Natural Language Engineering **17** (2): 221–242. © Cambridge University Press 2011
doi:10.1017/S1351324911000027

*An alternative to synchronous tree substitution grammars**

ANDREAS MALETTI†

Universitat Rovira i Virgili, Departament de Filologies Romàniques Avinguda de Catalunya 35, 43002 Tarragona, Spain
email: andreas.maletti@urv.cat

(*Received 15 June 2010; accepted 1 September 2010*)

Abstract

Synchronous tree substitution grammars (STSG) are a (formal) tree transformation model that is used in the area of syntax-based machine translation. A competitor that is at least as expressive as STSG is proposed and compared to STSG. The competitor is the extended multi bottom-up tree transducer (MBOT), which is the bottom-up analogue with the additional feature that states have non-unary ranks. Unweighted MBOT have already been investigated with respect to their basic properties, but the particular properties of the constructions that are required in the machine translation task are largely unknown. STSG and MBOT are compared with respect to binarization, regular restriction, and application. Particular attention is paid to the complexity of the constructions.

1 Introduction

Machine translation is a subfield of natural language processing. Every machine translation system uses a translation model, which is a formal model that describes the translation process. Such systems can be hand-crafted (in rule-based translation systems) or trained with the help of statistical processes. Automatically trainable translation models are discussed in Brown *et al.* (1990). The IBM models of Brown *et al.* (1993) are string-based in the sense that they base the translation decision on the words and the surrounding context. In the field of syntax-based machine translation, the translation models have access to the syntax (in the form of parse trees) of the sentences. A good exposition to both fields is presented in Knight (2007).

In this paper, we focus on syntax-based translation models, and in particular, synchronous tree substitution grammars (STSG), or the (essentially) equally powerful (linear and nondeleting) extended top-down tree transducers of Graehl, Knight and May (2008). A good introduction to STSG, which originate from the syntax-directed

*This is an extended and revised version of [A. Maletti. *Why synchronous tree substitution grammars?* In Proc. NAACL, 2010].

† The author was financially supported by the *Ministerio de Educación y Ciencia* (MEC) grants JDCI-2007-760 and MTM-2007-63422.

translation schemes of Aho and Ullman (1972) [nowadays more commonly known as synchronous context-free grammars] is presented in Chiang and Knight (2006). Roughly speaking, an STSG has rules in which a nonterminal is replaced by two trees containing terminal and nonterminal symbols. In addition, the nonterminals in the two trees are linked and a rule is only applied to such linked nonterminals.

Several algorithms for STSG have been discussed in the literature. For example, we can

- train them (Graehl *et al.* 2008),
- attempt to binarize them using the methods of Zhang *et al.* (2006) or Huang *et al.* (2009) or DeNero, Pauls and Klein (2009), or
- parse them (DeNero *et al.* 2009).

However, some important algorithms are partial because it is known that the construction is not possible in general. This is the case, for example, for binarization and composition.

Alternative models have been explored in the literature. One such alternative is the multi bottom-up tree transducer (MBOT) of Arnold and Dauchet (1982), Lilin (1981), and Engelfriet, Lilin and Maletti (2009), which essentially is the bottom-up analogue of STSG with the additional feature that nonterminals can have an arbitrary rank (the rank of a nonterminal of an STSG can be considered to be fixed to 1). This model is more expressive than STSG, but offers good computational properties. In this contribution, we will compare STSG and MBOT with respect to some standard algorithms. Generally, MBOT offer algorithmic benefits over STSG, which can be summarized as follows:

- Every STSG can be transformed into an equivalent MBOT in linear time.
- MBOT can be binarized in linear time whereas only partial binarizations (or asynchronous binarizations) are possible for STSG.
- The input language of an MBOT M can be regularly restricted in $\mathcal{O}(|M| \cdot |P|^3)$, whereas the corresponding BAR-HILLEL construction for an STSG M runs in time $\mathcal{O}(|M| \cdot |P|^{2rk(M)+5})$ where $rk(M)$ is the maximal number of nonterminals in a rule of the STSG M and P are the states of the restricting string automaton.
- The output language of an MBOT M can be regularly restricted in time $\mathcal{O}(|M| \cdot |P|^{2rk(M)+2})$, whereas the corresponding BAR-HILLEL construction for an STSG M runs in time $\mathcal{O}(|M| \cdot |P|^{2rk(M)+5})$.

Overall, we thus conclude that, from an algorithmic perspective, it would be beneficial to work with MBOT instead of STSG. However, the full power of MBOT should not be tapped because, in general, MBOT are not symmetric and have the finite-copying property (Engelfriet, Rozenberg and Slutzki 1980), which complicates the algorithms for forward and backward application (see Section 6) and makes the forward application partial. An implementation and experimental verification of these advantages is in preparation.

2 Preliminaries

The set of nonnegative integers is $\mathbb{N}$, and we let $[n] = \{i \in \mathbb{N} \mid 1 \leq i \leq n\}$ for every $n \in \mathbb{N}$. An alphabet is a finite set Σ of symbols. The set of all strings over Σ is Σ^*, of which the empty string is ε. Concatenation of strings is denoted by juxtaposition. For each $w \in \Sigma^*$, the length $|w|$ of w is the number of occurrences of symbols in w. The ith letter in w with $1 \leq i \leq |w|$ is denoted by w_i.

A commutative semiring (Hebisch and Weinert 1998; Golan 1999) is an algebraic structure $(A, +, \cdot, 0, 1)$ such that

- $(A, +, 0)$ and $(A, \cdot, 1)$ are commutative monoids, and
- the multiplication $\cdot$ distributes over finite sums (in particular, $a \cdot 0 = 0 = 0 \cdot a$ for every $a \in A$).

Examples of commutative semirings are

- the real number semiring $(\mathbb{R}, +, \cdot, 0, 1)$,
- the BOOLEAN semiring $(\{0, 1\}, \max, \min, 0, 1)$, and
- the tropical semiring $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$.

In addition, every commutative field or ring is a commutative semiring.

In the rest of the paper, let $(A, +, \cdot, 0, 1)$ be a commutative semiring. For every mapping $f : S \to A$, we let $supp(f) = \{s \in S \mid f(s) \neq 0\}$ be the support of f. As usual, we write $\sum_{s \in S} f(s)$ for the sum of the elements $f(s)$ for every $s \in S$ provided that $supp(f)$ is finite. We also write $\prod_{s \in S} f(s)$ for the product of the elements $f(s)$ for every $s \in S$ provided that S is finite.

A weighted string automaton (WSA) (Schützenberger 1961; Eilenberg 1974) is a system $N = (P, \Gamma, J, v, G)$, where

- P and Γ are alphabets of states and input symbols, respectively,
- $J, G : P \to A$ assign initial and final weights, respectively, and
- $v : P \times \Gamma \times P \to A$ assigns a weight to each transition.

Let $w = \gamma_1 \cdots \gamma_k \in \Gamma^*$ be an input string with $\gamma_i \in \Gamma$ for every $i \in [k]$. Every mapping $r : [k + 1] \to P$ is a run on w. The set of all runs on w is denoted by $Run_N(w)$, and we write r_i instead of $r(i)$ for every $r \in Run_N(w)$ and $i \in [k + 1]$. The weight $wt_N(r)$ of the run $r \in Run_N(w)$ is $\prod_{i=1}^{k} v(r_i, \gamma_i, r_{i+1})$. Finally, the semantics of the WSA N assigns to w the weight

$$N(w) = \sum_{r \in Run_N(w)} J(r_1) \cdot wt_N(r) \cdot G(r_{k+1}).$$

A good introduction to WSA can be found in Mohri (2009) or Sakarovitch (2009).

A ranked alphabet $\Sigma = \bigcup_{k \in \mathbb{N}} \Sigma_k$ is an alphabet whose symbols have assigned ranks. Contrary to some of the standard literature, we do not assume that each symbol of a ranked alphabet has only one fixed rank. For every $k \in \mathbb{N}$, the set Σ_k contains all symbols of rank k. For a given set T, we let $\Sigma(T) = \{\sigma(t_1, \ldots, t_k) \mid \sigma \in \Sigma_k, t_1, \ldots, t_k \in T\}$. The set $T_\Sigma(V)$ of Σ-trees indexed by a set V is the smallest set T such that $V \subseteq T$ and $\Sigma(T) \subseteq T$. For every $t \in T_\Sigma(V)$ and $S \subseteq \Sigma \cup V$, let

$pos_S(t) \subseteq \mathbb{N}^*$ be the set of positions labeled by elements of S. Formally, for every $v \in V$, $\sigma \in \Sigma_k$, and $t_1, \dots, t_k \in T_\Sigma(V)$

$$pos_S(v) = \begin{cases} \{\varepsilon\} & \text{if } v \in S \\ \emptyset & \text{otherwise} \end{cases}$$

$$pos_S(\sigma(t_1, \dots, t_k)) = \begin{cases} \{\varepsilon\} \cup \{iw \mid 1 \le i \le k, w \in pos_S(t_i)\} & \text{if } \sigma \in S \\ \{iw \mid 1 \le i \le k, w \in pos_S(t_i)\} & \text{otherwise.} \end{cases}$$

Clearly, positions are lexicographically ordered. We denote this order simply by $\le$. Moreover, let $|t|_S = |pos_S(t)|$ be the number of occurrences of symbols of S in t. If $S = \{s\}$, then we write $pos_s(t)$ and $|t|_s$ instead of $pos_S(t)$ and $|t|_S$, respectively. The positions $pos(t)$ and the size $|t|$ of t are $pos_{\Sigma \cup V}(t)$ and $|t|_{\Sigma \cup V}$, respectively. Moreover, let $var(t) = \{v \in V \mid pos_v(t) \neq \emptyset\}$. A tree $t \in T_\Sigma(V)$ is linear in V if $|pos_v(t)| \le 1$ for every $v \in V$.

Let $S \subseteq \Sigma_0 \cup V$. The *$S$-yield* $yd_S(t)$ of a tree $t \in T_\Sigma(V)$ is recursively defined as follows:

$$yd_S(s) = \begin{cases} s & \text{if } s \in S \\ \varepsilon & \text{otherwise} \end{cases}$$

$$yd_S(\sigma(t_1, \dots, t_k)) = yd_S(t_1) \cdots yd_S(t_k)$$

for every $s \in \Sigma_0 \cup V$, $\sigma \in \Sigma_k$ with $k \ge 1$, and $t_1, \dots, t_k \in T_\Sigma(V)$. If $S = \Sigma_0 \cup V$, then we simply write $yd(t)$ instead of $yd_S(t)$.

Let $\square$ be a distinguished symbol that has only the rank 0. Let Δ be the ranked alphabet such that $\Delta_0 = \Sigma_0 \cup \{\square\}$ and $\Delta_k = \Sigma_k$ for every $k \ge 1$. A Σ-context c indexed by V is a tree of $T_\Delta(V)$ such that $|c|_\square = 1$. The set of all Σ-contexts indexed by V is $C_\Sigma(V)$. The tree $c[t]$ is obtained from c by replacing the symbol $\square$ by t. More generally, $t[u]_w$ denotes the result of replacing the subtree at position $w \in pos(t)$ in $t \in T_\Sigma(V)$ by the tree $u \in T_\Sigma(V)$. In particular, $c[t] = c[t]_w$ where w is the unique element in $pos_\square(c)$. Finally, any mapping $\theta : V' \to T_\Sigma(V)$ with $V' \subseteq V$ is a substitution. The application $t\theta$ of the substitution θ to a tree $t \in T_\Sigma(V)$ is defined by

$$v\theta = \begin{cases} \theta(v) & \text{if } v \in V' \\ v & \text{otherwise} \end{cases}$$

$$\sigma(t_1, \dots, t_k)\theta = \sigma(t_1\theta, \dots, t_k\theta)$$

for every $v \in V$, $\sigma \in \Sigma_k$, and $t_1, \dots, t_k \in T_\Sigma(V)$. We often use elements of the fixed set $X = \{x_1, x_2, \dots\}$ of (formal) variables as substitution variables. If we write x_i, then we implicitly assume that $i \ge 1$. A tree $t \in T_\Sigma(X)$ is k-normed if $yd_X(t) = x_1 \cdots x_k$. Given $t \in T_\Sigma$, we let

$$match(t) = \{(l, x_1\theta, \dots, x_k\theta) \mid l \in T_\Sigma(X) \text{ } k\text{-normed}, \theta : var(l) \to T_\Sigma, t = l\theta\},$$

which is finite.

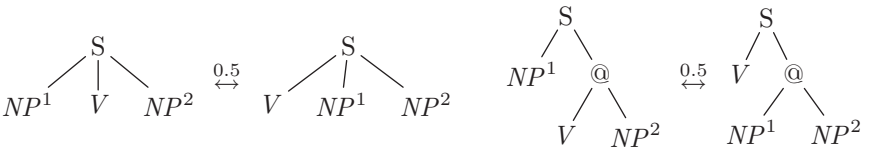

Fig. 1. STSG rule and the same rule with binarized trees where @ is an arbitrary symbol that is introduced in the binarization process. We could (without harm) add another @-symbol to represent the children of S in a list.

A weighted synchronous tree substitution grammar (STSG) (Chiang and Knight 2006) is a system $M = (Q, \Sigma, \Delta, I, R)$ where

- Q is an alphabet of nonterminals such that $Q \cap (\Sigma \cup \Delta) = \emptyset$,
- Σ and Δ are ranked alphabets of input and output symbols, respectively,
- $I : Q \to A$ assigns initial weights, and
- $R : T_\Sigma(Q) \times Q \times T_\Delta(Q) \to A$ are weighted rules such that $supp(R)$ is finite and for every $(t, q, u) \in supp(R)$
  - t and u are linear in Q,
  - $var(t) = var(u)$, and
  - $t \notin Q$ or $u \notin Q$.

In a rule $(t, q, u) \in supp(R)$, the tree t is the left-hand side, q is the target state, and u is the right-hand side. The first two restrictions on rules ensure that exactly the same states occur in t and u. Moreover, no state is allowed to occur twice in the left- or right-hand side. Intuitively, the links between the states are implicit by the assumption that equal states in the left- and right-hand side are linked. A sample STSG rule is displayed in Figure 1, where the nonterminals are slanted, and we assume that the root terminal is also the nonterminal.

The distinction between nonterminals and terminals is uncommon for STSG (Chiang 2005), but it increases the generative power. Our STSG are equivalent to the (nondeleting and linear) extended top-down tree transducers of Graehl *et al.* (2008) and Maletti *et al.* (2009). The size $|(t, q, u)|$ of a rule $(t, q, u) \in supp(R)$ is $|t| + |u|$, and the size $|M|$ of the STSG M is $\sum_{\rho \in supp(R)} |\rho|$. The STSG M is a weighted synchronous context-free grammar (SCFG) if $t \in \Sigma(Q)$ and $u \in \Delta(Q)$ for every rule $(t, q, u) \in supp(R)$. Finally, M is a weighted tree substitution grammar (TSG) if $t = u$ for all rules $(t, q, u) \in supp(R)$. A detailed exposition to STSG, SCFG, and TSG can be found in Chiang (2005), Berstel and Reutenauer (1982), or Fülöp and Vogler (2009).

Let us proceed with the semantics of an STSG $M = (Q, \Sigma, \Delta, I, R)$. Equal nonterminals in t and u of a rule $(t, q, u) \in supp(R)$ are *linked*. Those links need to be remembered in sentential forms. Given $t \in T_\Sigma(Q)$, $u \in T_\Delta(Q)$, and $\ell : pos_Q(t) \to pos_Q(u)$, the triple (t, ℓ, u) is a *sentential form* if

- ℓ is a bijection and
- $\ell(w) \in pos_q(u)$ for every $q \in Q$ and $w \in pos_q(t)$.

In other words, ℓ is a bijection that respects the nonterminals. Given two sentential forms $\xi = (t_1, \ell_1, u_1)$ and $\zeta = (t_2, \ell_2, u_2)$, we write $\xi \overset{a}{\Rightarrow}_M \zeta$ if there is a rule $(t, q, u) \in supp(R)$ such that

- $t_1 = t_1[q]_{w_1}$ and $u_1 = u_1[q]_{w_2}$,
- $t_2 = t_1[t]_{w_1}$ and $u_2 = u_1[u]_{w_2}$,
- $a = R(t, q, u)$, and
- $\ell_2 = \big(\ell_1 \setminus \{(w_1, w_2)\}\big) \cup \{(w_1 w', w_2 w'') \mid q \in Q, w' \in pos_q(t), w'' \in pos_q(u)\}$,

where $w_1 = \min(pos_Q(t_1))$ and $w_2 = \ell_1(w_1)$. The selection of the minimal state-labeled position in t_1 ensures that we obtain a left-most derivation with respect to the input side. A sequence $D = (\xi_0, \dots, \xi_k)$ of sentential forms is a *derivation* if there exist $a_1, \dots, a_k \in A$ such that $\xi_{i-1} \overset{a_i}{\Rightarrow}_M \xi_i$ for every $i \in [k]$. Note that if such weights $a_1, \dots, a_k$ exist, then they are unique. Consequently, we define the weight $wt_M(D)$ of the derivation D to be $wt_M(D) = \prod_{i=1}^{k} a_i$. Moreover, for every $q \in Q$ and sentential form $\zeta = (t, \ell, u)$ we let

$$D_M^q(\zeta) = \{D \mid D = (\xi_0, \dots, \xi_{k-1}, \zeta) \text{ is a derivation with } \xi_0 = (q, \{(\varepsilon, \varepsilon)\}, q)\}.$$

Note that the set $D_M^q(\zeta)$ is finite because each derivation step creates an input or an output symbol by the final condition on rules in the definition of STSG. The q-weight $M_q(\zeta)$ assigned by the STSG M to ζ is $M_q(\zeta) = \sum_{D \in D_M^q(\zeta)} wt_M(D)$ and $M(t, u) = \sum_{q \in Q} I(q) \cdot M_q(t, \emptyset, u)$ for every $t \in T_\Sigma$ and $u \in T_\Delta$.

For a TSG $M = (Q, \Sigma, \Sigma, F, R)$, we will simply write $R(q \to t)$ instead of $R(t, q, t)$ for every $q \in Q$ and $t \in T_\Sigma(Q)$. Moreover, we write $M_q(t)$ and $M(t)$ instead of $M_q(t, \emptyset, t)$ and $M(t, t)$ for every $q \in Q$ and $t \in T_\Sigma$. Finally, we drop the linearity requirement for TSG; i.e., t need not be linear in Q for a rule $(q \to t) \in supp(R)$. A mapping $\varphi : T_\Sigma \to A$ is recognizable (Fülöp and Vogler 2009) if there exists a TSG N such that $\varphi = N$. The following proposition is well-known (Alexandrakis and Bozapalidis 1987).

Proposition 1
For every TSG there exists an equivalent TSG that is also a SCFG.

While the derivation semantics of an STSG $M = (Q, \Sigma, \Delta, I, R)$ is very instructive, it is difficult to handle in formal proofs. We observe the following equality:

$$M_q(t, \emptyset, u) = \sum_{\substack{(t', q, u') \in supp(R) \\ \theta : var(t') \to T_\Sigma, t = t'\theta \\ \theta' : var(u') \to T_\Delta, u = u'\theta'}} R(t', q, u') \cdot \prod_{p \in var(t')} M_p(p\theta, \emptyset, p\theta')$$

for every $q \in Q$, $t \in T_\Sigma$, and $u \in T_\Delta$. Note that we will write $M_q(t, \emptyset, u)$ instead of $M_q(t, u)$ if $t \in T_\Sigma$. This equality can also be used as a recursive definition of $M_q(t, u)$ because the recursive calls $M_p(p\theta, p\theta')$ are made to smaller trees (i.e., $|p\theta| \leq |t|$ and $|p\theta'| \leq |u|$ and one of these inequalities is strict) since $t' \notin Q$ or $u' \notin Q$. Let us prove this equality. Note that we essentially separate the first derivation step,

which is typical for top-down devices.

$$
\begin{aligned}
M_q(t,u) &= \sum_{D \in D_M^q(t,\emptyset,u)} wt_M(D) \\
&= \sum_{\substack{D=(\xi_0,\ldots,\xi_k) \text{ derivation} \\ \xi_0=(q,\{(\varepsilon,\varepsilon)\},q),\, \xi_k=(t,\emptyset,u)}} wt_M(D) \\
&= \sum_{(q,\{(\varepsilon,\varepsilon)\},q) \overset{a_1}{\Rightarrow}_M \xi_1 \overset{a_2}{\Rightarrow}_M \cdots \overset{a_k}{\Rightarrow}_M (t,\emptyset,u)} \prod_{i=1}^{k} a_i \\
&= \sum_{\substack{(t',q,u') \in supp(R) \\ (q,\{(\varepsilon,\varepsilon)\},q) \overset{a_1}{\Rightarrow}_M (t',\ell,u') \overset{a_2}{\Rightarrow}_M \cdots \overset{a_k}{\Rightarrow}_M (t,\emptyset,u)}} R(t',q,u') \cdot \prod_{i=2}^{k} a_i \\
&= \sum_{\substack{(t',q,u') \in supp(R) \\ \theta : var(t') \to T_\Sigma,\, t=t'\theta \\ \theta' : var(u') \to T_\Delta,\, u=u'\theta' \\ \forall q_i \in var(t') : (q_i,(\{\varepsilon,\varepsilon)\},q_i) \overset{a_{i1}}{\Rightarrow}_M \cdots \overset{a_{ik_i}}{\Rightarrow}_M (q_i\theta,\emptyset,q_i\theta')}} R(t',q,u') \cdot \prod_{q_i \in var(t')} \left(\prod_{j=1}^{k_i} a_{ij} \right) \\
&= \sum_{\substack{(t',q,u') \in supp(R) \\ \theta : var(t') \to T_\Sigma,\, t=t'\theta \\ \theta' : var(u') \to T_\Delta,\, u=u'\theta'}} R(t',q,u') \cdot \prod_{p \in var(t')} M_p(p\theta, p\theta').
\end{aligned}
$$

3 Multi bottom-up tree transducers

We already mentioned in the Introduction that we want to compare STSG to another model that we propose as an alternative. The alternative is the weighted (linear and nondeleting) multi bottom-up tree transducer, which has been introduced by Arnold and Dauchet (1982) and Lilin (1981). A more detailed (and English) presentation is Engelfriet *et al.* (2009). Let us quickly recall the formal definition, which we extend with weights. Recall that $X = \{x_1, x_2, \ldots\}$.

Definition 2
A *weighted multi bottom-up tree transducer* (MBOT) is a system $(Q, \Sigma, \Delta, F, R)$ where

- Q, Σ, and Δ are ranked alphabets of states, input symbols, and output symbols, respectively,
- $F : Q_1 \to A$ assigns final weights to unary states Q_1, and
- $R : T_\Sigma(Q(X)) \times Q(T_\Delta(X)) \to A$ are weighted rules such that $supp(R)$ is finite and for every $(l,r) \in supp(R)$
  - l and r are linear in X,
  - $var(l) = var(r)$, and
  - $l \notin Q(X)$ or $r \notin Q(X)$.

The components l and r of a rule $(l,r) \in supp(R)$ are called left- and right-hand side, respectively. Although we will not use a linking structure for MBOT, the links

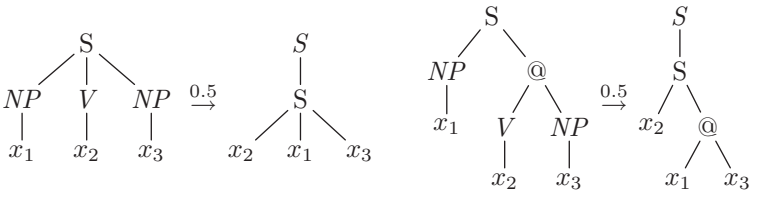

Fig. 2. MBOT rule (states are slanted) and the same rule with binarized trees where @ is an arbitrary symbol that is introduced in the binarization process.

can be imagined between equal variables of X (not between the symbols of Q) in the left- and right-hand side of a rule. Figure 2 displays example rules of an MBOT. Roughly speaking, an MBOT is a weighted (linear and nondeleting) extended bottom-up tree transducer (Engelfriet, Fülöp and Vogler 2002; Engelfriet, Lilin and Maletti 2009), in which the states of Q can have ranks different from 1. To keep the presentation simple, we assume that final states have rank 1 (hence the final weight assignment F is of type $F : Q_1 \to A$). The size $|(l,r)|$ of a rule $(l,r) \in supp(R)$ is $|l| + |r|$, and the size $|M|$ of the MBOT M is $\sum_{\rho \in supp(R)} |\rho|$.

The final condition on the rules in Definition 2 ensures that derivations are finite because each step will consume an input symbol or generate an output symbol. We continue with the rewrite semantics for the MBOT $M = (Q, \Sigma, \Delta, F, R)$. Again, we define left-most derivations only.

Definition 3
Let $c \in C_\Sigma(Q(T_\Delta))$ and $\theta : X \to T_\Delta$ such that $w < w'$ for every $w \in pos_\square(c)$ and $w' \in pos_Q(c)$. Then $c[l\theta] \overset{a}{\Rightarrow}_M c[r\theta]$ if $R(l,r) = a$. A sequence $D = (\xi_0, \ldots, \xi_k)$ of $\xi_i \in T_\Sigma(Q(T_\Delta))$ is a *derivation* if there are $a_1, \ldots, a_k \in A$ such that $\xi_0 \overset{a_1}{\Rightarrow}_M \cdots \overset{a_k}{\Rightarrow}_M \xi_k$. Note that such weights $a_1, \ldots, a_k$ are unique. The weight $wt_M(D)$ of the derivation D is $wt_M(D) = \prod_{i=1}^k a_i$. Moreover, for every $\xi, \zeta \in T_\Sigma(Q(T_\Delta))$, we let

$$D_M(\xi, \zeta) = \{D \mid D = (\xi_0, \ldots, \xi_k) \text{ is a derivation with } \xi_0 = \xi \text{ and } \xi_k = \zeta\}.$$

Note that the set $D_M(\xi, \zeta)$ is finite. For every $t \in T_\Sigma$, and $r' \in Q(T_\Delta)$, let $M(t, r') = \sum_{D \in D_M(t,r')} wt_M(D)$. The weight $M(t,u)$ assigned by the MBOT M is $M(t,u) = \sum_{q \in Q_1} F(q) \cdot M(t, q(u))$ for every $t \in T_\Sigma$ and $u \in T_\Delta$.

Note that the condition on the context c in Definition 3 ensures that the derivation is left-most. The rules of Figure 3 are applied in a derivation in Figure 4. The first displayed derivation step uses the context $S(NP(t_1), \square)$ and any substitution θ such that $\theta(x_2) = t_2$ and $\theta(x_3) = t_3$.

Again, the derivation semantics is very instructive, but difficult to handle in proofs. To overcome this problem, we observe the following equality:

$$M(t, \xi) = \sum_{\substack{(l,r) \in supp(R) \\ (l', t_1, \ldots, t_n) \in match(t) \\ \theta : var(l') \to Q(X),\, l = l'\theta \\ \theta' : var(r) \to T_\Delta,\, \xi = r\theta'}} R(l,r) \cdot \prod_{i=1}^n M(t_i, x_i\theta\theta')$$

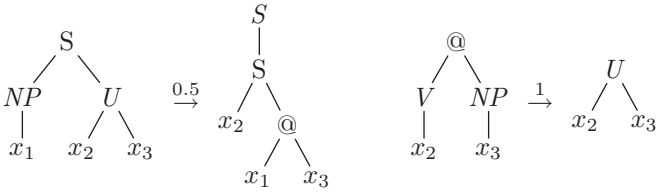

Fig. 3. Sample MBOT rules in one-symbol normal form (see Definition 5; in essence each rule contains exactly one input or output symbol). The states of Q are slanted.

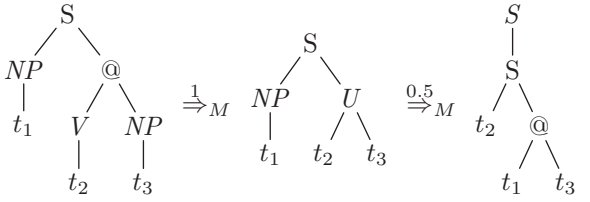

Fig. 4. Derivation using the MBOT rules of Figure 3.

for every $t \in T_\Sigma$ and $\xi \in Q(T_\Delta)$. Again, this equation also yields a recursive definition of $M(t, \xi)$ because the recursive calls are made to smaller input or output trees.

Again, let us prove this equality, so that we have a solid base for the following proofs. For every $t \in T_\Sigma$ and $\xi \in Q(T_\Delta)$,

$$\begin{aligned}
M(t,\xi) &= \sum_{D \in D_M(t,\xi)} wt_M(D) \\
&= \sum_{t \overset{a_1}{\Rightarrow}_M \xi_1 \overset{a_2}{\Rightarrow}_M \cdots \overset{a_k}{\Rightarrow}_M \xi} \prod_{i=1}^{k} a_i \\
&= \sum_{\substack{(l,r) \in supp(R) \\ \theta' : var(r) \to T_\Delta \\ t \overset{a_1}{\Rightarrow}_M \cdots \overset{a_{k-1}}{\Rightarrow}_M l\theta' \overset{a_k}{\Rightarrow}_M r\theta' = \xi}} R(l,r) \cdot \prod_{i=1}^{k-1} a_i \\
&= \sum_{\substack{(l,r) \in supp(R) \\ (l', t_1, \ldots, t_n) \in match(t) \\ \theta : var(l') \to Q(X),\ l = l'\theta \\ \theta' : var(r) \to T_\Delta,\ \xi = r\theta'}} R(l,r) \cdot \prod_{i=1}^{n} M(t_i, x_i\theta\theta').
\end{aligned}$$

This proof is similar to the corresponding proof for STSG, but here we separated the last derivation step, which is common for bottom-up devices.

It is argued by Chiang (2005), Knight (2007), and Graehl *et al.* (2008) that STSG (and extended top-down tree transducers) have sufficient power for syntax-based machine translation. MBOT should be at least as powerful as STSG, we first demonstrate how each STSG can be encoded as an MBOT.

Theorem 4
For every STSG M, we can construct an equivalent MBOT in time $\mathcal{O}(|M|)$.

Proof
Let $M = (Q, \Sigma, \Delta, I, R)$ be an STSG. Then we construct the MBOT $M' = (Q', \Sigma, \Delta, I, R')$ with $Q'_1 = Q$ and $Q'_k = \emptyset$ for every $k \neq 1$ and for every $(t, q, u) \in supp(R)$ we let $R'(l, r) = R(t, q, u)$ where

- $var(t) = \{q_1, \ldots, q_k\}$,
- $\theta_{q_1 \cdots q_k} : var(t) \to Q'(X)$ and $\theta'_{q_1 \cdots q_k} : var(u) \to X$ with $\theta_{q_1 \cdots q_k}(q_i) = q_i(x_i)$ and $\theta'_{q_1 \cdots q_k}(q_i) = x_i$ for every $i \in [k]$, and
- $l = t\theta_{q_1 \cdots q_k}$ and $r = q(u\theta'_{q_1 \cdots q_k})$.

Clearly, M' can be constructed in time $\mathcal{O}(|M|)$. To prove the correctness of the construction, we have to prove the statement $M_q(t', u') = M'(t', q(u'))$ for every $q \in Q$, $t' \in T_\Sigma$, and $u' \in T_\Delta$.

$$\begin{aligned}
M_q(t', u') &= \sum_{\substack{(t,q,u) \in supp(R) \\ \theta : var(t) \to T_\Sigma, t' = t\theta \\ \theta' : var(u) \to T_\Delta, u' = u\theta'}} R(t, q, u) \cdot \prod_{p \in var(t)} M_p(p\theta, p\theta') \\
&= \sum_{\substack{(t,q,u) \in supp(R), (l,r) \in supp(R') \\ var(t) = \{q_1, \ldots, q_k\} \\ \theta : var(t) \to T_\Sigma, t' = t\theta \\ \theta' : var(u) \to T_\Delta, u' = u\theta' \\ l = t\theta_{q_1 \cdots q_k}, r = q(u\theta'_{q_1 \cdots q_k})}} R'(l, r) \cdot \prod_{p \in var(t)} M'(p\theta, p(p\theta')) \\
&= \sum_{\substack{(l,r) \in supp(R') \\ (l', t_1, \ldots, t_n) \in match(t') \\ \theta : var(l') \to Q(X), l = l'\theta \\ \theta' : var(r) \to T_\Delta, q(u') = r\theta'}} R'(l, r) \cdot \prod_{i=1}^{n} M'(t_i, x_i\theta\theta') \\
&= M'(t', q(u')).
\end{aligned}$$

With this auxiliary statement, the main statement is now easy to prove for every $t \in T_\Sigma$ and $u \in T_\Delta$.

$$M(t, u) = \sum_{q \in Q} I(q) \cdot M_q(t, u) = \sum_{q \in Q} I(q) \cdot M'(t, q(u)) = M'(t, u) \qquad \square$$

STSG rules and their corresponding MBOT rules according to the construction in the proof of Theorem 4 are displayed in Figures 1 and 2.

4 Binarization

Binarization is an important tool for efficiency reasons in nondeterministic devices. This is based on the simple and powerful observation (Wang, Knight and Marcu 2007) that instead of making 5 choices from a space of n in one instant (represented by n^5 rules), we can sometimes make them one-by-one (represented by only $5n$ rules). The best-known example of this approach is the binarization of context-free grammars [see CHOMSKY normal form in Hopcroft and Ullman (1979)]. An STSG $M = (Q, \Sigma, \Delta, I, R)$ is *binarized* if $|t|_Q \leq 2$ for every $(t, q, u) \in supp(R)$. The benefits of binarized STSG are presented in Zhang *et al.* (2006) and Wang *et al.* (2007), which in addition, also present linear-time algorithms for the binarization

of *binarizable* STSG. It should be mentioned that those algorithms are presented for tree-to-string devices (in which the right-hand side of an STSG rule is a string; for example, the yield of our right-hand sides), but they can easily be generalized to our tree-to-tree devices. However, not all (tree-to-tree or tree-to-string) STSG are binarizable. More precisely, binarizable (tree-to-tree) STSG cannot even handle simple rotations, which severely limits their expressive power.

Binarization consists of two steps: (i) binarization of the involved trees (using the auxiliary symbol @; see Figure 1) and (ii) binarization of the derivations of the processing device (e.g., tree automata, tree transducers, STSG, or MBOT). Let us formalize binarization of trees next. Let $t \in T_\Sigma(V)$ and $@ \notin \Sigma \cup V$ be a new binary symbol. Then $bin(t)$ is recursively defined as follows:

$$bin(v) = v$$

$$bin(\sigma(t_1, \ldots, t_k)) = \begin{cases} \sigma(bin(t_1), \ldots, bin(t_k)) & \text{if } k \leq 2 \\ \sigma(t_1, @(t_2, \ldots, @(t_{k-1}, t_k)\ldots)) & \text{otherwise} \end{cases}$$

for every $v \in V$, $\sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in T_\Sigma(V)$. In fact, there are several ways to binarize trees, but to keep the presentation simple, we will always assume the presented form.

We show that the benefits of binarization can be reaped for *all* weighted tree transformations computable by STSG. We have already demonstrated that every STSG can be transformed into an equivalent MBOT in linear time. Next, we show that MBOT can be efficiently binarized. The MBOT $M = (Q, \Sigma, \Delta, F, R)$ is *(input) binarized* if $|l|_Q \leq 2$ for every $(l, r) \in supp(R)$.

Definition 5
The MBOT $M = (Q, \Sigma, \Delta, F, R)$ is in *one-symbol normal form* if $|l|_\Sigma + |r|_\Delta = 1$ for every $(l, r) \in supp(R)$.

Figure 3 presents some MBOT rules in one-symbol normal form. Given an input ranked alphabet Σ such that $\Sigma_k = \emptyset$ for every $k \geq 3$ (i.e., there are at most binary input symbols), every MBOT M in one-symbol normal form is (input) binarized. Every unweighted MBOT can be transformed into one-symbol normal form in linear time (Raoult 1993; Engelfriet *et al.* 2009) in the size of the MBOT. This procedure can easily be extended to the weighted case, which we show in the next theorem.

Theorem 6
For every MBOT M an equivalent MBOT in one-symbol normal form can be constructed in time $\mathcal{O}(|M|)$.

Proof
Let $M = (Q, \Sigma, \Delta, F, R)$. The proof will have two stages. First, we ensure that $|l|_\Sigma \leq 1$ for every rule $(l, r) \in supp(R)$. The same construction for the output side can then be used to also ensure that $|r|_\Delta \leq 1$ for every $(l, r) \in supp(R)$. Since they are essentially the same constructions, we will only cover the former (the input side). We decompose rules $(l, r) \in supp(R)$ with more than one input symbol in the left-hand side l into several rules using essentially the construction in Lemma 14

and Theorem 15 of Engelfriet *et al.* (2009), which in turn is (a variation of) the construction in Proposition II.B.5 of Lilin (1978).

Let $(l, r) \in supp(R)$ with $|l|_\Sigma \geq 2$. Then $l = \sigma(l_1, \ldots, l_k)$ for some $\sigma \in \Sigma_k$ and $l_1, \ldots, l_k \in T_\Sigma(Q(X))$. For every $i \in [k]$, let $var(l_i) = \{x_{i1}, \ldots, x_{in_i}\}$ and $q_i \notin Q$ be a new state of rank n_i. Moreover, let

$$l'_i = \begin{cases} l_i & \text{if } l_i \in Q(X) \\ q_i(x_{i1}, \ldots, x_{in_i}) & \text{otherwise.} \end{cases}$$

We construct the MBOT $M' = (Q', \Sigma, \Delta, F', R')$ such that

- $Q' = Q \cup \{q_1, \ldots, q_k\}$,
- $F'(q) = F(q)$ for every $q \in Q_1$ and $F'(q_i) = 0$ for every $i \in [k]$ with $n_i = 1$,
- for every $(l', r') \in supp(R)$

$$R'(l', r') = \begin{cases} 0 & \text{if } (l', r') = (l, r) \\ R(l', r') & \text{otherwise,} \end{cases}$$

- $R'(l_i, l'_i) = 1$ for every $i \in [k]$ such that $l_i \notin Q(X)$, and
- $R'(\sigma(l'_1, \ldots, l'_k), r) = R(l, r)$.

It is straightforward to prove that M' and M are equivalent. This can easily be proved using the derivation semantics because every derivation that uses the rule (l, r) of M can be simulated by several rules of M'. Moreover, since the states $q_1, \ldots, q_k$ do not have any other rules any derivation in M' will simulate the effect of the rule (l, r). Let $m = \max\{|l|_\Sigma \mid (l, r) \in supp(R)\}$. Clearly, M' will have one less rule with m input symbols than M. Hence, repeated application of the above construction eventually yields an equivalent MBOT $M'' = (Q'', \Sigma, \Delta, F'', R'')$ such that $|l|_\Sigma \leq 1$ for every $(l, r) \in supp(R'')$. Moreover, the process clearly terminates in time $\mathcal{O}(|M|)$. □

In Figures 1 and 2 we show rules and their corresponding rules in binarized form (i.e., with binarized trees in the rules). It is known (Aho and Ullman 1972) that, in general, STSG (or SCFG or extended top-down tree transducers) cannot be binarized. In Figure 3 we illustrate the binarization construction in the proof of Theorem 6 on the rule of Figure 2. How the decomposed rules combine into the original rule is demonstrated in Figure 4, which also shows two example derivation steps. In the next section, we show one benefit of the binarization on the BAR-HILLEL construction.

Corollary 7 (of Theorem 6)
For every STSG M an equivalent, binarized MBOT can be constructed in $\mathcal{O}(|M|)$.

Finally, let us shortly discuss why we call 'binarized' also 'input binarized'. In a binarized MBOT each rule concerns at most two input subtrees. Note that the variables of X in rules represent output trees. Consequently, we can call the MBOT $M = (Q, \Sigma, \Delta, F, R)$ *output binarized* if $|r|_X \leq 2$ for all $(l, r) \in supp(R)$. We already mentioned that STSG cannot be binarized and these problems translate into output binarization. More specifically, binarizable STSG (Zhang *et al.* 2006; Wang *et al.* 2007) can be transformed into output-binarized MBOT, but in general, MBOT cannot be output-binarized.

5 Input and output restriction

A standard operation for tree transformations (and tree languages alike) is *regular restriction*. For transformations this operation can be applied to the input or output side. These constructions are used in parsing, integration of a language model, and the computation of certain metrics [see Nederhof and Satta (2003), Nederhof and Satta (2008), or Satta (2010) for a detailed account]. In addition, together with domain and range constructions (see Section 6), they can be used to prove preservation of recognizability. The construction is generally known as BAR-HILLEL construction [see Bar-Hillel, Perles and Shamir (1964) for the original construction on context-free grammars] if the restricting language is presented by a WSA.

Since STSG are symmetric, only one construction is needed for them. Let us formally define the input product. Given an STSG $M = (Q, \Sigma, \Delta, I, R)$ and a WSA N with states P, the input product is an STSG M' such that

$$M'(t,u) = M(t,u) \cdot N(yd(t))$$

for every $t \in T_\Sigma$ and $u \in T_\Delta$. The construction of M' is called BAR-HILLEL construction for M and N. The *rank* of an STSG rule $(t,q,u) \in supp(R)$ is $rk(t,q,u) = |t|_Q$. The maximal rank $rk(M) = \max_{\rho \in supp(R)} rk(\rho)$ of a rule of M enters as an exponent into the time complexity $\mathcal{O}(|M| \cdot |P|^{2rk(M)+5})$ of the BAR-HILLEL construction (Maletti and Satta 2010). Since binarization is not possible in general, the maximal rank $rk(M)$ cannot be limited to 2. In contrast, binarization is possible for MBOT with only linear overhead, so we investigate whether we can exploit this advantage in a BAR-HILLEL construction for MBOT.

We start with a classical product construction for the input side. Given an MBOT $M = (Q, \Sigma, \Delta, F, R)$ in one-symbol normal form such that the symbols in $\Sigma \cup \Delta$ have rank at most 2 and a TSG $N = (P, \Sigma, \Sigma, I, R')$ that is also a SCFG (see Proposition 1), we want to construct an MBOT M' such that $M'(t,u) = M(t,u) \cdot N(t)$ for every $t \in T_\Sigma$ and $u \in T_\Delta$. In other words, each input tree should be rescored according to N; in the unweighted case this yields that the translation of M is filtered to the set of input trees accepted by N.

We occasionally use the angled parentheses '$\langle$' and '$\rangle$' instead of the standard parentheses.

Definition 8
The input product $Prod(N, M)$ is the MBOT

$$Prod(N, M) = (P \times Q, \Sigma, \Delta, F'', R'')$$

such that

- $F''(\langle p, q\rangle) = I(p) \cdot F(q)$ for every $p \in P$ and $q \in Q_1$,
- for every rule $(l, r) \in supp(R)$ with $l = q(x_{i_1}, \ldots, x_{i_k})$ and $r = q'(r_1, \ldots, r_n)$ where $q \in Q_k$, $q' \in Q_n$, and $r_1, \ldots, r_n \in T_\Delta(X)$, and for every $p \in P$, let

$$R''(\langle p, q\rangle(x_{i_1}, \ldots, x_{i_k}), \langle p, q'\rangle(r_1, \ldots, r_n)) = R(l, r),$$

- for every $(l, r) \in supp(R)$ with

$$l = \sigma(q_1(x_{i_1}, \ldots, x_{i_{n_1}}), \ldots, q_k(x_{i_{n_{k-1}+1}}, \ldots, x_{i_n}))$$

and $r = q(x_{j_1}, \ldots, x_{j_n})$ where $\sigma \in \Sigma_k$, $q, q_1, \ldots, q_k \in Q$, and for every $p, p_1, \ldots, p_k \in P$, let

— $l' = \sigma(\langle p_1, q_1\rangle(x_{i_1}, \ldots, x_{i_{n_1}}), \ldots, \langle p_k, q_k\rangle(x_{i_{n_{k-1}+1}}, \ldots, x_{i_n}))$,
— $r' = \langle p, q\rangle(x_{j_1}, \ldots, x_{j_n})$, and
— $R''(l', r') = R(l, r) \cdot R'(p \to \sigma(p_1, \ldots, p_k))$.

The first type of rule constructed in the second item does not involve an input symbol. Thus, the nonterminal p of N is just forwarded to the new state in the right-hand side. Since no step with respect to the TSG N is made, the weight is taken from the rule of M. The second type of rule constructed in the third item uses a rule of R with the input symbol σ and a rule of R' that also contains σ. Both rules are executed in parallel in the resulting rule and its weight is the product of the weights of the original rules. Overall, this is a classical product construction, which is similar to other product constructions such as Borchardt (2004).

Theorem 9
For every STSG $M = (Q, \Sigma, \Delta, F, R)$ in one-symbol normal form and TSG $N = (P, \Sigma, \Sigma, I, R')$ that is also an SCFG, we have

$$Prod(N, M)(t, u) = M(t, u) \cdot N(t)$$

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

Proof
Let $M'' = Prod(N, M) = (P \times Q, \Sigma, \Delta, F'', R'')$. We first prove that

$$M''(t, \langle p, q\rangle(u_1, \ldots, u_n)) = M(t, q(u_1, \ldots, u_n)) \cdot N_p(t)$$

for every $p \in P$, $q \in Q_n$, $t \in T_\Sigma$, and $u_1, \ldots, u_n \in T_\Delta$. Let $\xi = \langle p, q\rangle(u_1, \ldots, u_n)$ and $t = \sigma(t_1, \ldots, t_k)$ for some $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma$.

$$\begin{aligned}
M''(t, \xi) &= \sum_{\substack{(l,r) \in supp(R'') \\ (l', t'_1, \ldots, t'_m) \in match(t) \\ \theta : var(l') \to (P \times Q)(X), l = l'\theta \\ \theta' : var(r) \to T_\Delta, \xi = r\theta'}} R''(l, r) \cdot \prod_{i=1}^{m} M''(t'_i, x_i\theta\theta') \\
&= \sum_{\substack{(l,r) \in supp(R'') \\ l = \langle p', q'\rangle(x_{j_1}, \ldots, x_{j_m}) \\ \theta' : var(r) \to T_\Delta, \xi = r\theta'}} R''(l, r) \cdot M''(t, l\theta') \\
&\quad + \sum_{\substack{(l,r) \in supp(R'') \\ l = \sigma(\langle p_1, q_1\rangle(\cdots), \ldots, \langle p_k, q_k\rangle(\cdots)) \\ \theta' : var(r) \to T_\Delta, \xi = r\theta'}} R''(l, r) \cdot \prod_{i=1}^{k} M''(t_i, \langle p_i, q_i\rangle(\cdots)\theta') \\
&= \sum_{\substack{(l,r) \in supp(R), l \in Q(X) \\ \theta' : var(r) \to T_\Delta \\ r\theta' = q(u_1, \ldots, u_n)}} R(l, r) \cdot M(t, l\theta') \cdot N_p(t)
\end{aligned}$$

$$+ \sum_{\substack{(l,r)\in supp(R)\\ l=\sigma(q_1(\cdots),\ldots,q_k(\cdots))\\ p_1,\ldots,p_k\in P\\ \theta':var(r)\to T_\Delta\\ r\theta'=q(u_1,\ldots,u_n)}} R(l,r)\cdot R'(p\to\sigma(p_1,\ldots,p_k))\cdot\prod_{i=1}^{k} M(t_i,q_i(\cdots)\theta')\cdot N_{p_i}(t_i)$$

$$= \sum_{\substack{(l,r)\in supp(R),\, l\in Q(X)\\ \theta':var(r)\to T_\Delta\\ r\theta'=q(u_1,\ldots,u_n)}} R(l,r)\cdot M(t,l\theta')\cdot N_p(t)$$

$$+ \sum_{\substack{(l,r)\in supp(R)\\ l=\sigma(q_1(\cdots),\ldots,q_k(\cdots))\\ \theta':var(r)\to T_\Delta\\ r\theta'=q(u_1,\ldots,u_n)}} R(l,r)\cdot\prod_{i=1}^{k} M(t_i,q_i(\cdots)\theta')\cdot N_p(t)$$

$$= \sum_{\substack{(l,r)\in supp(R)\\ (l',t'_1,\ldots,t'_m)\in match(t)\\ \theta:var(l')\to Q(X),\, l=l'\theta\\ \theta':var(r)\to T_\Delta\\ r\theta'=q(u_1,\ldots,u_n)}} R(l,r)\cdot\prod_{i=1}^{m} M(t'_i,x_i\theta\theta')\cdot N_p(t)$$

$$= M(t,q(u_1,\ldots,u_n))\cdot N_p(t)$$

Now we can complete the proof as follows. For every $t\in T_\Sigma$ and $u\in T_\Delta$

$$M''(t,u) = \sum_{\langle p,q\rangle\in P\times Q_1} F''(\langle p,q\rangle)\cdot M''(t,\langle p,q\rangle(u))$$

$$= \sum_{p\in P,q\in Q_1} I(p)\cdot F(q)\cdot M(t,q(u))\cdot N_p(t) = M(t,u)\cdot N(t). \qquad \square$$

Finally, let us discuss the time complexity. The MBOT $Prod(N,M)$ can be obtained in time $\mathcal{O}(|M|\cdot|N|)$. Furthermore, it is known [see, for example, Maletti and Satta (2009)] that for every WSA N with states P, we can construct a TSG N' that is also a SCFG, which has size $\mathcal{O}(|\Sigma|\cdot|P|^3)$, such that $N'(t)=N(yd(t))$ for every $t\in T_\Sigma$. The main idea of this well-known construction is illustrated in Figure 5.

Corollary 10 (of Theorem 9)
For every MBOT $M=(Q,\Sigma,\Delta,F,R)$ and every WSA $N=(P,\Gamma,I,v,G)$, an MBOT M' can be constructed in time $\mathcal{O}(|M|\cdot|P|^3)$ such that $M'(t,u)=M(t,u)\cdot N(yd(t))$ for every $t\in T_\Sigma$ and $u\in T_\Delta$.

Thus, our BAR-HILLEL construction has the well-known (parsing) complexity $\mathcal{O}(|M|\cdot|P|^3)$. Compared to the time complexity $\mathcal{O}(|M|\cdot|P|^{2rk(M)+5})$ of the BAR-HILLEL construction for an STSG M, a huge efficiency gain is achieved by binarization (and one-symbol normal form). In fact, if the original STSG M is binarizable (i.e., $rk(M)=2$), then the BAR-HILLEL construction can still run in time $\mathcal{O}(|M|\cdot|P|^{6+3})$, which is due to the fact that several symbols can occur in each rule. This can be explained on the left illustration of Figure 5. If several symbols are present in the tree represented by a triangle in the illustration, then the arrows need no longer indicate identity. Consequently, all indicated state positions could hold different states, which

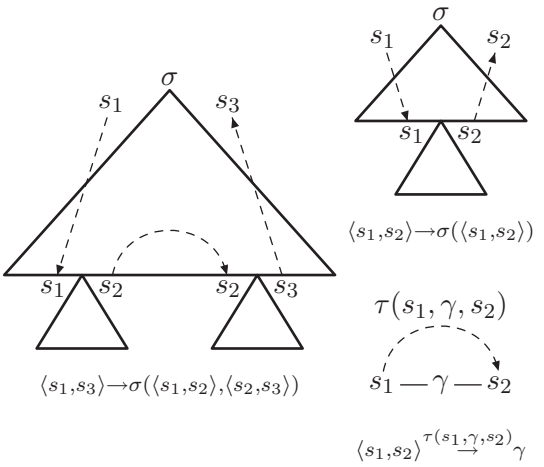

Fig. 5. Constructing a TSG from a WSA.

yields the factor $\mathcal{O}(|P|^6)$. Since we also have to run the WSA N on several symbols, we obtain a processing factor $\mathcal{O}(|P|^3)$. The latter factor is a pure processing factor, so the size of the output STSG would be bound by $\mathcal{O}(|M| \cdot |P|^6)$. Only if a normal form analogous to one-symbol normal form can be achieved for M, then the complexity reduces to $\mathcal{O}(|M| \cdot |P|^3)$. Consequently, the STSG M should be transformed into an equivalent MBOT in one-symbol normal form, which can be achieved in linear time (see Corollary 7), and the BAR-HILLEL construction should be performed on this MBOT.

Since MBOT are not symmetric, a special construction is needed for the output side. We will see that we cannot obtain a similar efficiency improvement in this case. To simplify the presentation, we assume without loss of generality that the variables occur in order in l (i.e., l is k-normed) for every rule $(l, r) \in supp(R)$.

Definition 11
The output product $Prod(M, N)$ is the MBOT

$$Prod(M, N) = (Q(P), \Sigma, \Delta, F'', R''),$$

where

- $F''(q\langle p\rangle) = I(p) \cdot F(q)$ for every $p \in P$ and $q \in Q_1$,
- for every rule $(l, r) \in supp(R)$ with $l = q(x_{j_1}, \ldots, x_{j_k})$ and

$$r = q'(x_{i_1}, \ldots, x_{i_j}, \delta(x_{i_{j+1}}, \ldots, x_{i_m}), x_{i_{m+1}}, \ldots, x_{i_n}),$$

  where $q \in Q_k$, $q' \in Q_n$ and $\delta \in \Delta_{m-j}$, and for every $p, p_1, \ldots, p_k \in P$, let
  - $l' = q\langle p_1, \ldots, p_k\rangle(x_{j_1}, \ldots, x_{j_k})$,
  - $r'' = \delta(x_{i_{j+1}}, \ldots, x_{i_m})$,
  - $r' = q'\langle p_{i_1}, \ldots, p_{i_j}, p, p_{i_{m+1}}, \ldots, p_{i_n}\rangle(x_{i_1}, \ldots, x_{i_j}, r'', x_{i_{m+1}}, \ldots, x_{i_n})$, and
  - $R''(l', r') = R(l, r) \cdot R'(p \to \delta(p_{i_{j+1}}, \ldots, p_{i_m}))$,

- for all $(l,r) \in supp(R)$ with $l = \sigma(q_1(x_{i_1},\ldots,x_{i_{j_1}}),\ldots,q_k(x_{i_{j_{k-1}+1}},\ldots,x_{i_n}))$ and $r = q(x_{n_1},\ldots,x_{n_m})$ where $\sigma \in \Sigma_k$, $q,q_1,\ldots,q_k \in Q$, and $p_1,\ldots,p_n \in P$, let
  - $j = j_{k-1} + 1$,
  - $l' = \sigma(q_1\langle p_{i_1},\ldots,p_{i_{j_1}}\rangle(x_{i_1},\ldots,x_{i_{j_1}}),\ldots,q_k\langle p_{i_j},\ldots,p_{i_n}\rangle(x_{i_j},\ldots,x_{i_n}))$,
  - $r' = q\langle p_{n_1},\ldots,p_{n_m}\rangle(x_{n_1},\ldots,x_{n_m})$, and
  - $R''(l',r') = R(l,r)$.

Since the first type of rule constructed in the second item involves an output symbol, we perform steps with respect to M and N. The second type of rule does not contain an output symbol, and thus, the nonterminals of N are just forwarded to the right-hand side.

Theorem 12
For every MBOT $M = (Q,\Sigma,\Delta,F,R)$ in one-symbol normal form and TSG $N = (P,\Delta,\Delta,I,R')$ that is also an SCFG, we have

$$Prod(M,N)(t,u) = M(t,u)\cdot N(u)$$

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

Proof
The proof is similar to the proof of Theorem 9. □

The BAR-HILLEL construction for the output side of an MBOT M with a WSA N with states P can be obtained in time $\mathcal{O}(|M|\cdot|P|^{2rk(M)+2})$ where $rk(M)$ is the maximal number of variables of X in a rule of M. Compared to the time complexity $\mathcal{O}(|M|\cdot|P|^{2rk(M)+5})$ of the BAR-HILLEL construction for an STSG M, the efficiency gain is minimal and only due to the fact that at most one output symbol needs to be processed in the BAR-HILLEL construction for MBOT (because the MBOT can be assumed to be in one-symbol normal form). A similar normal form cannot, in general, be obtained for STSG, and thus, an additional evaluation effort $\mathcal{O}(|P|^3)$ occurs, which represents the time complexity of running the WSA N on the output symbols present in a rule. It was already mentioned that if the STSG M is binarizable, then we can obtain an equivalent output-binarized MBOT. In this case, the problem is symmetric and the results for the input product apply also for the output product.

6 Forward and backward application

Finally, we want to apply a weighted tree transformation not just to a single input or output tree, but rather to a set of (potentially weighted) input or output trees. Let M be an MBOT or an STSG. Forward application aims to compute a TSG of output trees under M given a TSG of input trees. Conversely, backward application aims to compute a TSG of input trees given a TSG of output trees. Formally, let $N = (P,\Sigma,\Sigma,I,R')$ and $N' = (P',\Delta,\Delta,I',R'')$ be TSG. Then the forward application $M\langle N\rangle$ and the backward application $M^{-1}\langle N'\rangle$ are

$$M\langle N\rangle(u) = \sum_{t\in T_\Sigma} M(t,u)\cdot N(t)$$

$$M^{-1}\langle N'\rangle(t) = \sum_{u\in T_\Delta} M(t,u)\cdot N'(u)$$

for every $t \in T_\Sigma$ and $u \in T_\Delta$. Note that the sums in the above equations might be infinite. We will later address this issue by additional restrictions, which will ensure that the sums are finite.

In general, forward and backward application can be reduced to range and domain, respectively, with the help of the product constructions of the previous section. Formally, the range ran_M and the domain dom_M of the transformation computed by M are

$$ran_M(u) = \sum_{t \in T_\Sigma} M(t,u) \qquad \text{and} \qquad dom_M(t) = \sum_{u \in T_\Delta} M(t,u)$$

for every $t \in T_\Sigma$ and $u \in T_\Delta$. Again, those sums might be infinite. Let M be an MBOT. The relation between the two notions is

$$M\langle N\rangle(u) = \sum_{t \in T_\Sigma} M(t,u) \cdot N(t) = \sum_{t \in T_\Sigma} M'(t,u) = ran_{M'}(u)$$
$$M^{-1}\langle N'\rangle(t) = \sum_{u \in T_\Delta} M(t,u) \cdot N'(u) = \sum_{u \in T_\Delta} M''(t,u) = dom_{M''}(t)$$

for every $t \in T_\Sigma$ and $u \in T_\Delta$ where $M' = Prod(N, M)$ and $M'' = Prod(M, N')$. The same relation also holds for an STSG M.

Let us first discuss STSG. Let $M = (Q, \Sigma, \Delta, I, R)$ be an STSG. It is *input ε-free* (respectively, *output ε-free*) if $t \notin Q$ (respectively, $u \notin Q$) for every $(t, q, u) \in supp(R)$. It is known (Fülöp, Maletti and Vogler 2010) that the forward application $M\langle N\rangle$ (respectively, the backward application $M^{-1}\langle N'\rangle$) can be represented by a TSG if M is output ε-free (respectively, input ε-free). In other words, STSG preserve recognizability under forward and backward application in all reasonable cases.

Unfortunately, the range of an MBOT is not necessarily representable by a TSG, which is not due to well-definedness problems (infinite sums) but rather the *finite-copying* property (Engelfriet *et al.* 1980) of MBOT. Because of this copying property the range might not be recognizable. Consequently, forward application suffers from the same problem, which yields that MBOT do not preserve recognizability under forward application. In analogy to the definition for STSG, let the MBOT $M = (Q, \Sigma, \Delta, F, R)$ be *input ε-free* if $l \notin Q(X)$ for every $(l, r) \in supp(R)$. For such MBOT we will now discuss the domain (and backward application).

Definition 13
Let $M = (Q, \Sigma, \Delta, F, R)$ be an input ε-free MBOT such that $Q_i \cap Q_j = \emptyset$ for all $i \neq j$. In other words, we assume without loss of generality that each state has a unique rank. We construct the TSG $N = (Q, \Sigma, \Sigma, I, R')$ such that

- $I(q) = F(q)$ for every $q \in Q_1$ and $I(q) = 0$ for all $q \in Q \setminus Q_1$,
- for every $t \in T_\Sigma(Q)$

$$R'(q \to t) = \sum_{\substack{(l,r) \in supp(R) \\ cut(l) = t \\ r = q(r_1, \dots, r_k)}} R(l, r)$$

where $cut(l) \in T_\Sigma(Q)$ is tree obtained from $l \in T_\Sigma(Q(X))$ by deleting the nodes labeled by an element of X.

Theorem 14
Let M be an input ε-free MBOT and N be the TSG as in Definition 13. Then $dom_M = N$.

Proof
We need to prove that $\sum_{u_1,\ldots,u_k \in T_\Delta} M(t, q(u_1,\ldots,u_k)) = N_q(t)$ for every $q \in Q_k$ and $t \in T_\Sigma$.

$$
\begin{aligned}
\sum_{u_1,\ldots,u_k \in T_\Delta} M(t, q(u_1,\ldots,u_k)) &= \sum_{u_1,\ldots,u_k \in T_\Delta} \left(\sum_{\substack{(l,r)\in supp(R)\\ (l',t_1,\ldots,t_m)\in match(t)\\ \theta : var(l')\to Q(X),\, l=l'\theta\\ \theta' : var(r)\to T_\Delta\\ q(u_1,\ldots,u_k)=r\theta'}} R(l,r)\cdot \prod_{i=1}^{m} M(t_i, x_i\theta\theta') \right) \\
&= \sum_{\substack{(l,r)\in supp(R)\\ (l',t_1,\ldots,t_m)\in match(t)\\ \theta : var(l')\to Q(X),\, l=l'\theta\\ r=q(r_1,\ldots,r_k)\\ \forall i\in[m] : \theta_i : var(x_i\theta)\to T_\Delta}} R(l,r)\cdot \prod_{i=1}^{m} M(t_i, x_i\theta\theta_i) \\
&= \sum_{\substack{(l,r)\in supp(R)\\ (l',t_1,\ldots,t_m)\in match(t)\\ \theta : var(l')\to Q(X),\, l=l'\theta\\ r=q(r_1,\ldots,r_k)}} R(l,r)\cdot \prod_{i=1}^{m} \left(\sum_{\theta_i : var(x_i\theta)\to T_\Delta} M(t_i, x_i\theta\theta_i) \right) \\
&= \sum_{\substack{(l,r)\in supp(R)\\ (l',t_1,\ldots,t_m)\in match(t)\\ \theta : var(l')\to Q(X),\, l=l'\theta\\ r=q(r_1,\ldots,r_k)\\ x_i\theta\in\{q_i\}(X)}} R(l,r)\cdot \prod_{i=1}^{m} N_{q_i}(t_i) \\
&= \sum_{\substack{(l,r)\in supp(R)\\ (l',t_1,\ldots,t_m)\in match(t)\\ \theta : var(l')\to Q\\ cut(l)=l'\theta\\ r=q(r_1,\ldots,r_k)}} R(l,r)\cdot \prod_{i=1}^{m} N_{x_i\theta}(t_i) \\
&= \sum_{\substack{(l',t_1,\ldots,t_m)\in match(t)\\ \theta : var(l')\to Q}} R'(q\to l'\theta)\cdot \prod_{i=1}^{m} N_{x_i\theta}(t_i) \\
&= N_q(t)
\end{aligned}
$$

Then

$$
\begin{aligned}
dom_M(t) &= \sum_{u\in T_\Delta} M(t,u) = \sum_{q\in Q_1} \left(F(q)\cdot \sum_{u\in T_\Delta} M(t,q(u)) \right) \\
&= \sum_{q\in Q_1} I(q)\cdot N_q(t) = \sum_{q\in Q} I(q)\cdot N_q(t) = N(t)
\end{aligned}
$$

for every $t \in T_\Sigma$, which proves the statement. □

Consequently, every input ε-free MBOT preserves recognizability under backward application, which is essentially the same statement as for STSG. However, the situation is entirely different for forward application. STSG preserve recognizability under forward application, but MBOT do not necessarily.

Conclusion

We have shown that MBOT have significant advantages over STSG since each STSG can be transformed into an equivalent MBOT in linear time and the obtained MBOT can be binarized and transformed into one-symbol normal form (also in linear time). The input and output product algorithms for MBOT have a better asymptotic runtime complexity than their counterparts for STSG. This is illustrated in the following table:

	STSG	binarizable STSG	MBOT
input:	$\mathcal{O}(\lvert M\rvert \cdot \lvert P\rvert^{2rk(M)+5})$	$\mathcal{O}(\lvert M\rvert \cdot \lvert P\rvert^{9})$	$\mathcal{O}(\lvert M\rvert \cdot \lvert P\rvert^{3})$
output:	$\mathcal{O}(\lvert M\rvert \cdot \lvert P\rvert^{2rk(M)+5})$	$\mathcal{O}(\lvert M\rvert \cdot \lvert P\rvert^{9})$	$\mathcal{O}(\lvert M\rvert \cdot \lvert P\rvert^{2rk(M)+2})$

where $|M|$ is the size of the tree transformation device (STSG or MBOT) and $|P|$ is the number of states of the restricting WSA. We add that the output product for an MBOT corresponding to a binarizable STSG can be constructed in time $\mathcal{O}(|M| \cdot |P|^6)$. An implementation of MBOT and the experimental verification of these theoretical advantages remains future work.

In the final section, we showed that the additional power of MBOT should be used carefully. The range of an MBOT is, in general, not recognizable, which yields that forward application does not preserve recognizability. This is due to the *finite-copying* feature that MBOT can employ. This feature is not present in MBOT that have been obtained from STSG (i.e., that are equivalent to an STSG).

Acknowledgments

The author would like to thank the reviewers for their insightful comments, which improved the presentation of the article. In addition, the author gratefully acknowledges the financial support by the *Ministerio de Educación y Ciencia* (MEC) grants JDCI-2007-760 and MTM-2007-63422.

References

Aho, A. V., and Ullman, J. D. 1972. *The Theory of Parsing, Translation, and Compiling*. Upper Saddle River, NJ, USA: Prentice Hall.

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

Arnold, A., and Dauchet, M. 1982. Morphismes et bimorphismes d'arbres. *Theoretical Computer Science* **20**(1): 33–93.

Bar-Hillel, Y., Perles, M., and Shamir, E. 1964. On formal properties of simple phrase structure grammars. In Y. Bar-Hillel (ed.), *Language and Information: Selected Essays on their Theory and Application*, Chapter 9, pp. 116–150. Reading, MA, USA: Addison Wesley.

Berstel, J., and Reutenauer, C. 1982. Recognizable formal power series on trees. *Theoretical Computer Science* **18**(2): 115–48.

Borchardt, B. 2004. A pumping lemma and decidability problems for recognizable tree series. *Acta Cybernetica* **16**(4): 509–44.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. 1990. A statistical approach to machine translation. *Computational Linguistics* **16**(2): 79–85.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. 1993. Mathematics of statistical machine translation: parameter estimation. *Computational Linguistics* **19**(2): 263–311.

Chiang, D. 2005. A hierarchical phrase-based model for statistical machine translation. In K. Knight, H. T. Ng, and K. Oflazer (eds.), *Association for Computational Linguistics: 43rd Annual Meeting*, pp. 263–70. Ann Arbor, MI, USA: Association for Computational Linguistics.

Chiang, D., and Knight, K. 2006. Tutorial: an introduction to synchronous grammars. In N. Calzolari, C. Cardie, and P. Isabelle (eds.), *Association for Computational Linguistics: 44th Annual Meeting*, Sydney, Australia: Association for Computational Linguistics.

DeNero, J., Bansal, M., Pauls, A., and Klein, D. 2009. Efficient parsing for transducer grammars. In M. Ostendorf, M. Collins, S. Narayanan, D. W. Oard, and L. Vanderwende (eds.), *Human Language Technologies: 2009 Annual Conference*, pp. 227–35. Boulder, CO, USA: Association for Computational Linguistics.

DeNero, J., Pauls, A., and Klein, D. 2009. Asynchronous binarization for synchronous grammars. In K.-Y. Su, J. Su, J. Wiebe, and H. Li (eds.), *Association for Computational Linguistics: 47th Annual Meeting*, pp. 141–4. Singapore, Singapore: Association for Computational Linguistics.

Eilenberg, S. 1974. *Automata, Languages, and Machines*, Volume 59 of *Pure and Applied Math.* Orlando, FL, USA: Academic Press.

Engelfriet, J., Fülöp, Z., and Vogler, H. 2002. Bottom-up and top-down tree series transformations. *Journal of Automata, Languages and Combinatorics* **7**(1): 11–70.

Engelfriet, J., Lilin, E., and Maletti, A. 2009. Extended multi bottom-up tree transducers: Composition and decomposition. *Acta Informatica* **46**(8): 561–90.

Engelfriet, J., Rozenberg, G., and Slutzki, G. 1980. Tree transducers, L systems, and two-way machines. *Journal of Computer and System Sciences* **20**(2): 150–202.

Fülöp, Z., Maletti, A., and Vogler, H. 2010. Preservation of recognizability for synchronous tree substitution grammars. In Drewes, F., and Kuhlmann, M. (eds.), *Applications of Tree Automata in Natural Language Processing: 2010 Workshop*, pp. 1–9. Uppsala, Sweden: Association for Computational Linguistics.

Fülöp, Z., and Vogler, H. 2009. Weighted tree automata and tree transducers. In M. Droste, W. Kuich, and H. Vogler (eds.), *Handbook of Weighted Automata*, pp. 313–403. EATCS Monographs on Theoretical Computer Science, Chapter IX. Berlin, Germany: Springer.

Golan, J. S. 1999. *Semirings and their Applications.* Dordrecht: Kluwer Academic.

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

Hebisch, U., and Weinert, H. J. 1998. *Semirings — Algebraic Theory and Applications in Computer Science*. Singapore: World Scientific.

Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages and Computation.* Reading, MA, USA: Addison Wesley.

Huang, L., Zhang, H., Gildea, D., and Knight, K. 2009. Binarization of synchronous context-free grammars. *Computational Linguistics* **35**(4), 559–95.

Knight, K. 2007. Capturing practical natural language transformations. *Machine Translation* **21**(2): 121–33.

Lilin, E. 1978. *Une Généralisation des Transducteurs D'états Finis D'arbres: les S-transducteurs.* Thèse 3ème cycle, Université de Lille.

Lilin, E. 1981. Propriétés de clôture d'une extension de transducteurs d'arbres déterministes. In E. Astesiano, and C. Böhm (eds.), *Trees in Algebra and Programming: 6th Colloquium*,

Volume 112 of *Lecture Notes in Computer Science*, pp. 280–289. Genoa, Italy: Springer.

Maletti, A., Graehl, J., Hopkins, M., and Knight, K. 2009. The power of extended top-down tree transducers. *SIAM Journal on Computing* **39**(2): 410–30.

Maletti, A., and Satta, G. 2009. Parsing algorithms based on tree automata. In E. V. de la Clergerie, H. Bunt, and L. Danlos (eds.), *Parsing Technologies: 11th International Conference*, pp. 1–12. Paris, France: Association for Computational Linguistics.

Maletti, A., and Satta, G. 2010. Parsing and translation algorithms based on weighted extended tree transducers. In F. Drewes, and M. Kuhlmann (eds.), *Applications of Tree Automata in Natural Language Processing: 2010 Workshop*, pp. 19–27. Uppsala, Sweden: Association for Computational Linguistics.

Mohri, M. 2009. Weighted automata algorithms. In M. Droste, W. Kuich, and H. Vogler (eds.), *Handbook of Weighted Automata*, pp. 213–254. EATCS Monographs on Theoretical Computer Science, Chapter IV. Berlin, Germany: Springer.

Nederhof, M.-J., and Satta, G. 2003. Probabilistic parsing as intersection. In *Parsing Technologies: 8th International Conference*, pp. 137–148. Nancy, France: Association for Computational Linguistics.

Nederhof, M.-J., and Satta, G. 2008. Computation of distances for regular and context-free probabilistic languages. *Theoretical Computer Science* **395**(2–3): 235–54.

Raoult, J.-C. 1993. Recursively defined tree transductions. In C. Kirchner (ed.), *Rewriting Techniques and Applications: 5th International Conference*, Volume 690 of *Lecture Notes in Computer Science*, pp. 343–357. Montreal, Canada: Springer.

Sakarovitch, J. 2009. Rational and recognisable power series. In M. Droste, W. Kuich, and H. Vogler (eds.), *Handbook of Weighted Automata*, pp. 105–174. EATCS Monographs on Theoretical Computer Science, Chapter IV. Berlin, Germany: Springer.

Satta, G. 2010. Translation algorithms by means of language intersection. (Manuscript).

Schützenberger, M. P. 1961. On the definition of a family of automata. *Information and Control* **4**(2–3): 245–70.

Wang, W., Knight, K., and Marcu, D. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In J. Eisner (ed.), *Empirical Methods in Natural Language Processing: 2007 Joint Conference*, pp. 746–754. Prague, Czech Republic: Association for Computational Linguistics.

Zhang, H., Huang, L., Gildea, D., and Knight, K. 2006. Synchronous binarization for machine translation. In R. C. Moore, J. Bilmes, J. Chu-Carroll, and M. Sanderson (eds.), *Human Language Technology: 2006 Annual Conference*, pp. 256–263. New York, NY, USA: Association for Computational Linguistics.