas Maletti, and Heiko Vogler.
2010. Preservation of recognizability for syn-
chronous tree substitution grammars. In Proc. 1st
Workshop Applications of Tree Automata in Natu-
ral Language Processing, pages 1–9. Association for Computational Linguistics.

Zoltán Fülöp, Andreas Maletti, and Heiko Vogler.
2011. Weighted extended tree transducers. Fund-
am. Inform., 111(2):163–202.

Jonathan S. Golan. 1999. Semirings and their Appli-
cations. Kluwer Academic, Dordrecht.

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

Jonathan Graehl, Mark Hopkins, Kevin Knight, and
Andreas Maletti. 2009. The power of extended
top-down tree transducers. SIAM J. Comput.,
39(2):410–430.

Udo Hebisch and Hanns J. Weinert. 1998. Semirings
— Algebraic Theory and Applications in Computer
Science. World Scientific.

Georg Kermer. 2004. Continuous monoids and semi-
rings. Theoret. Comput. Sci., 318(3):355–372.

Kevin Knight and Jonathan Graehl. 2005. An over-
view of probabilistic tree transducers for natural
language processing. In Proc. 6th Int. Conf. Com-
putational Linguistics and Intelligent Text Proces-
sing, volume 3406 of LNCS, pages 1–24. Springer.

Kevin Knight. 2007. Capturing practical natural
language transformations. Machine Translation,
21(2):121–133.

Karim Lari and Steve J. Young. 1990. The esti-
mation of stochastic context-free grammars using
the inside-outside algorithm. Computer Speech and
Language, 4(1):35–56.

Andreas Maletti and Giorgio Satta. 2009. Parsing al-
gorithms based on tree automata. In Proc. 11th Int.
Workshop Parsing Technologies, pages 1–12. Asso-
ciation for Computational Linguistics.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-
based translation. In Proc. 46th Ann. Meeting of
the ACL, pages 192–199. Association for Computa-
tional Linguistics.

William C. Rounds. 1970. Mappings and grammars
on trees. Math. Systems Theory, 4(3):257–287.

Andreas Stolcke. 1995. An efficient probabilistic
context-free parsing algorithm that computes prefix
probabilities. Comput. Linguist., 21(2):165–201.

James W. Thatcher. 1970. Generalized2 sequential
machine maps. J. Comput. System Sci., 4(4):339–
367.

Andrew J. Viterbi. 1967. Error bounds for convo-
lutional codes and an asymptotically optimum de-
coding algorithm. IEEE Trans. Inform. Theory,
13(2):260–269.