HAL
open science

Composing extended top-down tree transducers

Aurélie Lagoutte, Fabienne Braune, Daniel Quernheim, Andreas Maletti

To cite this version:

Aurélie Lagoutte, Fabienne Braune, Daniel Quernheim, Andreas Maletti. Composing extended top-down tree transducers. European Chapter of the Association for Computational Linguistics (EACL), 2012, Avignon, France. pp.808-817. hal-01508546

HAL Id: hal-01508546

https://hal.science/hal-01508546v1

Submitted on 14 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Composing extended top-down tree transducers*

Aurélie Lagoutte
École normale supérieure de Cachan, Département Informatique
alagoutt@dptinfo.ens-cachan.fr

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

Abstract

A composition procedure for linear and nondeleting extended top-down tree transducers is presented. It is demonstrated that the new procedure is more widely applicable than the existing methods. In general, the result of the composition is an extended top-down tree transducer that is no longer linear or nondeleting, but in a number of cases these properties can easily be recovered by a post-processing step.

1 Introduction

Tree-based translation models such as synchronous tree substitution grammars (Eisner, 2003; Shieber, 2004) or multi bottom-up tree transducers (Lilin, 1978; Engelfriet et al., 2009; Maletti, 2010; Maletti, 2011) are used for several aspects of syntax-based machine translation (Knight and Graehl, 2005). Here we consider the *extended top-down tree transducer* (XTOP), which was studied in (Arnold and Dauchet, 1982; Knight, 2007; Graehl et al., 2008; Graehl et al., 2009) and implemented in the toolkit TIBURON (May and Knight, 2006; May, 2010). Specifically, we investigate compositions of linear and nondeleting XTOPs (ln-XTOP). Arnold and Dauchet (1982) showed that ln-XTOPs compute a class of transformations that is not closed under composition, so we cannot compose two arbitrary ln-XTOPs into a single ln-XTOP. However, we will show that ln-XTOPs can be composed into a (not necessarily linear or nondeleting) XTOP. To illustrate the use of ln-XTOPs in machine translation, we consider the following English sentence together with a German reference translation:

* All authors were financially supported by the EMMY NOETHER project MA/4959/1-1 of the German Research Foundation (DFG).

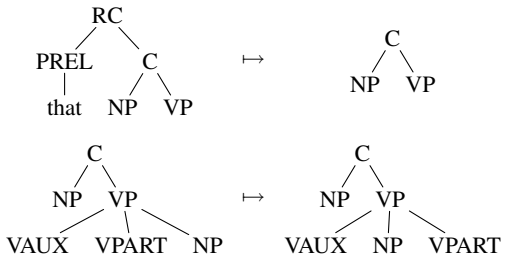

Figure 1: Word drop [top] and reordering [bottom].

The newswire reported yesterday *that the Serbs have completed the negotiations*.

Gestern [Yesterday] berichtete [reported] die [the] Nachrichtenagentur [newswire] *die* [the] *Serben* [Serbs] *hätten* [would have] *die* [the] *Verhandlungen* [negotiations] *beendet* [completed].

The relation between them can be described (Yamada and Knight, 2001) by three operations: drop of the relative pronoun, movement of the participle to end of the clause, and word-to-word translation. Figure 1 shows the first two operations, and Figure 2 shows ln-XTOP rules performing them. Let us now informally describe the execution of an ln-XTOP on the top rule ρ of Figure 2. In general, ln-XTOPs process an input tree from the root towards the leaves using a set of rules and states. The state p in the left-hand side of ρ controls the particular operation of Figure 1 [top]. Once the operation has been performed, control is passed to states p_{NP} and p_{VP}, which use their own rules to process the remaining input subtree governed by the variable below them (see Figure 2). In the same fashion, an ln-XTOP containing the bottom rule of Figure 2 reorders the English verbal complex.

In this way we model the word drop by an ln-XTOP M and reordering by an ln-XTOP N. The syntactic properties of linearity and nondeletion yield nice algorithmic properties, and the mod-

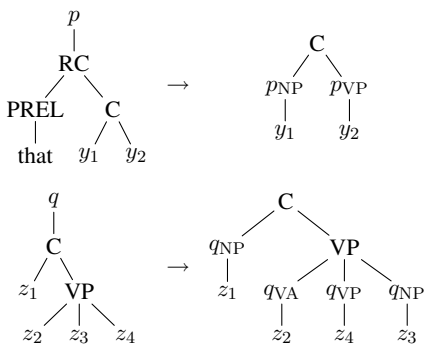

Figure 2: XTOP rules for the operations of Figure 1.

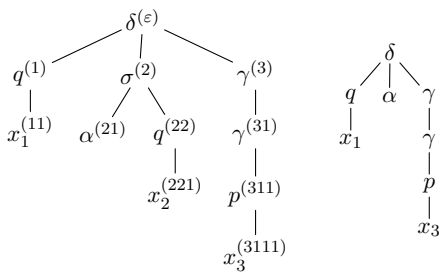

Figure 3: Linear normalized tree $t \in T_\Sigma(Q(X))$ [left] and $t[\alpha]_2$ [right] with $\mathrm{var}(t) = \{x_1, x_2, x_3\}$. The positions are indicated in t as superscripts. The subtree $t|_2$ is $\sigma(\alpha, q(x_2))$.

ular approach is desirable for better design and parametrization of the translation model (May et al., 2010). Composition allows us to recombine those parts into one device modeling the whole translation. In particular, it gives all parts the chance to vote at the same time. This is especially important if pruning is used because it might otherwise exclude candidates that score low in one part but well in others (May et al., 2010).

Because ln-XTOP is not closed under composition, the composition of M and N might be outside ln-XTOP. These cases have been identified by Arnold and Dauchet (1982) as infinitely "overlapping cuts", which occur when the right-hand sides of M and the left-hand sides of N are unboundedly overlapping. This can be purely syntactic (for a given ln-XTOP) or semantic (inherent in all ln-XTOPs for a given transformation). Despite the general impossibility, several strategies have been developed: (i) Extension of the model (Maletti, 2010; Maletti, 2011), (ii) online composition (May et al., 2010), and (iii) restriction of the model, which we follow. Compositions of subclasses in which the XTOP N has at most one input symbol in its left-hand sides have already been studied in (Engelfriet, 1975; Baker, 1979; Maletti and Vogler, 2010). Such compositions are implemented in the toolkit TIBURON. However, there are translation tasks in which the used XTOPs do not fulfill this requirement. Suppose that we simply want to compose the rules of Figure 2, The bottom rule does not satisfy the requirement that there is at most one input symbol in the left-hand side.

We will demonstrate how to compose two linear and nondeleting XTOPs into a single XTOP, which might however no longer be linear or nondeleting. However, when the syntactic form of the composed XTOP has only bounded overlapping cuts, post-processing will get rid of them and restore an ln-XTOP. In the remaining cases, in which unbounded overlapping is necessary or occurs in the syntactic form but would not be necessary, we will compute an XTOP. This is still an improvement on the existing methods that just fail. Since general XTOPs are implemented in TIBURON and the new composition covers (essentially) all cases currently possible, our new composition procedure could replace the existing one in TIBURON. Our approach to composition is the same as in (Engelfriet, 1975; Baker, 1979; Maletti and Vogler, 2010): We simply parse the right-hand sides of the XTOP M with the left-hand sides of the XTOP N. However, to facilitate this approach we have to adjust the XTOPs M and N in two pre-processing steps. In a first step we cut left-hand sides of rules of N into smaller pieces, which might introduce non-linearity and deletion into N. In certain cases, this can also introduce finite look-ahead (Engelfriet, 1977; Graehl et al., 2009). To compensate, we expand the rules of M slightly. Section 4 explains those preparations. Next, we compose the prepared XTOPs as usual and obtain a single XTOP computing the composition of the transformations computed by M and N (see Section 5). Finally, we apply a post-processing step to expand rules to reobtain linearity and nondeletion. Clearly, this cannot be successful in all cases, but often removes the non-linearity introduced in the pre-processing step.

2 Preliminaries

Our trees have labels taken from an alphabet Σ of symbols, and in addition, leaves might be labeled by elements of the countably infinite

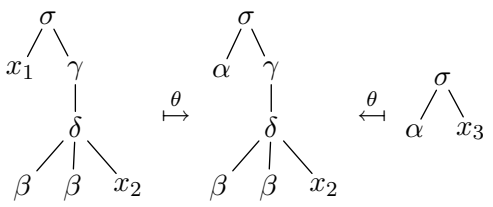

Figure 4: Substitution where $\theta(x_1) = \alpha$, $\theta(x_2) = x_2$, and $\theta(x_3) = \gamma(\delta(\beta, \beta, x_2))$.

set $X = \{x_1, x_2, \dots\}$ of formal variables. Formally, for every $V \subseteq X$ the set $T_\Sigma(V)$ of Σ-trees with V-leaves is the smallest set such that $V \subseteq T_\Sigma(V)$ and $\sigma(t_1, \dots, t_k) \in T_\Sigma(V)$ for all $k \in \mathbb{N}$, $\sigma \in \Sigma$, and $t_1, \dots, t_k \in T_\Sigma(V)$. To avoid excessive universal quantifications, we drop them if they are obvious from the context.

For each tree $t \in T_\Sigma(X)$ we identify nodes by positions. The root of t has position ε and the position iw with $i \in \mathbb{N}$ and $w \in \mathbb{N}^*$ addresses the position w in the i-th direct subtree at the root. The set of all positions in t is $\mathrm{pos}(t)$. We write $t(w)$ for the label (taken from $\Sigma \cup X$) of t at position $w \in \mathrm{pos}(t)$. Similarly, we use

- $t|_w$ to address the subtree of t that is rooted in position w, and
- $t[u]_w$ to represent the tree that is obtained from replacing the subtree $t|_w$ at w by $u \in T_\Sigma(X)$.

For a given set $L \subseteq \Sigma \cup X$ of labels, we let

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

be the set of all positions whose label belongs to L. We also write $\mathrm{pos}_l(t)$ instead of $\mathrm{pos}_{\{l\}}(t)$. The tree $t \in T_\Sigma(V)$ is *linear* if $|\mathrm{pos}_x(t)| \leq 1$ for every $x \in X$. Moreover,

$$\mathrm{var}(t) = \{x \in X \mid \mathrm{pos}_x(t) \neq \emptyset\}$$

collects all variables that occur in t. If the variables occur in the order $x_1, x_2, \dots$ in a pre-order traversal of the tree t, then t is *normalized*. Given a finite set Q, we write $Q(T)$ with $T \subseteq T_\Sigma(X)$ for the set $\{q(t) \mid q \in Q, t \in T\}$. We will treat elements of $Q(T)$ as special trees of $T_{\Sigma \cup Q}(X)$. The previous notions are illustrated in Figure 3.

A substitution θ is a mapping $\theta\colon X \to T_\Sigma(X)$. When applied to a tree $t \in T_\Sigma(X)$, it will return the tree $t\theta$, which is obtained from t by replacing all occurrences of $x \in X$ (in parallel) by $\theta(x)$. This can be defined recursively by $x\theta = \theta(x)$ for all $x \in X$ and $\sigma(t_1, \dots, t_k)\theta = \sigma(t_1\theta, \dots, t_k\theta)$

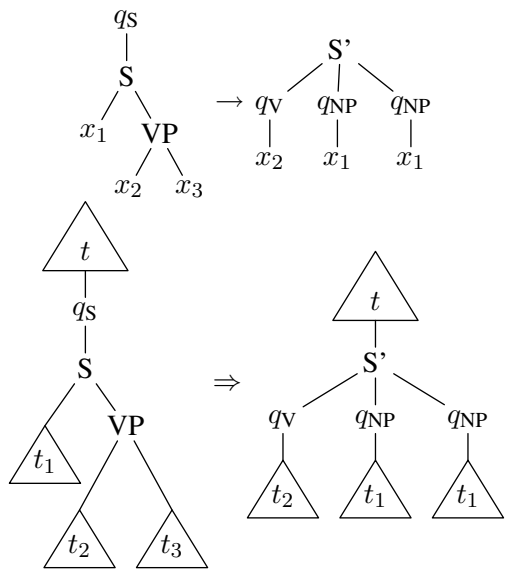

Figure 5: Rule and its use in a derivation step.

for all $\sigma \in \Sigma$ and $t_1, \dots, t_k \in T_\Sigma(X)$. The effect of a substitution is displayed in Figure 4. Two substitutions $\theta, \theta'\colon X \to T_\Sigma(X)$ can be composed to form a substitution $\theta\theta'\colon X \to T_\Sigma(X)$ such that $\theta\theta'(x) = \theta(x)\theta'$ for every $x \in X$.

Next, we define two notions of compatibility for trees. Let $t, t' \in T_\Sigma(X)$ be two trees. If there exists a substitution θ such that $t' = t\theta$, then t' is an *instance* of t. Note that this relation is not symmetric. A *unifier* θ for t and t' is a substitution θ such that $t\theta = t'\theta$. The unifier θ is a *most general unifier* (short: mgu) for t and t' if for every unifier θ'' for t and t' there exists a substitution θ' such that $\theta\theta' = \theta''$. The set $\mathrm{mgu}(t, t')$ is the set of all mgus for t and t'. Most general unifiers can be computed efficiently (Robinson, 1965; Martelli and Montanari, 1982) and all mgus for t and t' are equal up to a variable renaming.

Example 1. Let $t = \sigma(x_1, \gamma(\delta(\beta, \beta, x_2)))$ and $t' = \sigma(\alpha, x_3)$. Then $\mathrm{mgu}(t, t')$ contains θ such that $\theta(x_1) = \alpha$ and $\theta(x_3) = \gamma(\delta(\beta, \beta, x_2))$. Figure 4 illustrates the unification. □

3 The model

The discussed model in this contribution is an extension of the classical *top-down tree transducer*, which was introduced by Rounds (1970) and Thatcher (1970). The *extended top-down tree transducer with finite look-ahead* or just XTOP^{F} and its variations were studied in (Arnold and Dauchet, 1982; Knight and Graehl, 2005;

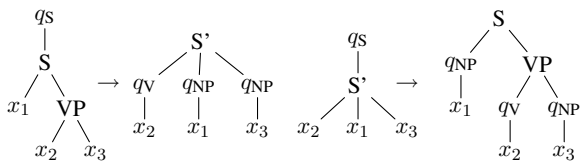

Figure 6: Rule [left] and reversed rule [right].

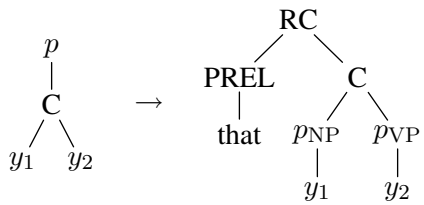

Figure 7: Top rule of Figure 2 reversed.

Knight, 2007; Graehl et al., 2008; Graehl et al., 2009). Formally, an *extended top-down tree transducer with finite look-ahead* (XTOP^{F}) is a system $M = (Q, \Sigma, \Delta, I, R, c)$ where

- Q is a finite set of *states*,
- Σ and Δ are alphabets of *input* and *output symbols*, respectively,
- $I \subseteq Q$ is a set of *initial states*,
- R is a finite set of (rewrite) *rules* of the form $\ell \to r$ where $\ell \in Q(T_\Sigma(X))$ is linear and $r \in T_\Delta(Q(\mathrm{var}(\ell)))$, and
- $c\colon R \times X \to T_\Sigma(X)$ assigns a *look-ahead restriction* to each rule and variable such that $c(\rho, x)$ is linear for each $\rho \in R$ and $x \in X$.

The XTOP^{F} M is *linear* (respectively, *nondeleting*) if r is linear (respectively, $\mathrm{var}(r) = \mathrm{var}(\ell)$) for every rule $\ell \to r \in R$. It has *no look-ahead* (or it is an XTOP) if $c(\rho, x) \in X$ for all rules $\rho \in R$ and $x \in X$. In this case, we drop the look-ahead component c from the description. A rule $\ell \to r \in R$ is *consuming* (respectively, *producing*) if $\mathrm{pos}_\Sigma(\ell) \neq \emptyset$ (respectively, $\mathrm{pos}_\Delta(r) \neq \emptyset$). We let $\mathrm{Lhs}(M) = \{l \mid \exists q, r\colon q(l) \to r \in R\}$.

Let $M = (Q, \Sigma, \Delta, I, R, c)$ be an XTOP^{F}. In order to facilitate composition, we define sentential forms more generally than immediately necessary. Let Σ' and Δ' be such that $\Sigma \subseteq \Sigma'$ and $\Delta \subseteq \Delta'$. To keep the presentation simple, we assume that $Q \cap (\Sigma' \cup \Delta') = \emptyset$. A *sentential form* of M (using Σ' and Δ') is a tree of $\mathrm{SF}(M) = T_{\Delta'}(Q(T_{\Sigma'}))$. For every $\xi, \zeta \in \mathrm{SF}(M)$, we write $\xi \Rightarrow_M \zeta$ if there exist a position $w \in \mathrm{pos}_Q(\xi)$, a rule $\rho = \ell \to r \in R$, and a substitution $\theta\colon X \to T_{\Sigma'}$ such that $\theta(x)$ is an instance of $c(\rho, x)$ for every $x \in X$ and $\xi = \xi[\ell\theta]_w$ and $\zeta = \xi[r\theta]_w$. If the applicable rules are restricted to a certain subset $R' \subseteq R$, then we also write $\xi \Rightarrow_{R'} \zeta$. Figure 5 illustrates a derivation step. The *tree transformation computed by* M is

$$\tau_M = \{(t, u) \in T_\Sigma \times T_\Delta \mid \exists q \in I\colon q(t) \Rightarrow_M^* u\}$$

where $\Rightarrow_M^*$ is the reflexive, transitive closure of $\Rightarrow_M$. It can easily be verified that the definition of τ_M is independent of the choice of Σ' and Δ'. Moreover, it is known (Graehl et al., 2009) that each XTOP^{F} can be transformed into an equivalent XTOP preserving both linearity and nondeletion. However, the notion of XTOP^{F} will be convenient in our composition construction. A detailed exposition to XTOPs is presented by Arnold and Dauchet (1982) and Graehl et al. (2009).

A linear and nondeleting XTOP M with rules R can easily be reversed to obtain a linear and nondeleting XTOP M^{-1} with rules R^{-1}, which computes the inverse transformation $\tau_{M^{-1}} = \tau_M^{-1}$, by reversing all its rules. A (suitable) rule is reversed by exchanging the locations of the states. More precisely, given a rule $q(l) \to r \in R$, we obtain the rule $q(r') \to l'$ of R^{-1}, where $l' = l\theta$ and r' is the unique tree such that there exists a substitution $\theta\colon X \to Q(X)$ with $\theta(x) \in Q(\{x\})$ for every $x \in X$ and $r = r'\theta$. Figure 6 displays a rule and its corresponding reversed rule. The reversed form of the XTOP rule modeling the insertion operation in Figure 2 is displayed in Figure 7.

Finally, let us formally define composition. The XTOP M computes the tree transformation $\tau_M \subseteq T_\Sigma \times T_\Delta$. Given another XTOP N that computes a tree transformation $\tau_N \subseteq T_\Delta \times T_\Gamma$, we might be interested in the tree transformation computed by the composition of M and N (i.e., running M first and then N). Formally, the composition $\tau_M \,;\, \tau_N$ of the tree transformations τ_M and τ_N is defined by

$$\tau_M \,;\, \tau_N = \{(s, u) \mid \exists t\colon (s, t) \in \tau_M, (t, u) \in \tau_N\}$$

and we often also use the notion 'composition' for XTOP with the expectation that the composition of M and N computes exactly $\tau_M \,;\, \tau_N$.

4 Pre-processing

We want to compose two linear and nondeleting XTOPs $M = (P, \Sigma, \Delta, I_M, R_M)$ and

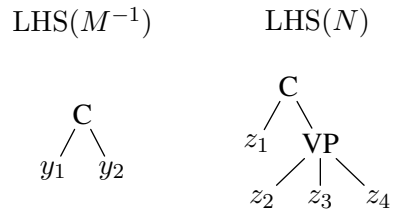

Figure 8: Incompatible left-hand sides of Example 3.

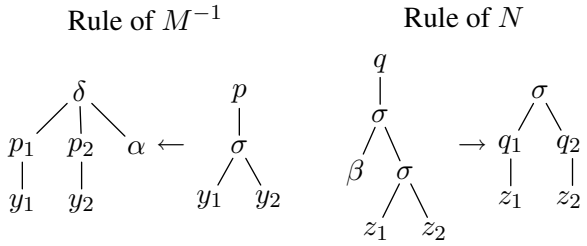

Figure 9: Rules used in Example 5.

$N = (Q, \Delta, \Gamma, I_N, R_N)$. Before we actually perform the composition, we will prepare M and N in two pre-processing steps. After these two steps, the composition is very simple. To avoid complications, we assume that (i) all rules of M are producing and (ii) all rules of N are consuming. For convenience, we also assume that the XTOPs M and N only use variables of the disjoint sets $Y \subseteq X$ and $Z \subseteq X$, respectively.

4.1 Compatibility

In the existing composition results for subclasses of XTOPs (Engelfriet, 1975; Baker, 1979; Maletti and Vogler, 2010) the XTOP N has at most one input symbol in its left-hand sides. This restriction allows us to match rule applications of N to positions in the right-hand sides of M. Namely, for each output symbol in a right-hand side of M, we can select a rule of N that can consume that output symbol. To achieve a similar decomposition strategy in our more general setup, we introduce a compatibility requirement on right-hand sides of M and left-hand sides of N. Roughly speaking, we require that the left-hand sides of N are small enough to completely process right-hand sides of M. However, a comparison of left- and right-hand sides is complicated by the fact that their shape is different (left-hand sides have a state at the root, whereas right-hand sides have states in front of the variables). We avoid these complications by considering reversed rules of M. Thus, an original right-hand side of M is now a left-hand side in the reversed rules and thus has the right format for a comparison. Recall that $\mathrm{Lhs}(N)$ contains all left-hand sides of the rules of N, in which the state at the root was removed.

Definition 2. The XTOP N is *compatible* to M if $\theta(Y) \subseteq X$ for all unifiers $\theta \in \mathrm{mgu}(l_1|_w, l_2)$ between a subtree at a Δ-labeled position $w \in \mathrm{pos}_\Delta(l_1)$ in a left-hand side $l_1 \in \mathrm{Lhs}(M^{-1})$ and a left-hand side $l_2 \in \mathrm{Lhs}(N)$. □

Intuitively, for every Δ-labeled position w in a right-hand side r_1 of M and any left-hand side l_2 of N, we require (ignoring the states) that either (i) $r_1|_w$ and l_2 are not unifiable or (ii) $r_1|_w$ is an instance of l_2.

Example 3. The XTOPs for the English-to-German translation task in the Introduction are not compatible. This can be observed on the left-hand side $l_1 \in \mathrm{Lhs}(M^{-1})$ of Figure 7 and the left-hand side $l_2 \in \mathrm{Lhs}(N)$ of Figure 2[bottom]. These two left-hand sides are illustrated in Figure 8. Between them there is an mgu such that $\theta(Y) \not\subseteq X$ (e.g., $\theta(y_1) = z_1$ and $\theta(y_2) = \mathrm{VP}(z_2, z_3, z_4)$ is such an mgu). □

Theorem 4. There exists an XTOP^{F} N' that is equivalent to N and compatible with M.

Proof. We achieve compatibility by cutting offending rules of the XTOP N into smaller pieces. Unfortunately, both linearity and nondeletion of N might be lost in the process. We first let $N' = (Q, \Delta, \Gamma, I_N, R_N, c_N)$ be the XTOP^{F} such that $c_N(\rho, x) = x$ for every $\rho \in R_N$ and $x \in X$.

If N' is compatible with M, then we are done. Otherwise, let $l_1 \in \mathrm{Lhs}(M^{-1})$ be a left-hand side, $q(l_2) \to r_2 \in R_N$ be a rule, and $w \in \mathrm{pos}_\Delta(l_1)$ be a position such that $\theta(y) \notin X$ for some $\theta \in \mathrm{mgu}(l_1|_w, l_2)$ and $y \in Y$. Let $v \in \mathrm{pos}_y(l_1|_w)$ be the unique position of y in $l_1|_w$.

Now we have to distinguish two cases: (i) Either $\mathrm{var}(l_2|_v) = \emptyset$ and there is no leaf in r_2 labeled by a symbol from Γ. In this case, we have to introduce deletion and look-ahead into N'. We replace the old rule $\rho = q(l_2) \to r_2$ by the new rule $\rho' = q(l_2[z]_v) \to r_2$, where $z \in X \setminus \mathrm{var}(l_2)$ is a variable that does not appear in l_2. In addition, we let $c_N(\rho', z) = l_2|_v$ and $c_N(\rho', x) = c_N(\rho, x)$ for all $x \in X \setminus \{z\}$.

(ii) Otherwise, let $V \subseteq \mathrm{var}(l_2|_v)$ be a maximal set such that there exists a minimal (with respect to the prefix order) position $w' \in \mathrm{pos}(r_2)$ with

Another rule of N

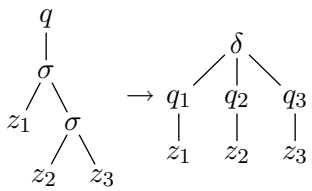

Figure 10: Additional rule used in Example 5.

$\mathrm{var}(r_2|_{w'}) \subseteq \mathrm{var}(l_2|_v)$ and $\mathrm{var}(r_2[\beta]_{w'}) \cap V = \emptyset$, where $\beta \in \Gamma$ is arbitrary. Let $z \in X \setminus \mathrm{var}(l_2)$ be a fresh variable, q' be a new state of N, and $V' = \mathrm{var}(l_2|_v) \setminus V$. We replace the rule $\rho = q(l_2) \to r_2$ of R_N by

$$\begin{aligned} \rho_1 &= q(l_2[z]_v) \to \mathrm{trans}(r_2)[q'(z)]_{w'} \\ \rho_2 &= \ \ q'(l_2|_v) \to r_2|_{w'} \ \ . \end{aligned}$$

The look-ahead for z is trivial and otherwise we simply copy the old look-ahead, so $c_N(\rho_1, z) = z$ and $c_N(\rho_1, x) = c_N(\rho, x)$ for all $x \in X \setminus \{z\}$. Moreover, $c_N(\rho_2, x) = c_N(\rho, x)$ for all $x \in X$. The mapping 'trans' is given for $t = \gamma(t_1, \ldots, t_k)$ and $q''(z'') \in Q(Z)$ by

$$\mathrm{trans}(t) = \gamma(\mathrm{trans}(t_1), \ldots, \mathrm{trans}(t_k))$$

$$\mathrm{trans}(q''(z'')) = \begin{cases} \langle l_2|_v, q'', v' \rangle(z) & \text{if } z'' \in V' \\ q''(z'') & \text{otherwise,} \end{cases}$$

where $v' = \mathrm{pos}_{z''}(l_2|_v)$.

Finally, we collect all newly generated states of the form $\langle l, q, v \rangle$ in Q_l and for every such state with $l = \delta(l_1, \ldots, l_k)$ and $v = iw$, let $l' = \delta(z_1, \ldots, z_k)$ and

$$\langle l, q, v \rangle(l') \to \begin{cases} q(z_i) & \text{if } w = \varepsilon \\ \langle l_i, q, w \rangle(z_i) & \text{otherwise} \end{cases}$$

be a new rule of N without look-ahead.

Overall, we run the procedure until N' is compatible with M. The procedure eventually terminates since the left-hand sides of the newly added rules are always smaller than the replaced rules. Moreover, each step preserves the semantics of N', which completes the proof. □

We note that the look-ahead of N' after the construction used in the proof of Theorem 4 is either trivial (i.e., a variable) or a ground tree (i.e., a tree without variables). Let us illustrate the construction used in the proof of Theorem 4.

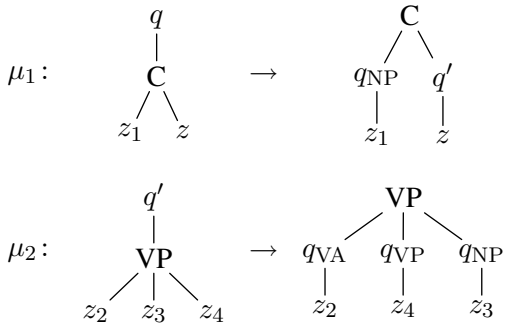

Figure 11: Rules replacing the rule in Figure 7.

Example 5. Let us consider the rules illustrated in Figure 9. We might first note that y_1 has to be unified with β. Since β does not contain any variables and the right-hand side of the rule of N does not contain any non-variable leaves, we are in case (i) in the proof of Theorem 4. Consequently, the displayed rule of N is replaced by a variant, in which β is replaced by a new variable z with look-ahead β.

Secondly, with this new rule there is an mgu, in which y_2 is mapped to $\sigma(z_1, z_2)$. Clearly, we are now in case (ii). Furthermore, we can select the set $V = \{z_1, z_2\}$ and position $w' = \epsilon$. Correspondingly, the following two new rules for N replace the old rule:

$$\begin{aligned} q(\sigma(z, z')) &\to q'(z') \\ q'(\sigma(z_1, z_2)) &\to \sigma(q_1(z_1), q_2(z_2)) \ \ , \end{aligned}$$

where the look-ahead for z remains β.

Figure 10 displays another rule of N. There is an mgu, in which y_2 is mapped to $\sigma(z_2, z_3)$. Thus, we end up in case (ii) again and we can select the set $V = \{z_2\}$ and position $w' = 2$. Thus, we replace the rule of Figure 10 by the new rules

$$\begin{aligned} q(\sigma(z_1, z)) &\to \delta(q_1(z_1), q'(z), \overline{q_3}(z)) \qquad (\star) \\ q'(\sigma(z_2, z_3)) &\to q_2(z_2) \\ \overline{q_3}(\sigma(z_1, z_2)) &\to q_3(z_2) \ \ , \end{aligned}$$

where $\overline{q_3} = \langle \sigma(z_2, z_3), q_3, 2 \rangle$. □

Let us use the construction in the proof of Theorem 4 to resolve the incompatibility (see Example 3) between the XTOPs presented in the Introduction. Fortunately, the incompatibility can be resolved easily by cutting the rule of N (see Figure 7) into the rules of Figure 11. In this example, linearity and nondeletion are preserved.

4.2 Local determinism

After the first pre-processing step, we have the original linear and nondeleting XTOP M and an XTOP^{F} $N' = (Q', \Delta, \Gamma, I_N, R'_N, c_N)$ that is equivalent to N and compatible with M. However, in the first pre-processing step we might have introduced some non-linear (copying) rules in N' (see rule ($\star$) in Example 5), and it is known that "nondeterminism [in M] followed by copying [in N']" is a feature that prevents composition to work (Engelfriet, 1975; Baker, 1979). However, our copying is very local and the copies are only used to project to different subtrees. Nevertheless, during those projection steps, we need to make sure that the processing in M proceeds deterministically. We immediately note that all but one copy are processed by states of the form $\langle l, q, v\rangle \in Q_l$. These states basically process (part of) the tree l and project (with state q) to the subtree at position v. It is guaranteed that each such subtree (indicated by v) is reached only once. Thus, the copying is "resolved" once the states of the form $\langle l, q, v\rangle$ are left. To keep the presentation simple, we just add expanded rules to M such that any rule that can produce a part of a tree l immediately produces the whole tree. A similar strategy is used to handle the look-ahead of N'. Any right-hand side of a rule of M that produces part of a left-hand side of a rule of N' with look-ahead is expanded to produce the required look-ahead immediately.

Let $L \subseteq T_\Delta(Z)$ be the set of trees l such that

- $\langle l, q, v\rangle$ appears as a state of Q_l, or
- $l = l_2\theta$ for some $\rho_2 = q(l_2) \to r_2 \in R'_N$ of N' with non-trivial look-ahead (i.e., $c_N(\rho_2, z) \notin X$ for some $z \in X$), where $\theta(x) = c_N(\rho_2, x)$ for every $x \in X$.

To keep the presentation uniform, we assume that for every $l \in L$, there exists a state of the form $\langle l, q, v\rangle \in Q'$. If this is not already the case, then we can simply add useless states without rules for them. In other words, we assume that the first case applies to each $l \in L$.

Next, we add two sets of rules to R_M, which will not change the semantics but prove to be useful in the composition construction. First, for every tree $t \in L$, let R_t contain all the rules $\overline{p}(l) \to r$, where $\overline{p} = p(l) \to r$ is a new state with $p \in P$, minimal normalized tree $l \in T_\Sigma(X)$, and an instance $r \in T_\Delta(P(X))$ of t such that

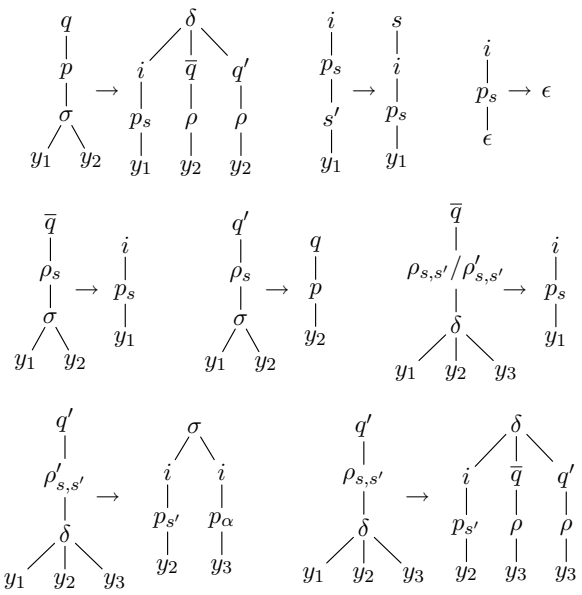

Figure 12: Useful rules for the composition $M'\,;N'$ of Example 8, where $s, s' \in \{\alpha, \beta\}$ and $\rho \in P_{\sigma(z_2,z_3)}$.

$p(l) \Rightarrow^*_{M'} \xi \Rightarrow_{M'} r$ for some ξ that is not an instance of t. In other words, we construct each rule of R_t by applying existing rules of R_M in sequence to generate a (minimal) right-hand side that is an instance of t. We thus potentially make the right-hand sides of M bigger by joining several existing rules into a single rule. Note that this affects neither compatibility nor the semantics. In the second step, we add pure ε-rules that allow us to change the state to one that we constructed in the previous step. For every new state $\overline{p} = p(l) \to r$, let $\mathrm{base}(\overline{p}) = p$. Then $R'_M = R_M \cup R_L \cup R_E$ and $P' = P \cup \bigcup_{t\in L} P_t$ where

$$R_L = \bigcup_{t\in L} R_t \text{ and } P_t = \{\ell(\varepsilon) \mid \ell \to r \in R_t\}$$

$$R_E = \{\mathrm{base}(\overline{p})(x_1) \to \overline{p}(x_1) \mid \overline{p} \in \bigcup_{t\in L} P_t\} \ .$$

Clearly, this does not change the semantics because each rule of R'_M can be simulated by a chain of rules of R_M. Let us now do a full example for the pre-processing step. We consider a nondeterministic variant of the classical example by Arnold and Dauchet (1982).

Example 6. Let $M = (P, \Sigma, \Sigma, \{p\}, R_M)$ be the linear and nondeleting XTOP such that $P = \{p, p_\alpha, p_\beta\}$, $\Sigma = \{\delta, \sigma, \alpha, \beta, \epsilon\}$, and R_M contains the following rules

$$p(\sigma(y_1, y_2)) \to \sigma(p_s(y_1), p(y_2)) \qquad (\dagger)$$

$$
\begin{aligned}
p(\delta(y_1,y_2,y_3)) &\to \sigma(p_s(y_1),\sigma(p_{s'}(y_2),p(y_3)))\\
p(\delta(y_1,y_2,y_3)) &\to \sigma(p_s(y_1),\sigma(p_{s'}(y_2),p_\alpha(y_3)))\\
p_s(s'(y_1)) &\to s(p_s(y_1))\\
p_s(\epsilon) &\to \epsilon
\end{aligned}
$$

for every $s, s' \in \{\alpha, \beta\}$. Similarly, we let $N = (Q, \Sigma, \Sigma, \{q\}, R_N)$ be the linear and non-deleting XTOP such that $Q = \{q, i\}$ and R_N contains the following rules

$$
\begin{aligned}
q(\sigma(z_1,z_2)) &\to \sigma(i(z_1),i(z_2))\\
q(\sigma(z_1,\sigma(z_2,z_3))) &\to \delta(i(z_1),i(z_2),q(z_3)) \quad (\ddagger)\\
i(s(z_1)) &\to s(i(z_1))\\
i(\epsilon) &\to \epsilon
\end{aligned}
$$

for all $s \in \{\alpha, \beta\}$. It can easily be verified that M and N meet our requirements. However, N is not yet compatible with M because an mgu between rules (†) of M and (‡) of N might map y_2 to $\sigma(z_2, z_3)$. Thus, we decompose (‡) into

$$
\begin{aligned}
q(\sigma(z_1,z)) &\to \delta(i(z_1),\overline{q}(z),q'(z))\\
q'(\sigma(z_2,z_3)) &\to q(z_3)\\
\overline{q}(\sigma(z_1,z_2)) &\to i(z_1)
\end{aligned}
$$

where $\overline{q} = \langle \sigma(z_2, z_3), i, 1\rangle$. This newly obtained XTOP N' is compatible with M. In addition, we only have one special tree $\sigma(z_2, z_3)$ that occurs in states of the form $\langle l, q, v\rangle$. Thus, we need to compute all minimal derivations whose output trees are instances of $\sigma(z_2, z_3)$. This is again simple since the first three rule schemes ρ_s, $\rho_{s,s'}$, and $\rho'_{s,s'}$ of M create such instances, so we simply create copies of them:

$$
\begin{aligned}
\rho_s(\sigma(y_1,y_2)) &\to \sigma(p_s(y_1),p(y_2))\\
\rho_{s,s'}(\delta(y_1,y_2,y_3)) &\to \sigma(p_s(y_1),\sigma(p_{s'}(y_2),p(y_3)))\\
\rho'_{s,s'}(\delta(y_1,y_2,y_3)) &\to \sigma(p_s(y_1),\sigma(p_{s'}(y_2),p_\alpha(y_3)))
\end{aligned}
$$

for all $s, s' \in \{\alpha, \beta\}$. These are all the rules of $R_{\sigma(z_2,z_3)}$. In addition, we create the following rules of R_E:

$$
\begin{aligned}
p(x_1) \to \rho_s(x_1) \qquad & p(x_1) \to \rho_{s,s'}(x_1)\\
& p(x_1) \to \rho'_{s,s'}(x_1)
\end{aligned}
$$

for all $s, s' \in \{\alpha, \beta\}$. □

Especially after reading the example it might seem useless to create the rule copies in R_l [in Example 6 for $l = \sigma(z_2, z_3)$]. However, each such rule has a distinct state at the root of the left-hand side, which can be used to trigger only this rule. In this way, the state selects the next rule to apply, which yields the desired local determinism.

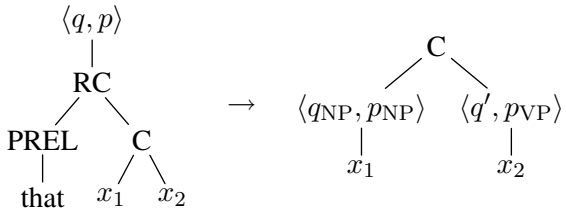

Figure 13: Composed rule created from the rule of Figure 7 and the rules of N' displayed in Figure 11.

5 Composition

Now we are ready for the actual composition. For space efficiency reasons we reuse the notations used in Section 4. Moreover, we identify trees of $T_\Gamma(Q'(P'(X)))$ with trees of $T_\Gamma((Q' \times P')(X))$. In other words, when meeting a subtree $q(p(x))$ with $q \in Q'$, $p \in P'$, and $x \in X$, then we also view this equivalently as the tree $\langle q, p\rangle(x)$, which could be part of a rule of our composed XTOP. However, not all combinations of states will be allowed in our composed XTOP, so some combinations will never yield valid rules.

Generally, we construct a rule of $M'; N'$ by applying a single rule of M' followed by any number of pure ε-rules of R_E, which can turn states $\mathrm{base}(\overline{p})$ into $\overline{p}$. Then we apply any number of rules of N' and try to obtain a sentential form that has the required shape of a rule of M' ; N'.

Definition 7. Let $M' = (P', \Sigma, \Delta, I_M, R'_M)$ and $N' = (Q', \Delta, \Gamma, I_N, R'_N)$ be the XTOPs constructed in Section 4, where $\bigcup_{l\in L} P_l \subseteq P'$ and $\bigcup_{l\in L} Q_l \subseteq Q'$. Let $Q'' = Q' \setminus \bigcup_{l\in L} Q_l$. We construct the XTOP $M'; N' = (S, \Sigma, \Gamma, I_N \times I_M, R)$ where

$$S = \bigcup_{l\in L}(Q_l \times P_l) \cup (Q'' \times P')$$

and R contains all normalized rules $\ell \to r$ (of the required shape) such that

$$\ell \Rightarrow_{M'} \xi \Rightarrow^*_{R_E} \zeta \Rightarrow^*_{N'} r$$

for some $\xi, \zeta \in T_\Gamma(Q'(T_\Delta(P'(X))))$. □

The required rule shape is given by the definition of an XTOP. Most importantly, we must have that $\ell \in S(T_\Sigma(X))$, which we identify with a certain subset of $Q'(P'(T_\Sigma(X)))$, and $r \in T_\Gamma(S(X))$, which similarly corresponds to a subset of $T_\Gamma(Q'(P'(X)))$. The states are simply combinations of the states of M' and N', of

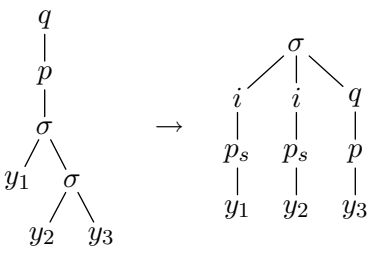

Figure 14: Successfully expanded rule from Example 9.

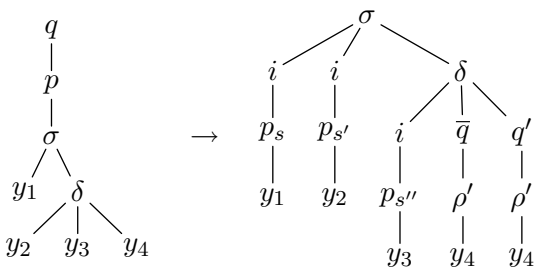

Figure 15: Expanded rule that remains copying (see Example 9).

which however the combinations of a state $q \in Q_l$ with a state $p \notin P_l$ are forbidden. This reflects the intuition of the previous section. If we entered a special state of the form $\langle l, q, v\rangle$, then we should use a corresponding state $p \in P_l$ of M, which only has rules producing instances of l. We note that look-ahead of N' is checked normally in the derivation process.

Example 8. Now let us illustrate the composition on Example 6. Let us start with rule (†) of M.

$$\begin{aligned} & q(p(\sigma(x_1, x_2))) \\ \Rightarrow_{M'}\ & q(\sigma(p_s(x_1), p(x_2))) \\ \Rightarrow_{R_E}\ & q(\sigma(p_s(x_1), \rho_{s',s''}(x_2))) \\ \Rightarrow_{N'}\ & \delta(i(p_s(x_1)), \overline{q}(\rho_{s',s''}(x_2)), q'(\rho_{s',s''}(x_2))) \end{aligned}$$

is a rule of $M'\,;N'$ for every $s, s', s'' \in \{\alpha, \beta\}$. Note if we had not applied the R_E-step, then we would not have obtained a rule of $M\,;N$ (because we would have obtained the state combination $\langle \overline{q}, p\rangle$ instead of $\langle \overline{q}, \rho_{s',s''}\rangle$, and $\langle \overline{q}, p\rangle$ is not a state of $M'\,;N'$). Let us also construct a rule for the state combination $\langle \overline{q}, \rho_{s',s''}\rangle$.

$$\begin{aligned} & \overline{q}(\rho_{s',s''}(\delta(x_1, x_2, x_3))) \\ \Rightarrow_{M'}\ & \overline{q}(\sigma(p_{s'}(x_1), \sigma(p_{s''}(x_2), p(x_3)))) \\ \Rightarrow_{N'}\ & q'(p_{s'}(x_1)) \end{aligned}$$

Finally, let us construct a rule for the state combination $\langle q'', \rho_{s',s''}\rangle$.

$$\begin{aligned} & q''(\rho_{s',s''}(\delta(x_1, x_2, x_3))) \\ \Rightarrow_{M'}\ & \overline{q}(\sigma(p_{s'}(x_1), \sigma(p_{s''}(x_2), p(x_3)))) \\ \Rightarrow_{R_E}\ & \overline{q}(\sigma(p_{s'}(x_1), \sigma(p_{s''}(x_2), \rho_s(x_3)))) \\ \Rightarrow_{N'}\ & q(\sigma(p_{s''}(x_2), \rho_s(x_3))) \\ \Rightarrow_{N'}\ & \delta(q'(p_{s''}(x_1)), \overline{q}(\rho_s(x_2)), q''(\rho_s(x_2))) \end{aligned}$$

for every $s \in \{\alpha, \beta\}$. □

After having pre-processed the XTOPs in our introductory example, the devices M and N' can be composed into $M\,;N'$. One rule of the composed XTOP is illustrated in Figure 13.

6 Post-processing

Finally, we will compose rules again in an effort to restore linearity (and nondeletion). Since the composition of two linear and nondeleting XTOPs cannot always be computed by a single XTOP (Arnold and Dauchet, 1982), this method can fail to return such an XTOP. The presented method is not a characterization, which means it might even fail to return a linear and nondeleting XTOP although an equivalent linear and nondeleting XTOP exists. However, in a significant number of examples, the recombination succeeds to rebuild a linear (and nondeleting) XTOP.

Let $M'\,;N' = (S, \Sigma, \Gamma, I, R)$ be the composed XTOP constructed in Section 5. We simply inspect each non-linear rule (i.e., each rule with a non-linear right-hand side) and expand it by all rule options at the copied variables. Since the method is pretty standard and variants have already been used in the pre-processing steps, we only illustrate it on the rules of Figure 12.

Example 9. The first (top row, left-most) rule of Figure 12 is non-linear in the variable y_2. Thus, we expand the calls $\langle \overline{q}, \rho\rangle(y_2)$ and $\langle q', \rho\rangle(y_2)$. If $\rho = \rho_s$ for some $s \in \{\alpha, \beta\}$, then the next rules are uniquely determined and we obtain the rule displayed in Figure 14. Here the expansion was successful and we could delete the original rule for $\rho = \rho_s$ and replace it by the displayed expanded rule. However, if $\rho = \rho'_{s',s''}$, then we can also expand the rule to obtain the rule displayed in Figure 15. It is still copying and we could repeat the process of expansion here, but we cannot get rid of all copying rules using this approach (as expected since there is no linear XTOP computing the same tree transformation). □

References

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

Brenda S. Baker. 1979. Composition of top-down and bottom-up tree transductions. *Information and Control*, 41(2):186–213.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. ACL*, pages 205–208. Association for Computational Linguistics.

Joost Engelfriet, Eric Lilin, and Andreas Maletti. 2009. Composition and decomposition of extended multi bottom-up tree transducers. *Acta Informatica*, 46(8):561–590.

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

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

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

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

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

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

Eric Lilin. 1978. *Une généralisation des transducteurs d'états finis d'arbres: les S-transducteurs*. Thèse 3ème cycle, Université de Lille.

Andreas Maletti and Heiko Vogler. 2010. Compositions of top-down tree transducers with ε-rules. In *Proc. FSMNLP*, volume 6062 of LNAI, pages 69–80. Springer.

Andreas Maletti. 2010. Why synchronous tree substitution grammars? In *Proc. HLT-NAACL*, pages 876–884. Association for Computational Linguistics.

Andreas Maletti. 2011. An alternative to synchronous tree substitution grammars. *Natural Language Engineering*, 17(2):221–242.

Alberto Martelli and Ugo Montanari. 1982. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282.

Jonathan May and Kevin Knight. 2006. Tiburon: A weighted tree automata toolkit. In *Proc. CIAA*, volume 4094 of LNCS, pages 102–113. Springer.

Jonathan May, Kevin Knight, and Heiko Vogler. 2010. Efficient inference through cascades of weighted tree transducers. In *Proc. ACL*, pages 1058–1066. Association for Computational Linguistics.

Jonathan May. 2010. *Weighted Tree Automata and Transducers for Syntactic Natural Language Processing*. Ph.D. thesis, University of Southern California, Los Angeles.

John Alan Robinson. 1965. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41.

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

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

James W. Thatcher. 1970. Generalized^2 sequential machine maps. *Journal of Computer and System Sciences*, 4(4):339–367.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. ACL*, pages 523–530. Association for Computational Linguistics.