Preservation of Recognizability for Synchronous Tree Substitution Grammars

Zoltán Fülöp
Department of Computer Science
University of Szeged
Szeged, Hungary

Andreas Maletti
Departament de Filologies Romàniques
Universitat Rovira i Virgili
Tarragona, Spain

Heiko Vogler
Faculty of Computer Science
Technische Universität Dresden
Dresden, Germany

Abstract

We consider synchronous tree substitution grammars (STSG). With the help of a characterization of the expressive power of STSG in terms of weighted tree bimorphisms, we show that both the forward and the backward application of an STSG preserve recognizability of weighted tree languages in all reasonable cases. As a consequence, both the domain and the range of an STSG without chain rules are recognizable weighted tree languages.

1 Introduction

The syntax-based approach to statistical machine translation (Yamada and Knight, 2001) becomes more and more competitive in machine translation, which is a subfield of natural language processing (NLP). In this approach the full parse trees of the involved sentences are available to the translation model, which can base its decisions on this rich structure. In the competing phrase-based approach (Koehn et al., 2003) the translation model only has access to the linear sentence structure.

There are two major classes of syntax-based translation models: *tree transducers* and *synchronous grammars*. Examples in the former class are the top-down tree transducer (Rounds, 1970; Thatcher, 1970), the extended top-down tree transducer (Arnold and Dauchet, 1982; Galley et al., 2004; Knight and Graehl, 2005; Graehl et al., 2008; Maletti et al., 2009), and the extended multi bottom-up tree transducer (Lilin, 1981; Engelfriet et al., 2009; Maletti, 2010). The latter class contains the syntax-directed transductions of Lewis II and Stearns (1968), the generalized syntax-directed transductions (Aho and Ullman, 1969), the synchronous tree substitution grammar (STSG) by Schabes (1990) and the synchronous tree adjoining grammar (STAG) by Abeillé et al. (1990) and Shieber and Schabes (1990). The first bridge between those two classes were established in (Martin and Vere, 1970). Further comparisons can be found in (Shieber, 2004) for STSG and in (Shieber, 2006) for STAG.

One of the main challenges in NLP is the ambiguity that is inherent in natural languages. For instance, the sentence *"I saw the man with the telescope"* has several different meanings. Some of them can be distinguished by the parse tree, so that probabilistic parsers (Nederhof and Satta, 2006) for natural languages can (partially) achieve the disambiguation. Such a parser returns a set of parse trees for each input sentence, and in addition, each returned parse tree is assigned a likelihood. Thus, the result can be seen as a mapping from parse trees to probabilities where the impossible parses are assigned the probability 0. Such mappings are called weighted tree languages, of which some can be finitely represented by weighted regular tree grammars (Alexandrakis and Bozapalidis, 1987). Those weighted tree languages are *recognizable* and there exist algorithms (Huang and Chiang, 2005) that efficiently extract the k-best parse trees (i.e., those with the highest probability) for further processing.

In this paper we consider synchronized tree substitution grammars (STSG). To overcome a technical difficulty we add (grammar) nonterminals to them. Since an STSG often uses the nonterminals of a context-free grammar as terminal symbols (i.e., its derived trees contain both terminal and nonterminal symbols of the context-free grammar), we call the newly added (grammar) nonterminals of the STSG *states*. Substitution does no longer take place at synchronized nonterminals (of the context-free grammar) but at synchronized states (one for the input and one for the output side). The states themselves will not appear in the final derived trees, which yields that it is sufficient to assume that only identical states are synchronized. Under those conventions a rule of an STSG has the form $q \to (s, t, V, a)$ where q is a state, $a \in \mathbb{R}_{\geq 0}$ is the rule weight, s is an input tree that can contain states at the leaves, and t is an output tree that can also contain states. Finally, the synchronization is defined by V, which is a bijection between the state-labeled leaves of s and t. We require that V only relates identical states.

The rules of an STSG are applied in a step-wise manner. Here we use a derivation relation to define the semantics of an STSG. It can be understood as the synchronization of the derivation relations of two regular tree grammars (Gécseg and Steinby, 1984; Gécseg and Steinby, 1997) where the synchronization is done on nonterminals (or states) in the spirit of syntax-directed transductions (Lewis II and Stearns, 1968). Thus each sentential form is a pair of (nonterminal-) connected trees.

An STSG $\mathcal{G}$ computes a mapping $\tau_{\mathcal{G}}$, called its weighted tree transformation, that assigns a weight to each pair of input and output trees, where both the input and output tree may not contain any state. This transformation is obtained as follows: We start with two copies of the initial state that are synchronized. Given a connected tree pair (ξ, ζ), we can apply the rule $q \to (s, t, V, a)$ to each pair of synchronized states q. Such an application replaces the selected state q in ξ by s and the corresponding state q in ζ by t. All the remaining synchronized states and the synchronized states of V remain synchronized. The result is a new connected tree pair. This step charges the weight a. The weights of successive applications (or steps) are multiplied to obtain the weight of the derivation. The weighted tree transformation $\tau_{\mathcal{G}}$ assigns to each pair of trees the sum of all weights of derivations that derive that pair.

Shieber (2004) showed that for every classical unweighted STSG there exists an equivalent bimorphism (Arnold and Dauchet, 1982). The converse result only holds up to deterministic relabelings (Gécseg and Steinby, 1984; Gécseg and Steinby, 1997), which remove the state information from the input and output tree. It is this difference that motivates us to add states to STSG. We generalize the result of Shieber (2004) and prove that every weighted tree transformation that is computable by an STSG can also be computed by a weighted bimorphism and vice versa.

Given an STSG and a recognizable weighted tree language φ of input trees, we investigate under which conditions the weighted tree language obtained by applying $\mathcal{G}$ to φ is again recognizable. In other words, we investigate under which conditions the forward application of $\mathcal{G}$ preserves recognizability. The same question is investigated for backward application, which is the corresponding operation given a recognizable weighted tree language of output trees. Since STSG are symmetric (i.e., input and output can be exchanged), the results for backward application can be obtained easily from the results for forward application.

Our main result is that forward application preserves recognizability if the STSG $\mathcal{G}$ is output-productive, which means that each rule of $\mathcal{G}$ contains at least one output symbol that is not a state. Dually, backward application preserves recognizability if $\mathcal{G}$ is input-productive, which is the analogous property for the input side. In fact, those results hold for weights taken from an arbitrary commutative semiring (Hebisch and Weinert, 1998; Golan, 1999), but we present the results only for probabilities.

2 Preliminary definitions

In this contribution we will work with *ranked trees*. Each symbol that occurs in such a tree has a fixed rank that determines the number of children of nodes with this label. Formally, let Σ be a *ranked alphabet*, which is a finite set Σ together with a mapping $\mathrm{rk}_\Sigma \colon \Sigma \to \mathbb{N}$ that associates a rank $\mathrm{rk}_\Sigma(\sigma)$ with every $\sigma \in \Sigma$. We let $\Sigma_k = \{\sigma \in \Sigma \mid \mathrm{rk}_\Sigma(\sigma) = k\}$ be the set containing all symbols in Σ that have rank k. A *Σ-tree* indexed by a set Q is a tree with nodes labeled by elements of $\Sigma \cup Q$, where the nodes labeled by some $\sigma \in \Sigma$ have exactly $\mathrm{rk}_\Sigma(\sigma)$ children and the nodes with labels of Q have no children. Formally, the set $T_\Sigma(Q)$ of (term representations of) Σ-trees indexed by a set Q is the smallest set T such that

- $Q \subseteq T$ and
- $\sigma(t_1, \dots, t_k) \in T$ for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T$.

We generally write α instead of $\alpha()$ for all $\alpha \in \Sigma_0$.

We frequently work with the set $\mathrm{pos}(t)$ of *positions* of a Σ-tree t, which is defined as follows. If $t \in Q$, then $\mathrm{pos}(t) = \{\varepsilon\}$, and if $t = \sigma(t_1, \dots, t_k)$, then

$$\mathrm{pos}(t) = \{\varepsilon\} \cup \{iw \mid 1 \leq i \leq k, w \in \mathrm{pos}(t_i)\} \enspace .$$

Thus, each position is a finite (possibly empty) sequence of natural numbers. Clearly, each position designates a node of the tree, and vice versa. Thus we identify nodes with positions. As usual, a *leaf* is a node that has no children. The set of all leaves of t is denoted by $\mathrm{lv}(t)$. Clearly, $\mathrm{lv}(t) \subseteq \mathrm{pos}(t)$.

The label of a position $w \in \mathrm{pos}(t)$ is denoted by $t(w)$. Moreover, for every $A \subseteq \Sigma \cup Q$, let $\mathrm{pos}_A(t) = \{w \in \mathrm{pos}(t) \mid t(w) \in A\}$ and $\mathrm{lv}_A(t) = \mathrm{pos}_A(t) \cap \mathrm{lv}(t)$ be the sets of positions and leaves that are labeled with an element of A, respectively. Let $t \in T_\Sigma(Q)$ and $w_1, \dots, w_k \in \mathrm{lv}_Q(t)$ be k (pairwise) different leaves. We write $t[w_1 \leftarrow t_1, \dots, w_k \leftarrow t_k]$ or just $t[w_i \leftarrow t_i \mid 1 \leq i \leq k]$ with $t_1, \dots, t_k \in T_\Sigma(Q)$ for the tree obtained from t by replacing, for every $1 \leq i \leq k$, the leaf w_i with the tree t_i.

For the rest of this paper, let Σ and Δ be two arbitrary ranked alphabets. To avoid consistency issues, we assume that a symbol σ that occurs in both Σ and Δ has the same rank in Σ and Δ; i.e., $\mathrm{rk}_\Sigma(\sigma) = \mathrm{rk}_\Delta(\sigma)$. A *deterministic relabeling* is a mapping $r\colon \Sigma \to \Delta$ such that $r(\sigma) \in \Delta_k$ for every $\sigma \in \Sigma_k$. For a tree $s \in T_\Sigma$, the relabeled tree $r(s) \in T_\Delta$ is such that $\mathrm{pos}(r(s)) = \mathrm{pos}(s)$ and $\big(r(s)\big)(w) = r(s(w))$ for every $w \in \mathrm{pos}(s)$. The class of tree transformations computed by deterministic relabelings is denoted by dREL.

A tree language (over Σ) is a subset of T_Σ. Correspondingly, a *weighted tree language* (over Σ) is a mapping $\varphi\colon T_\Sigma \to \mathbb{R}_{\geq 0}$. A *weighted tree transformation* (over Σ and Δ) is a mapping $\tau\colon T_\Sigma \times T_\Delta \to \mathbb{R}_{\geq 0}$. Its *inverse* is the weighted tree transformation $\tau^{-1}\colon T_\Delta \times T_\Sigma \to \mathbb{R}_{\geq 0}$, which is defined by $\tau^{-1}(t, s) = \tau(s, t)$ for every $t \in T_\Delta$ and $s \in T_\Sigma$.

3 Synchronous tree substitution grammars with states

Let Q be a finite set of *states* with a distinguished *initial state* $q_\mathrm{S} \in Q$. A *connected tree pair* is a tuple (s, t, V, a) where $s \in T_\Sigma(Q)$, $t \in T_\Delta(Q)$, and $a \in \mathbb{R}_{\geq 0}$. Moreover, $V\colon \mathrm{lv}_Q(s) \to \mathrm{lv}_Q(t)$ is a bijective mapping such that $s(u) = t(v)$ for every $(u, v) \in V$. We will often identify V with its graph. Intuitively, a connected tree pair (s, t, V, a) is a pair of trees (s, t) with a weight a such that each node labeled by a state in s has a corresponding node in t, and vice versa. Such a connected tree pair (s, t, V, a) is *input-productive* and *output-productive* if $s \notin Q$ and $t \notin Q$, respectively. Let Conn denote the set of all connected tree pairs that use the index set Q. Moreover, let $\mathrm{Conn}_\mathrm{p} \subseteq \mathrm{Conn}$ contain all connected tree pairs that are input- or output-productive.

A *synchronous tree substitution grammar $\mathcal{G}$ (with states) over Σ, Δ, and Q* (for short: STSG), is a finite set of *rules* of the form $q \to (s, t, V, a)$ where $q \in Q$ and $(s, t, V, a) \in \mathrm{Conn}_\mathrm{p}$. We call a rule $q \to (s, t, V, a)$ a *q-rule*, of which q and (s, t, V, a) are the *left-hand* and *right-hand side*, respectively, and a is its *weight*. The STSG $\mathcal{G}$ is *input-productive* (respectively, *output-productive*) if each of its rules is so. To simplify the following development, we assume (without loss of generality) that two different q-rules differ on more than just their weight.[1]

To make sure that we do not account essentially the same derivation twice, we have to use a deterministic derivation mode. Since the choice is immaterial, we use the leftmost derivation mode for the output component t of a connected tree pair (s, t, V, a). For every $(s, t, V, a) \in \mathrm{Conn}$ such that $V \neq \emptyset$, the *leftmost output position* is the pair $(w, w') \in V$, where w' is the leftmost (i.e., the lexicographically smallest) position of $\mathrm{lv}_Q(t)$.

Next we define derivations. The *derivation relation induced by $\mathcal{G}$* is the binary relation $\Rightarrow_\mathcal{G}$ over Conn such that

$$\xi = (s_1, t_1, V_1, a_1) \Rightarrow_\mathcal{G} (s_2, t_2, V_2, a_2) = \zeta$$

if and only if the leftmost output position of ξ is $(w, w') \in V_1$ and there exists a rule

$$s_1(w) \to (s, t, V, a) \in \mathcal{G}$$

such that

- $s_2 = s_1[w \leftarrow s]$ and $t_2 = t_1[w' \leftarrow t]$,
- $V_2 = (V_1 \setminus \{(w, w')\}) \cup V'$ where $V' = \{(ww_1, w'w_2) \mid (w_1, w_2) \in V\}$, and
- $a_2 = a_1 \cdot a$.

A sequence $D = (\xi_1, \dots, \xi_n) \in \mathrm{Conn}^n$ is a *derivation* of $(s, t, V, a) \in \mathrm{Conn}$ from $q \in Q$ if

- $\xi_1 = (q, q, \{(\varepsilon, \varepsilon)\}, 1)$,
- $\xi_n = (s, t, V, a)$, and
- $\xi_i \Rightarrow_\mathcal{G} \xi_{i+1}$ for every $1 \leq i \leq n - 1$.

The set of all such derivations is denoted by $D^q_\mathcal{G}(s, t, V, a)$.

For every $q \in Q$, $s \in T_\Sigma(Q)$, $t \in T_\Delta(Q)$, and bijection $V\colon \mathrm{lv}_Q(s) \to \mathrm{lv}_Q(t)$, let

$$\tau^q_\mathcal{G}(s, t, V) = \sum_{a \in \mathbb{R}_{\geq 0}, D \in D^q_\mathcal{G}(s,t,V,a)} a \ .$$

[1]Formally, $q \to (s, t, V, a) \in G$ and $q \to (s, t, V, b) \in G$ implies $a = b$.

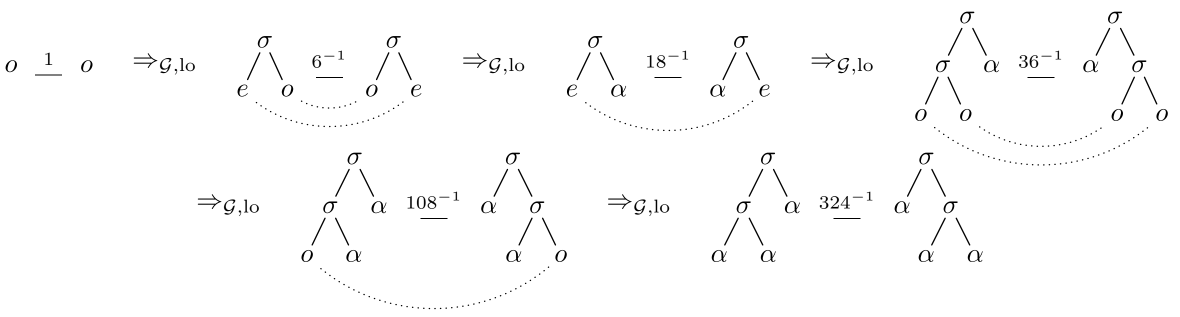

Figure 1: Example derivation with the STSG $\mathcal{G}$ of Example 1.

Finally, the *weighted tree transformation computed by* $\mathcal{G}$ is the weighted tree transformation $\tau_{\mathcal{G}} \colon T_\Sigma \times T_\Delta \to \mathbb{R}_{\geq 0}$ with $\tau_{\mathcal{G}}(s,t) = \tau_{\mathcal{G}}^{q_\mathrm{S}}(s,t,\emptyset)$ for every $s \in T_\Sigma$ and $t \in T_\Delta$. As usual, we call two STSG equivalent if they compute the same weighted tree transformation. We observe that every STSG is essentially a linear, nondeleting weighted extended top-down (or bottom-up) tree transducer (Arnold and Dauchet, 1982; Graehl et al., 2008; Engelfriet et al., 2009) without (both-sided) epsilon rules, and vice versa.

Example 1. *Let us consider the* STSG $\mathcal{G}$ *over* $\Sigma = \Delta = \{\sigma, \alpha\}$ *and* $Q = \{e, o\}$ *where* $q_\mathrm{S} = o$, $\mathrm{rk}(\sigma) = 2$, *and* $\mathrm{rk}(\alpha) = 0$. *The* STSG $\mathcal{G}$ *consists of the following rules where* $V = \{(1,2),(2,1)\}$ *and* $\mathrm{id} = \{(1,1),(2,2)\}$*:*

$$o \to (\sigma(o,e), \sigma(e,o), V, 1/3) \qquad (\rho_1)$$
$$o \to (\sigma(e,o), \sigma(o,e), V, 1/6) \qquad (\rho_2)$$
$$o \to (\sigma(e,o), \sigma(e,o), \mathrm{id}, 1/6) \qquad (\rho_3)$$
$$o \to (\alpha, \alpha, \emptyset, 1/3) \qquad (\rho_4)$$
$$e \to (\sigma(e,e), \sigma(e,e), V, 1/2) \qquad (\rho_5)$$
$$e \to (\sigma(o,o), \sigma(o,o), V, 1/2) \qquad (\rho_6)$$

Figure 1 shows a derivation induced by $\mathcal{G}$. *It can easily be checked that* $\tau_{\mathcal{G}}(s,t) = \frac{1}{6 \cdot 3 \cdot 2 \cdot 3 \cdot 3}$ *where* $s = \sigma(\sigma(\alpha,\alpha),\alpha)$ *and* $t = \sigma(\alpha, \sigma(\alpha,\alpha))$. *Moreover,* $\tau_{\mathcal{G}}(s,s) = \tau_{\mathcal{G}}(s,t)$. *If* $\tau_{\mathcal{G}}^q(s,t,\emptyset) \neq 0$ *with* $q \in \{e,o\}$, *then* s *and* t *have the same number of* α*-labeled leaves. This number is odd if* $q = o$, *otherwise it is even. Moreover, at every position* $w \in \mathrm{pos}(s)$, *the left and right subtrees* s_1 *and* s_2 *are interchanged in* s *and* t *(due to* V *in the rules* $\rho_1, \rho_2, \rho_5, \rho_6$*) except if* s_1 *and* s_2 *contain an even and odd number, respectively, of* α*-labeled leaves. In the latter case, the subtrees can be interchanged or left unchanged (both with probability* $1/6$*).*

4 Recognizable weighted tree languages

Next, we recall weighted regular tree grammars (Alexandrakis and Bozapalidis, 1987). To keep the presentation simple, we identify WRTG with particular STSG, in which the input and the output components are identical. More precisely, a *weighted regular tree grammar over* Σ *and* Q (for short: WRTG) is an STSG $\mathcal{G}$ over Σ, Σ, and Q where each rule has the form $q \to (s, s, \mathrm{id}, a)$ where id is the suitable (partial) identity mapping. It follows that $s \notin Q$, which yields that we do not have chain rules. In the rest of this paper, we will specify a rule $q \to (s, s, \mathrm{id}, a)$ of a WRTG simply by $q \xrightarrow{a} s$. For every $q \in Q$, we define the weighted tree language $\varphi_{\mathcal{G}}^q \colon T_\Sigma(Q) \to \mathbb{R}_{\geq 0}$ *generated by* $\mathcal{G}$ *from* q by $\varphi_{\mathcal{G}}^q(s) = \tau_{\mathcal{G}}^q(s, s, \mathrm{id}_{\mathrm{lv}_Q(s)})$ for every $s \in T_\Sigma(Q)$, where $\mathrm{id}_{\mathrm{lv}_Q(s)}$ is the identity on $\mathrm{lv}_Q(s)$. Moreover, the weighted tree language $\varphi_{\mathcal{G}} \colon T_\Sigma \to \mathbb{R}_{\geq 0}$ *generated by* $\mathcal{G}$ is defined by $\varphi_{\mathcal{G}}(s) = \varphi_{\mathcal{G}}^{q_\mathrm{S}}(s)$ for every $s \in T_\Sigma$.

A weighted tree language $\varphi \colon T_\Sigma \to \mathbb{R}_{\geq 0}$ is *recognizable* if there exists a WRTG $\mathcal{G}$ such that $\varphi = \varphi_{\mathcal{G}}$. We note that our notion of recognizability coincides with the classical one (Alexandrakis and Bozapalidis, 1987; Fülöp and Vogler, 2009).

Example 2. *We consider the* WRTG $\mathcal{K}$ *over the input alphabet* $\Sigma = \{\sigma, \alpha\}$ *and* $P = \{p, q\}$ *with* $q_\mathrm{S} = q$, $\mathrm{rk}(\sigma) = 2$, *and* $\mathrm{rk}(\alpha) = 0$. *The* WRTG $\mathcal{K}$ *contains the following rules:*

$$q \xrightarrow{0.4} \sigma(p, \alpha) \quad q \xrightarrow{0.6} \alpha \quad p \xrightarrow{1} \sigma(\alpha, q) \quad (\nu_1\text{–}\nu_3)$$

Let $s \in T_\Sigma$ *be such that* $\varphi_{\mathcal{K}}(s) \neq 0$. *Then* s *is a thin tree with zig-zag shape; i.e., there exists* $n \geq 1$ *such that* $\mathrm{pos}(s)$ *contains exactly the positions:*

- $(12)^i$ *for every* $0 \leq i \leq \lfloor \frac{n-1}{2} \rfloor$, *and*
- $(12)^i 1$, $(12)^i 2$, *and* $(12)^i 11$ *for every integer* $0 \leq i \leq \lfloor \frac{n-3}{2} \rfloor$.

The integer n *can be understood as the length of a derivation that derives* s *from* q. *Some example*

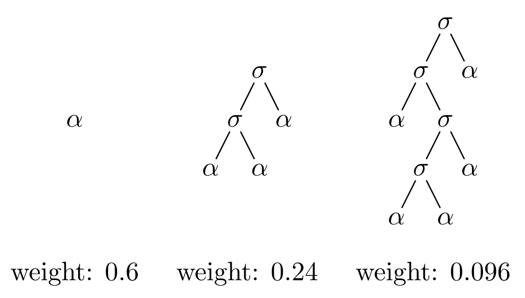

Figure 2: Example trees and their weight in $\varphi_{\mathcal{G}}$ where $\mathcal{G}$ is the WRTG of Example 2.

trees with their weights are displayed in Figure 2.

Proposition 3. *For every* WRTG $\mathcal{G}$ *there is an equivalent* WRTG $\mathcal{G}'$ *in* normal form*, in which the right-hand side of every rule contains exactly one symbol of* Σ.

Proof. We can obtain the statement by a trivial extension to the weighted case of the approach used in Lemma II.3.4 of (Gécseg and Steinby, 1984) and Section 6 of (Gécseg and Steinby, 1997). $\square$

5 STSG and weighted bimorphisms

In this section, we characterize the expressive power of STSG in terms of weighted bimorphisms. This will provide a conceptually clear pattern for the construction in our main result (see Theorem 6) concerning the closure of recognizable weighted tree languages under forward and backward application. For this we first recall tree homomorphisms. Let Γ and Σ be two ranked alphabets. Moreover, let $h\colon \Gamma \to T_\Sigma \times (\mathbb{N}^*)^*$ be a mapping such that $h(\gamma) = (s, u_1, \dots, u_k)$ for every $\gamma \in \Gamma_k$ where $s \in T_\Sigma$ and all leaves $u_1, \dots, u_k \in \mathrm{lv}(s)$ are pairwise different. The mapping h induces the (linear and complete) *tree homomorphism* $\widetilde{h}\colon T_\Gamma \to T_\Sigma$, which is defined by $\widetilde{h}(\gamma(d_1, \dots, d_k)) = s[u_1 \leftarrow \widetilde{d}_1, \dots, u_k \leftarrow \widetilde{d}_k]$ for every $\gamma \in \Gamma_k$ and $d_1, \dots, d_k \in T_\Gamma$ with $h(\gamma) = (s, u_1, \dots, u_k)$ and $\widetilde{d}_i = \widetilde{h}(d_i)$ for every $1 \leq i \leq k$. Moreover, every (linear and complete) tree homomorphism is induced in this way. In the rest of this paper we will not distinguish between h and $\widetilde{h}$ and simply write h instead of $\widetilde{h}$. The homomorphism h is *order-preserving* if $u_1 < \cdots < u_k$ for every $\gamma \in \Gamma_k$ where $h(\gamma) = (s, u_1, \dots, u_k)$. Finally, we note that every $\tau \in \mathrm{dREL}$ can be computed by a order-preserving tree homomorphism.

A *weighted bimorphism* $\mathcal{B}$ *over* Σ *and* Δ consists of a WRTG $\mathcal{K}$ over Γ and P and two tree homomorphisms

$$\begin{tikzcd}
T_\Gamma \arrow[r, "\varphi_{\mathcal{K}}"] \arrow[d, "(h_{\mathrm{in}}, h_{\mathrm{out}})"'] & \mathbb{R}_{\geq 0} \\
T_\Sigma \times T_\Delta \arrow[ur, bend right, "\tau_{\mathcal{B}}"'] &
\end{tikzcd}$$

Figure 3: Illustration of the semantics of the bimorphism $\mathcal{B}$.

$$h_{\mathrm{in}}\colon T_\Gamma \to T_\Sigma \quad \text{and} \quad h_{\mathrm{out}}\colon T_\Gamma \to T_\Delta \ .$$

The bimorphism $\mathcal{B}$ *computes* the weighted tree transformation $\tau_{\mathcal{B}}\colon T_\Sigma \times T_\Delta \to \mathbb{R}_{\geq 0}$ with

$$\tau_{\mathcal{B}}(s,t) = \sum_{d \in h_{\mathrm{in}}^{-1}(s) \cap h_{\mathrm{out}}^{-1}(t)} \varphi_{\mathcal{K}}(d)$$

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

Without loss of generality, we assume that every bimorphism $\mathcal{B}$ is presented by an WRTG $\mathcal{K}$ in normal form and an order-preserving output homomorphism h_{out}. Next, we prepare the relation between STSG and weighted bimorphisms. Let $\mathcal{G}$ be an STSG over Σ, Δ, and Q. Moreover, let $\mathcal{B}$ be a weighted bimorphism over Σ and Δ consisting of (i) $\mathcal{K}$ over Γ and P in normal form, (ii) h_{in}, and (iii) order-preserving h_{out}. We say that $\mathcal{G}$ and $\mathcal{B}$ are *related* if $Q = P$ and there is a bijection $\theta\colon \mathcal{G} \to \mathcal{K}$ such that, for every rule $\rho \in \mathcal{G}$ with $\rho = (q \to (s, t, V, a))$ and $\theta(\rho) = (p \overset{a}{\to} \gamma(p_1, \dots, p_k))$ we have

- $p = q$,
- $h_{\mathrm{in}}(\gamma) = (s, u_1, \dots, u_k)$,
- $h_{\mathrm{out}}(\gamma) = (t, v_1, \dots, v_k)$,
- $V = \{(u_1, v_1), \dots, (u_k, v_k)\}$, and
- $s(u_i) = p_i = t(v_i)$ for every $1 \leq i \leq k$.

Let $\mathcal{G}$ and $\mathcal{B}$ be related. The following three easy statements can be used to prove that $\mathcal{G}$ and $\mathcal{B}$ are equivalent:

1. For every derivation $D \in D^q_{\mathcal{G}}(s, t, \emptyset, a)$ with $q \in Q, s \in T_\Sigma, t \in T_\Delta, a \in \mathbb{R}_{\geq 0}$, there exists $d \in T_\Gamma$ and a derivation $D' \in D^q_{\mathcal{K}}(d, d, \emptyset, a)$ such that $h_{\mathrm{in}}(d) = s$ and $h_{\mathrm{out}}(d) = t$.
2. For every $d \in T_\Gamma$ and $D' \in D^q_{\mathcal{K}}(d, d, \emptyset, a)$ with $q \in Q$ and $a \in \mathbb{R}_{\geq 0}$, there exists a derivation $D \in D^q_{\mathcal{G}}(h_{\mathrm{in}}(d), h_{\mathrm{out}}(d), \emptyset, a)$.
3. The mentioned correspondence on derivations is a bijection.

Given an STSG $\mathcal{G}$, we can easily construct a weighted bimorphism $\mathcal{B}$ such that $\mathcal{G}$ and $\mathcal{B}$ are related, and vice versa. Hence, STSG and weighted bimorphisms are equally expressive, which generalizes the corresponding characterization result in the unweighted case by Shieber (2004), which we will state after the introduction of STSG$\downarrow$.

Classical synchronous tree substitution grammars (STSG$\downarrow$) do not have states. An STSG$\downarrow$ can be seen as an STSG by considering every substitution site (i.e., each pair of synchronised nonterminals) as a state.[2] We illustrate this by means of an example here. Let us consider the STSG$\downarrow$ $\mathcal{G}$ with the following rules:

- $(S(\alpha, B\downarrow), S(D\downarrow, \beta))$ with weight 0.2
- $(B(\gamma, B\downarrow), D(\delta, D\downarrow))$ with weight 0.3
- $(B(\alpha), D(\beta))$ with weight 0.4.

The substitution sites are marked with $\downarrow$. Any rule with root A can be applied to a substitution site $A\downarrow$. An equivalent STSG $\mathcal{G}'$ has the rules:

$$\begin{aligned}
&\langle S, S\rangle \to (S(\alpha, \langle B, D\rangle), S(\langle B, D\rangle, \beta), V, 0.2)\\
&\langle B, D\rangle \to (B(\gamma, \langle B, D\rangle), D(\delta, \langle B, D\rangle), V', 0.3)\\
&\langle B, D\rangle \to (B(\alpha), D(\beta), \emptyset, 0.4) \ ,
\end{aligned}$$

where $V = \{(2,1)\}$ and $V' = \{(2,2)\}$. It is easy to see that $\mathcal{G}$ and $\mathcal{G}'$ are equivalent.

Let $\Sigma = \{\gamma, \gamma', \gamma'', \alpha, \beta\}$ where $\gamma, \gamma', \gamma'' \in \Sigma_1$ and $\alpha, \beta \in \Sigma_0$ (and $\gamma' \neq \gamma''$ and $\alpha \neq \beta$). We write $\gamma^m(t)$ with $t \in T_\Sigma$ for the tree $\gamma(\cdots\gamma(t)\cdots)$ containing m occurrences of γ above t. STSG$\downarrow$ have a certain locality property, which yields that STSG$\downarrow$ cannot compute transformations like

$$\tau(s,t) = \begin{cases} 1 & \text{if } s = \gamma'(\gamma^m(\alpha)) = t \\ & \text{or } s = \gamma''(\gamma^m(\beta)) = t \\ 0 & \text{otherwise} \end{cases}$$

for every $s, t \in T_\Sigma$. The non-local feature is the correspondence between the symbols γ' and α (in the first alternative) and the symbols γ'' and β (in the second alternative). An STSG that computes τ is presented in Figure 4.

Theorem 4. *Let τ be a weighted tree transformation. Then the following are equivalent.*

1. *τ is computable by an* STSG.
2. *τ is computable by a weighted bimorphism.*
3. *There exists a* STSG$\downarrow$ *$\mathcal{G}$ and deterministic relabelings r_1 and r_2 such that*

$$\tau(s,t) = \sum_{s' \in r_1^{-1}(s), t' \in r_2^{-1}(t)} \tau_{\mathcal{G}}(s', t') \ .$$

[2]To avoid a severe expressivity restriction, several initial states are allowed for an STSG$\downarrow$.

The inverse of an STSG computable weighted tree transformation can be computed by an STSG. Formally, the *inverse* of the STSG $\mathcal{G}$ is the STSG

$$\mathcal{G}^{-1} = \{(t, s, V^{-1}, a) \mid (s, t, V, a) \in \mathcal{G}\}$$

where V^{-1} is the inverse of V. Then $\tau_{\mathcal{G}^{-1}} = \tau_{\mathcal{G}}^{-1}$.

6 Forward and backward application

Let us start this section with the definition of the concepts of forward and backward application of a weighted tree transformation $\tau\colon T_\Sigma \times T_\Delta \to \mathbb{R}_{\geq 0}$ to weighted tree languages $\varphi\colon T_\Sigma \to \mathbb{R}_{\geq 0}$ and $\psi\colon T_\Delta \to \mathbb{R}_{\geq 0}$. We will give general definitions first and deal with the potentially infinite sums later. The *forward application* of τ to φ is the weighted tree language $\tau(\varphi)\colon T_\Delta \to \mathbb{R}_{\geq 0}$, which is defined for every $t \in T_\Delta$ by

$$\big(\tau(\varphi)\big)(t) = \sum_{s \in T_\Sigma} \varphi(s) \cdot \tau(s,t) \ . \qquad (1)$$

Dually, the *backward application* of τ to ψ is the weighted tree language $\tau^{-1}(\psi)\colon T_\Sigma \to \mathbb{R}_{\geq 0}$, which is defined for every $s \in T_\Sigma$ by

$$\big(\tau^{-1}(\psi)\big)(s) = \sum_{t \in T_\Delta} \tau(s,t) \cdot \psi(t) \ . \qquad (2)$$

In general, the sums in Equations (1) and (2) can be infinite. Let us recall the important property that makes them finite in our theorems.

Proposition 5. *For every input-productive (resp., output-productive)* STSG *$\mathcal{G}$ and every tree $s \in T_\Sigma$ (resp., $t \in T_\Delta$), there exist only finitely many trees $t \in T_\Delta$ (respectively, $s \in T_\Sigma$) such that* $\tau_{\mathcal{G}}(s,t) \neq 0$.

Proof sketch. If $\mathcal{G}$ is input-productive, then each derivation step creates at least one input symbol. Consequently, any derivation for the input tree s can contain at most as many steps as there are nodes (or positions) in s. Clearly, there are only finitely many such derivations, which proves the statement. Dually, we can obtain the statement for output-productive STSG. $\square$

In the following, we will consider forward applications $\tau_{\mathcal{G}}(\varphi)$ where $\mathcal{G}$ is an output-productive STSG and φ is recognizable, which yields that (1) is well-defined by Proposition 5. Similarly, we consider backward applications $\tau_{\mathcal{G}}^{-1}(\psi)$ where $\mathcal{G}$ is input-productive and ψ is recognizable, which again yields that (2) is well-defined by Proposition 5. The question is whether $\tau_{\mathcal{G}}(\varphi)$ and $\tau_{\mathcal{G}}^{-1}(\psi)$

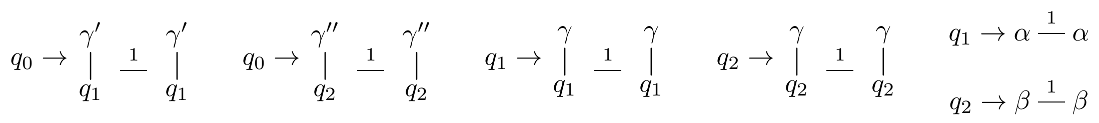
$q_0 \to \gamma'(q_1) \xrightarrow{1} \gamma'(q_1)$
$q_0 \to \gamma''(q_2) \xrightarrow{1} \gamma''(q_2)$
$q_1 \to \gamma(q_1) \xrightarrow{1} \gamma(q_1)$
$q_2 \to \gamma(q_2) \xrightarrow{1} \gamma(q_2)$
$q_1 \to \alpha \xrightarrow{1} \alpha$
$q_2 \to \beta \xrightarrow{1} \beta$

Figure 4: STSG computing the weighted tree transformation τ with initial state q_0.

are again recognizable. To avoid confusion, we occasionally use angled parentheses as in $\langle p, q\rangle$ instead of standard parentheses as in (p, q). Moreover, for ease of presentation, we identify the initial state q_S with $\langle q_S, q_S\rangle$.

Theorem 6. *Let $\mathcal{G}$ be an* STSG *over Σ, Δ, and Q. Moreover, let $\varphi\colon T_\Sigma \to \mathbb{R}_{\geq 0}$ and $\psi\colon T_\Delta \to \mathbb{R}_{\geq 0}$ be recognizable weighted tree languages.*

1. *If $\mathcal{G}$ is output-productive, then $\tau_{\mathcal{G}}(\varphi)$ is recognizable.*
2. *If $\mathcal{G}$ is input-productive, then $\tau_{\mathcal{G}}^{-1}(\psi)$ is recognizable.*

Proof. For the first item, let $\mathcal{K}$ be a WRTG over Σ and P such that $\varphi = \varphi_{\mathcal{K}}$. Without loss of generality, we suppose that $\mathcal{K}$ is in normal form.

Intuitively, we take each rule $q \to (s, t, V, a)$ of $\mathcal{G}$ and run the WRTG $\mathcal{K}$ with every start state p on the input side s of the rule. In this way, we obtain a weight b. The WRTG will reach the state leaves of s in certain states, which we then transfer to the linked states in t to obtain t'. Finally, we remove the input side and obtain a rule $\langle p, q\rangle \overset{ab}{\to} t'$ for the WRTG $\mathcal{L}$ that represents the forward application. We note that the same rule of $\mathcal{L}$ might be constructed several times. If this happens, then we replace the several copies by one rule whose weight is the sum of the weights of all copies. As already mentioned the initial state is $\langle q_S, q_S\rangle$. Clearly, this approach is inspired (and made reasonable) by the bimorphism characterization. We can take the HADAMARD product of the WRTG of the bimorphism with the inverse image of $\varphi_{\mathcal{K}}$ under its input homomorphism. Then we can simply project to the output side. Our construction performs those three steps at once. The whole process is illustrated in Figure 5.

Formally, we construct the WRTG $\mathcal{L}$ over Δ and $P \times Q$ with the following rules. Let $p \in P$, $q \in Q$, and $t' \in T_\Delta(P \times Q)$. Then $\langle p, q\rangle \overset{c}{\to} t'$ is a rule in $\mathcal{L}'$, where

$$c = \sum_{\substack{(q \to (s,t,V,a)) \in \mathcal{G} \\ V = \{(u_1,v_1),\dots,(u_k,v_k)\} \\ p_1,\dots,p_k \in P \\ t' = t[v_i \leftarrow \langle p_i, t(v_i)\rangle \mid 1 \leq i \leq k] \\ b = \varphi_{\mathcal{K}}^{p}(s[u_i \leftarrow p_i \mid 1 \leq i \leq k])}} ab \enspace .$$

This might create infinitely many rules in $\mathcal{L}'$, but clearly only finitely many will have a weight different from 0. Thus, we can obtain the finite rule set $\mathcal{L}$ by removing all rules with weight 0.

The main statement to prove is the following: for every $t \in T_\Delta(Q)$ with $\mathrm{lv}_Q(t) = \{v_1, \dots, v_k\}$, $p, p_1, \dots, p_k \in P$, and $q \in Q$

$$\sum_{\substack{s \in T_\Sigma(Q) \\ u_1,\dots,u_k \in \mathrm{lv}_Q(s)}} \varphi_{\mathcal{K}}^{p}(s') \cdot \tau_{\mathcal{G}}^{q}(s, t, V) = \varphi_{\mathcal{L}}^{\langle p,q\rangle}(t') \enspace ,$$

where

- $V = \{(u_1, v_1), \dots, (u_k, v_k)\}$,
- $s' = s[u_i \leftarrow p_i \mid 1 \leq i \leq k]$, and
- $t' = t[v_i \leftarrow \langle p_i, t(v_i)\rangle \mid 1 \leq i \leq k]$.

In particular, for $t \in T_\Delta$ we obtain

$$\sum_{s \in T_\Sigma} \varphi_{\mathcal{K}}^{p}(s) \cdot \tau_{\mathcal{G}}^{q}(s, t, \emptyset) = \varphi_{\mathcal{L}}^{\langle p,q\rangle}(t) \enspace ,$$

which yields

$$\begin{aligned}\big(\tau_{\mathcal{G}}(\varphi_{\mathcal{K}})\big)(t) &= \sum_{s \in T_\Sigma} \varphi_{\mathcal{K}}(s) \cdot \tau_{\mathcal{G}}(s, t) \\ &= \sum_{s \in T_\Sigma} \varphi_{\mathcal{K}}^{q_S}(s) \cdot \tau_{\mathcal{G}}^{q_S}(s, t, \emptyset) \\ &= \varphi_{\mathcal{L}}^{\langle q_S, q_S\rangle}(t) = \varphi_{\mathcal{L}}(t) \enspace .\end{aligned}$$

In the second item $\mathcal{G}$ is input-productive. Then $\mathcal{G}^{-1}$ is output-productive and $\tau_{\mathcal{G}}^{-1}(\psi) = \tau_{\mathcal{G}^{-1}}(\psi)$. Hence the first statement proves that $\tau_{\mathcal{G}}^{-1}(\psi)$ is recognizable. □

Example 7. *As an illustration of the construction in Theorem 6, let us apply the* STSG *$\mathcal{G}$ of Example 1 to the* WRTG *$\mathcal{K}$ over Σ and $P = \{p, q_S, q_\alpha\}$ and the following rules:*

$$q_S \overset{\frac{2}{5}}{\to} \sigma(p, q_\alpha) \qquad q_S \overset{\frac{3}{5}}{\to} \alpha$$
$$p \overset{1}{\to} \sigma(q_\alpha, q_S) \qquad q_\alpha \overset{1}{\to} \alpha \enspace .$$

In fact, $\mathcal{K}$ is in normal form and is equivalent to the WRTG *of Example 2. Using the construction in the proof of Theorem 6 we obtain the* WRTG *$\mathcal{L}$ over Σ and $P \times Q$ with $Q = \{e, o\}$. We will only*

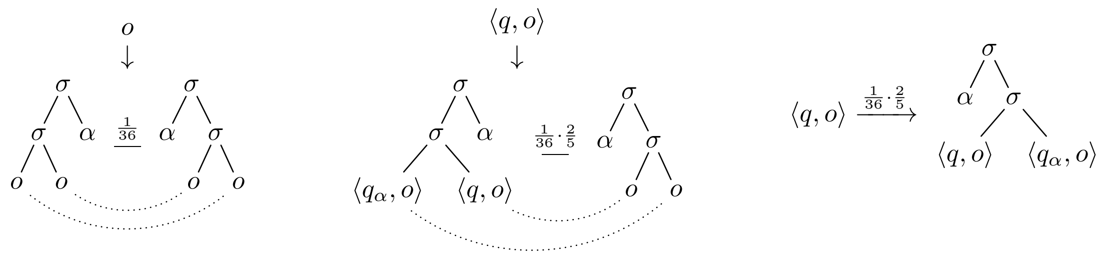

Figure 5: Illustration of the construction in the proof of Theorem 6 using the WRTG $\mathcal{K}$ of Example 7: some example rule (left), run of $\mathcal{K}$ on the input side of the rule (middle), and resulting rule (right).

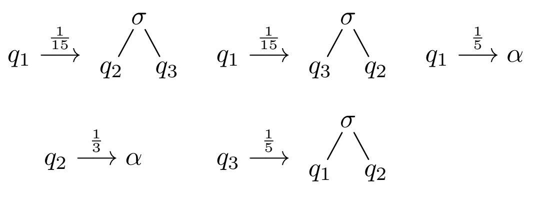

Figure 6: WRTG constructed in Example 7. We renamed the states and calculated the weights.

show rules of $\mathcal{L}$ that contribute to $\varphi_{\mathcal{L}}$. To the right of each rule we indicate from which state of $\mathcal{K}$ and which rule of $\mathcal{G}$ the rule was constructed.

$$\begin{aligned}
\langle q_S, o\rangle &\xrightarrow{\frac{1}{6}\cdot\frac{2}{5}} \sigma(\langle q_\alpha, o\rangle, \langle p, e\rangle) && q_S, \rho_2 \\
\langle q_S, o\rangle &\xrightarrow{\frac{1}{6}\cdot\frac{2}{5}} \sigma(\langle p, e\rangle, \langle q_\alpha, o\rangle) && q_S, \rho_3 \\
\langle q_S, o\rangle &\xrightarrow{\frac{1}{3}\cdot\frac{3}{5}} \alpha && q_S, \rho_4 \\
\langle q_\alpha, o\rangle &\xrightarrow{\frac{1}{3}\cdot 1} \alpha && q_\alpha, \rho_4 \\
\langle p, e\rangle &\xrightarrow{\frac{1}{2}\cdot\frac{2}{5}} \sigma(\langle q_S, o\rangle, \langle q_\alpha, o\rangle) && p, \rho_6
\end{aligned}$$

The initial state of $\mathcal{L}$ is $\langle q_S, o\rangle$. It is easy to see that every $t \in T_\Sigma$ such that $\varphi_{\mathcal{L}}(t) \neq 0$ is thin, which means that $|\mathrm{pos}(t) \cap \mathbb{N}^n| \leq 2$ for every $n \in \mathbb{N}$.

7 Domain and range

Finally, let us consider the domain and range of a weighted tree transformation $\tau \colon T_\Sigma \times T_\Delta \to \mathbb{R}_{\geq 0}$. Again, we first give general definitions and deal with the infinite sums that might occur in them later. The *domain* $\mathrm{dom}(\tau)$ of τ and the *range* $\mathrm{range}(\tau)$ of τ are defined by

$$\big(\mathrm{dom}(\tau)\big)(s) = \sum_{u \in T_\Delta} \tau(s, u) \qquad (3)$$

$$\big(\mathrm{range}(\tau)\big)(t) = \sum_{u \in T_\Sigma} \tau(u, t) \qquad (4)$$

for every $s \in T_\Sigma$ and $t \in T_\Delta$. Obviously, the domain $\mathrm{dom}(\tau)$ is the range $\mathrm{range}(\tau^{-1})$ of the inverse of τ. Moreover, we can express the domain $\mathrm{dom}(\tau)$ of τ as the backward application $\tau^{-1}(\underline{1})$ where $\underline{1}$ is the weighted tree language that assigns the weight 1 to each tree. Note that $\underline{1}$ is recognizable for every ranked alphabet.

We note that the sums in Equations (3) and (4) might be infinite, but for input-productive (respectively, output-productive) STSG $\mathcal{G}$ the domain $\mathrm{dom}(\tau_{\mathcal{G}})$ (respectively, the range $\mathrm{range}(\tau_{\mathcal{G}})$) are well-defined by Proposition 5. Using those observations and Theorem 6 we can obtain the following statement.

Corollary 8. *Let $\mathcal{G}$ be an* STSG. *If $\mathcal{G}$ is input-productive, then $\mathrm{dom}(\tau_{\mathcal{G}})$ is recognizable. Moreover, if $\mathcal{G}$ is output-productive, then $\mathrm{range}(\tau_{\mathcal{G}})$ is recognizable.*

Proof. These statements follow directly from Theorem 6 with the help of the observation that $\mathrm{dom}(\tau_{\mathcal{G}}) = \tau_{\mathcal{G}}^{-1}(\underline{1})$ and $\mathrm{range}(\tau_{\mathcal{G}}) = \tau_{\mathcal{G}}(\underline{1})$. ☐

Conclusion

We showed that every output-productive STSG preserves recognizability under forward application. Dually, every input-productive STSG preserves recognizability under backward application. We presented direct and effective constructions for these operations. Special cases of those constructions can be used to compute the domain of an input-productive STSG and the range of an output-productive STSG. Finally, we presented a characterization of the power of STSG in terms of weighted bimorphisms.

Acknowledgements

ZOLTÁN FÜLÖP and HEIKO VOGLER were financially supported by the TÁMOP-4.2.2/08/1/2008-0008 program of the Hungarian National Development Agency. ANDREAS MALETTI was financially supported by the *Ministerio de Educación y Ciencia* (MEC) grant JDCI-2007-760.

References

Anne Abeillé, Yves Schabes, and Aravind K. Joshi. 1990. Using lexicalized TAGs for machine translation. In *Proc. 13th CoLing*, volume 3, pages 1–6. University of Helsinki, Finland.

Alfred V. Aho and Jeffrey D. Ullman. 1969. Translations on a context-free grammar. In *Proc. 1st STOC*, pages 93–112. ACM.

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 bimorphismes d'arbres. *Theoret. Comput. Sci.*, 20(1):33–93.

Joost Engelfriet, Eric Lilin, and Andreas Maletti. 2009. Extended multi bottom-up tree transducers — composition and decomposition. *Acta Inform.*, 46(8):561–590.

Zoltán Fülöp and Heiko Vogler. 2009. Weighted tree automata and tree transducers. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, chapter 9, pages 313–403. Springer.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. HLT-NAACL 2004*, pages 273–280. ACL.

Ferenc Gécseg and Magnus Steinby. 1984. *Tree Automata*. Akadémiai Kiadó, Budapest, Hungary.

Ferenc Gécseg and Magnus Steinby. 1997. Tree languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, chapter 1, pages 1–68. Springer.

Jonathan S. Golan. 1999. *Semirings and their Applications*. Kluwer Academic.

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

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

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proc. 9th IWPT*, pages 53–64. ACL.

Kevin Knight and Jonathan Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Proc. 6th CICLing*, volume 3406 of LNCS, pages 1–24. Springer.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT-NAACL 2003*, pages 48–54. ACL.

Philip M. Lewis II and Richard Edwin Stearns. 1968. Syntax-directed transductions. *J. ACM*, 15(3):465–488.

Eric Lilin. 1981. Propriétés de clôture d'une extension de transducteurs d'arbres déterministes. In *Proc. 6th CAAP*, volume 112 of LNCS, pages 280–289. Springer.

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

Andreas Maletti. 2010. Why synchronous tree substitution grammars? In *Proc. HLT-NAACL 2010*. ACL. to appear.

David F. Martin and Steven A. Vere. 1970. On syntax-directed transduction and tree transducers. In *Proc. 2nd STOC*, pages 129–135. ACM.

Mark-Jan Nederhof and Giorgio Satta. 2006. Probabilistic parsing strategies. *J. ACM*, 53(3):406–436.

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

Yves Schabes. 1990. *Mathematical and computational aspects of lexicalized grammars*. Ph.D. thesis, University of Pennsylvania.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proc. 13th CoLing*, pages 253–258. ACL.

Stuart M. Shieber. 2004. Synchronous grammars as tree transducers. In *Proc. TAG+7*, pages 88–95. Simon Fraser University.

Stuart M. Shieber. 2006. Unifying synchronous tree adjoining grammars and tree transducers via bimorphisms. In *Proc. 11th EACL*, pages 377–384. ACL.

James W. Thatcher. 1970. Generalized^2 sequential machine maps. *J. Comput. System Sci.*, 4(4):339–367.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. 39th ACL*, pages 523–530. ACL.