How to train your multi bottom-up tree transducer

Andreas Maletti
Universität Stuttgart, Institute for Natural Language Processing
Azenbergstraße 12, 70174 Stuttgart, Germany
andreas.maletti@ims.uni-stuttgart.de

Abstract

The local multi bottom-up tree transducer is introduced and related to the (non-contiguous) synchronous tree sequence substitution grammar. It is then shown how to obtain a weighted local multi bottom-up tree transducer from a bilingual and biparsed corpus. Finally, the problem of non-preservation of regularity is addressed. Three properties that ensure preservation are introduced, and it is discussed how to adjust the rule extraction process such that they are automatically fulfilled.

1 Introduction

A (formal) translation model is at the core of every machine translation system. Predominantly, statistical processes are used to instantiate the formal model and derive a specific translation device. Brown et al. (1990) discuss automatically trainable translation models in their seminal paper. However, the IBM models of Brown et al. (1993) are string-based in the sense that they base the translation decision on the words and their surrounding context. Contrary, in the field of syntax-based machine translation, the translation models have full access to the syntax of the sentences and can base their decision on it. A good exposition to both fields is presented in (Knight, 2007).

In this paper, we deal exclusively with syntax-based translation models such as synchronous tree substitution grammars (STSG), multi bottom-up tree transducers (MBOT), and synchronous tree-sequence substitution grammars (STSSG). Chiang (2006) gives a good introduction to STSG, which originate from the syntax-directed translation schemes of Aho and Ullman (1972). Roughly speaking, an STSG has rules in which two linked nonterminals are replaced (at the same time) by two corresponding trees containing terminal and nonterminal symbols. In addition, the nonterminals in the two replacement trees are linked, which creates new linked nonterminals to which further rules can be applied. Henceforth, we refer to these two trees as input and output tree. MBOT have been introduced in (Arnold and Dauchet, 1982; Lilin, 1981) and are slightly more expressive than STSG. Roughly speaking, they allow one replacement input tree and several output trees in a single rule. This change and the presence of states yields many algorithmically advantageous properties such as closure under composition, efficient binarization, and efficient input and output restriction [see (Maletti, 2010)]. Finally, STSSG, which have been derived from rational tree relations (Raoult, 1997), have been discussed by Zhang et al. (2008a), Zhang et al. (2008b), and Sun et al. (2009). They are even more expressive than the local variant of the multi bottom-up tree transducer (LMBOT) that we introduce here and can have several input and output trees in a single rule.

In this contribution, we restrict MBOT to a form that is particularly relevant in machine translation. We drop the general state behavior of MBOT and replace it by the common locality tests that are also present in STSG, STSSG, and STAG (Shieber and Schabes, 1990; Shieber, 2007). The obtained device is the local MBOT (LMBOT).

Maletti (2010) argued the algorithmical advantages of MBOT over STSG and proposed MBOT as an implementation alternative for STSG. In particular, the training procedure would train STSG; i.e., it would not utilize the additional expressive power

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 825–834,
Portland, Oregon, June 19-24, 2011.

of MBOT. However, Zhang et al. (2008b) and Sun et al. (2009) demonstrate that the additional expressivity gained from non-contiguous rules greatly improves the translation quality. In this contribution we address this separation and investigate a training procedure for LMBOT that allows non-contiguous fragments while preserving the algorithmic advantages of MBOT. To this end, we introduce a rule extraction and weight training method for LMBOT that is based on the corresponding procedures for STSG and STSSG. However, general LMBOT can be too expressive in the sense that they allow translations that do not preserve regularity. Preservation of regularity is an important property for efficient representations and efficient algorithms [see (May et al., 2010)]. Consequently, we present 3 properties that ensure that an LMBOT preserves regularity. In addition, we shortly discuss how these properties could be enforced in the rule extraction procedure.

2 Notation

The set of nonnegative integers is $\mathbb{N}$. We write $[k]$ for the set $\{i \mid 1 \leq i \leq k\}$. We treat functions as special relations. For every relation $R \subseteq A \times B$ and $S \subseteq A$, we write

$$R(S) = \{b \in B \mid \exists a \in S\colon (a,b) \in R\}$$
$$R^{-1} = \{(b,a) \mid (a,b) \in R\} \ ,$$

where R^{-1} is called the *inverse* of R.

Given an alphabet Σ, the set of all words (or sequences) over Σ is Σ^*, of which the empty word is ε. The concatenation of two words u and w is simply denoted by the juxtaposition uw. The length of a word $w = \sigma_1 \cdots \sigma_k$ with $\sigma_i \in \Sigma$ for all $i \in [k]$ is $|w| = k$. Given $1 \leq i \leq j \leq k$, the (i,j)-*span* $w[i,j]$ of w is $\sigma_i \sigma_{i+1} \cdots \sigma_j$.

The set T_Σ of all Σ-*trees* is the smallest set T such that $\sigma(\mathbf{t}) \in T$ for all $\sigma \in \Sigma$ and $\mathbf{t} \in T^*$. We generally use bold-face characters (like $\mathbf{t}$) for sequences, and we refer to their elements using subscripts (like t_i). Consequently, a tree t consists of a labeled root node σ followed by a sequence $\mathbf{t}$ of its children. To improve readability we sometimes write a sequence $t_1 \cdots t_k$ as $t_1, \ldots, t_k$.

The *positions* $\mathrm{pos}(t) \subseteq \mathbb{N}^*$ of a tree $t = \sigma(\mathbf{t})$ are inductively defined by $\mathrm{pos}(t) = \{\varepsilon\} \cup \mathrm{pos}(\mathbf{t})$, where

$$\mathrm{pos}(\mathbf{t}) = \bigcup_{1 \leq i \leq |\mathbf{t}|} \{ip \mid p \in \mathrm{pos}(t_i)\} \ .$$

Note that this yields an undesirable difference between $\mathrm{pos}(t)$ and $\mathrm{pos}(\mathbf{t})$, but it will always be clear from the context whether we refer to a single tree or a sequence. Note that positions are ordered via the (standard) lexicographic ordering. Let $t \in T_\Sigma$ and $p \in \mathrm{pos}(t)$. The label of t at position p is $t(p)$, and the subtree rooted at position p is $t|_p$. Formally, they are defined by

$$t(p) = \begin{cases} \sigma & \text{if } p = \varepsilon \\ \mathbf{t}(p) & \text{otherwise} \end{cases} \qquad \mathbf{t}(ip) = t_i(p)$$

$$t|_p = \begin{cases} t & \text{if } p = \varepsilon \\ \mathbf{t}|_p & \text{otherwise} \end{cases} \qquad \mathbf{t}|_{ip} = t_i|_p$$

for all $t = \sigma(\mathbf{t})$ and $1 \leq i \leq |\mathbf{t}|$. As demonstrated, these notions are also used for sequences. A position $p \in \mathrm{pos}(t)$ is a *leaf* (in t) if $p1 \notin \mathrm{pos}(t)$. Given a subset $\mathrm{NT} \subseteq \Sigma$, we let

$$\downarrow_{\mathrm{NT}}(t) = \{p \in \mathrm{pos}(t) \mid t(p) \in \mathrm{NT},\ p \text{ leaf in } t\} \ .$$

Later NT will be the set of nonterminals, so that the elements of $\downarrow_{\mathrm{NT}}(t)$ will be the *leaf nonterminals* of t. We extend the notion to sequences $\mathbf{t}$ by

$$\downarrow_{\mathrm{NT}}(\mathbf{t}) = \bigcup_{1 \leq i \leq |\mathbf{t}|} \{ip \mid p \in \downarrow_{\mathrm{NT}}(t_i)\} \ .$$

We also need a substitution that replaces subtrees. Let $p_1, \ldots, p_n \in \mathrm{pos}(t)$ be pairwise incomparable positions and $t_1, \ldots, t_n \in T_\Sigma$. Then $t[p_i \leftarrow t_i \mid 1 \leq i \leq n]$ denotes the tree that is obtained from t by replacing (in parallel) the subtrees at p_i by t_i for every $i \in [k]$.

Finally, let us recall regular tree languages. A *finite tree automaton* M is a tuple (Q, Σ, δ, F) such that Q is a finite set, $\delta \subseteq Q^* \times \Sigma \times Q$ is a finite relation, and $F \subseteq Q$. We extend δ to a mapping $\underline{\delta}\colon T_\Sigma \to 2^Q$ by

$$\underline{\delta}(\sigma(\mathbf{t})) = \{q \mid (\mathbf{q}, \sigma, q) \in \delta, \forall i \in [\,|\mathbf{t}|\,]\colon q_i \in \underline{\delta}(t_i)\}$$

for every $\sigma \in \Sigma$ and $\mathbf{t} \in T_\Sigma^*$. The finite tree automaton M recognizes the tree language

$$L(M) = \{t \in T_\Sigma \mid \underline{\delta}(t) \cap F \neq \emptyset\} \ .$$

A tree language $L \subseteq T_\Sigma$ is *regular* if there exists a finite tree automaton M such that $L = L(M)$.

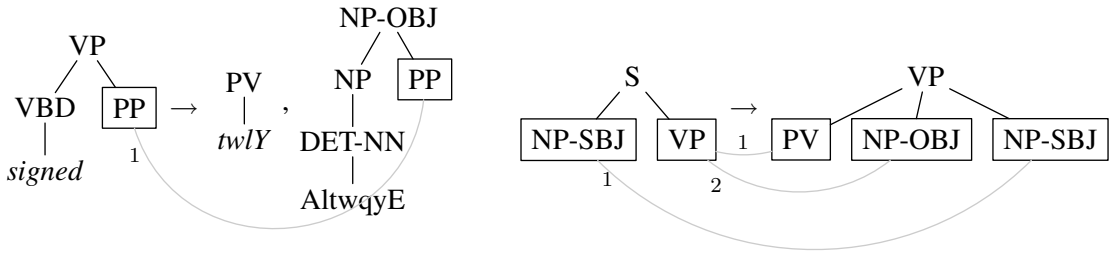

Figure 1: Sample LMBOT rules.

3 The model

In this section, we recall particular multi bottom-up tree transducers, which have been introduced by Arnold and Dauchet (1982) and Lilin (1981). A detailed (and English) presentation of the general model can be found in Engelfriet et al. (2009) and Maletti (2010). Using the nomenclature of Engelfriet et al. (2009), we recall a variant of *linear and nondeleting extended multi bottom-up tree transducers* (MBOT) here. Occasionally, we will refer to general MBOT, which differ from the local variant discussed here because they have explicit states.

Throughout the article, we assume sets Σ and Δ of input and output symbols, respectively. Moreover, let $\mathrm{NT} \subseteq \Sigma \cup \Delta$ be the set of designated nonterminal symbols. Finally, we avoid weights in the formal development to keep it simple. It is straightforward to add weights to our model.

Essentially, the model works on pairs $\langle t, \mathbf{u} \rangle$ consisting of an input tree $t \in T_\Sigma$ and a sequence $\mathbf{u} \in T_\Delta^*$ of output trees. Each such pair is called a *pre-translation* and the rank $\mathrm{rk}(\langle t, \mathbf{u} \rangle)$ the pre-translation $\langle t, \mathbf{u} \rangle$ is $|\mathbf{u}|$. In other words, the rank of a pre-translation equals the number of output trees stored in it. Given a pre-translation $\langle t, \mathbf{u} \rangle \in T_\Sigma \times T_\Delta^k$ and $i \in [k]$, we call u_i the i^{th} translation of t. An alignment for the pre-translation $\langle t, \mathbf{u} \rangle$ is an injective mapping $\psi \colon \downarrow_{\mathrm{NT}}(\mathbf{u}) \to \downarrow_{\mathrm{NT}}(t) \times \mathbb{N}$ such that $(p, j) \in \psi(\downarrow_{\mathrm{NT}}(\mathbf{u}))$ for every $(p, i) \in \psi(\downarrow_{\mathrm{NT}}(\mathbf{u}))$ and $j \in [i]$. In other words, an alignment should request each translation of a particular subtree at most once and if it requests the i^{th} translation, then it should also request all previous translations.

Definition 1 *A* local multi bottom-up tree transducer *(*LMBOT*) is a finite set R of* rules *such that every rule, written $l \to_\psi \mathbf{r}$, contains a pre-translation $\langle l, \mathbf{r} \rangle$ and an alignment ψ for it.*

The component l is the *left-hand side*, $\mathbf{r}$ is the *right-hand side*, and ψ is the *alignment* of a rule $l \to_\psi \mathbf{r} \in R$. The rules of an LMBOT are similar to the rules of an STSG (synchronous tree substitution grammar) of Eisner (2003) and Shieber (2004), but right-hand sides of LMBOT contain a sequence of trees instead of just a single tree as in an STSG. In addition, the alignments in an STSG rule are bijective between leaf nonterminals, whereas our model permits multiple alignments to a single leaf nonterminal in the left-hand side. A model that is even more powerful than LMBOT is the non-contiguous version of STSSG (synchronous tree-sequence substitution grammar) of Zhang et al. (2008a), Zhang et al. (2008b), and Sun et al. (2009), which allows sequences of trees on both sides of rules [see also (Raoult, 1997)]. Figure 1 displays sample rules of an LMBOT using a graphical representation of the trees and the alignment.

Next, we define the semantics of an LMBOT R. To avoid difficulties[1], we explicitly exclude rules like $l \to_\psi \mathbf{r}$ where $l \in \mathrm{NT}$ or $\mathbf{r} \in \mathrm{NT}^*$; i.e., rules where the left- or right-hand side are only leaf nonterminals. We first define the traditional bottom-up semantics. Let $\rho = l \to_\psi \mathbf{r} \in R$ be a rule and $p \in \downarrow_{\mathrm{NT}}(l)$. The p-rank $\mathrm{rk}(\rho, p)$ of ρ is $\mathrm{rk}(\rho, p) = |\{i \in \mathbb{N} \mid (p, i) \in \psi(\downarrow_{\mathrm{NT}}(\mathbf{r}))\}|$.

Definition 2 *The set $\tau(R)$ of pre-translations of an* LMBOT *R is inductively defined to be the smallest set such that: If $\rho = l \to_\psi \mathbf{r} \in R$ is a rule, $\langle t_p, \mathbf{u}_p \rangle \in \tau(R)$ is a pre-translation of R for every $p \in \downarrow_{\mathrm{NT}}(l)$, and*

- $\mathrm{rk}(\rho, p) = \mathrm{rk}(\langle t_p, \mathbf{u}_p \rangle)$,
- $l(p) = t_p(\varepsilon)$, *and*

[1] Actually, difficulties arise only in the weighted setting.

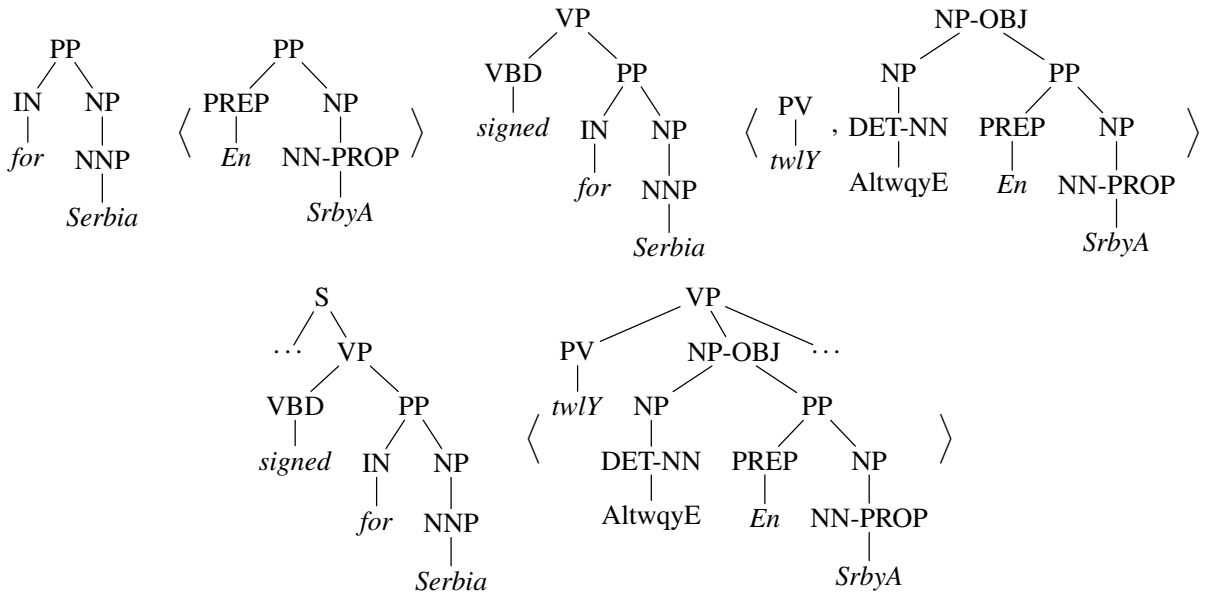

Figure 2: Top left: (a) Initial pre-translation; Top right: (b) Pre-translation obtained from the left rule of Fig. 1 and (a); Bottom: (c) Pre-translation obtained from the right rule of Fig. 1 and (b).

- $\mathbf{r}(p') = \mathbf{u}_{p''}(i)$ *with* $\psi(p') = (p'', i)$

for every $p' \in \downarrow_{\mathrm{NT}}(\mathbf{r})$*, then* $\langle t, \mathbf{u}\rangle \in \tau(R)$ *where*

- $t = l[p \leftarrow t_p \mid p \in \downarrow_{\mathrm{NT}}(l)]$ *and*
- $\mathbf{u} = \mathbf{r}[p' \leftarrow (\mathbf{u}_{p''})_i \mid p' \in \psi^{-1}(p'', i)]$.

In plain words, each nonterminal leaf p in the left-hand side of a rule ρ can be replaced by the input tree t of a pre-translation $\langle t, \mathbf{u}\rangle$ whose root is labeled by the same nonterminal. In addition, the rank $\mathrm{rk}(\rho, p)$ of the replaced nonterminal should match the rank $\mathrm{rk}(\langle t, \mathbf{u}\rangle)$ of the pre-translation and the nonterminals in the right-hand side that are aligned to p should be replaced by the translation that the alignment requests, provided that the nonterminal matches with the root symbol of the requested translation. The main benefit of the bottom-up semantics is that it works exclusively on pre-translations. The process is illustrated in Figure 2.

Using the classical bottom-up semantics, we simply obtain the following theorem by Maletti (2010) because the MBOT constructed there is in fact an LMBOT.

Theorem 3 *For every* STSG*, an equivalent* LMBOT *can be constructed in linear time, which in turn yields a particular* MBOT *in linear time.*

Finally, we want to relate LMBOT to the STSSG of Sun et al. (2009). To this end, we also introduce the top-down semantics for LMBOT. As expected, both semantics coincide. The top-down semantics is introduced using rule compositions, which will play an important rule later on.

Definition 4 *The set* R^k *of* k*-fold composed rules is inductively defined as follows:*

- $R^1 = R$ *and*
- $\ell \to_\varphi \mathbf{s} \in R^{k+1}$ *for all* $\rho = l \to_\psi \mathbf{r} \in R$ *and* $\rho_p = l_p \to_{\psi_p} \mathbf{r}_p \in R^k$ *such that*
  - $\mathrm{rk}(\rho, p) = \mathrm{rk}(\langle l_p, \mathbf{r}_p\rangle)$,
  - $l(p) = l_p(\varepsilon)$*, and*
  - $\mathbf{r}(p') = \mathbf{r}_{p''}(i)$ *with* $\psi(p') = (p'', i)$

  for every $p \in \downarrow_{\mathrm{NT}}(l)$ *and* $p' \in \downarrow_{\mathrm{NT}}(\mathbf{r})$ *where*
  - $\ell = l[p \leftarrow l_p \mid p \in \downarrow_{\mathrm{NT}}(l)]$,
  - $\mathbf{s} = \mathbf{r}[p' \leftarrow (\mathbf{r}_{p''})_i \mid p' \in \psi^{-1}(p'', i)]$*, and*
  - $\varphi(p'p) = p''\psi_{p''}(ip)$ *for all positions* $p' \in \psi^{-1}(p'', i)$ *and* $ip \in \downarrow_{\mathrm{NT}}(\mathbf{r}_{p''})$.

The rule closure $R^{\leq\infty}$ *of* R *is* $R^{\leq\infty} = \bigcup_{i\geq 1} R^i$*. The top-down pre-translation of* R *is*

$$\tau_{\mathrm{t}}(R) = \{\langle l, \mathbf{r}\rangle \mid l \to_\psi \mathbf{r} \in R^{\leq\infty}, \downarrow_{\mathrm{NT}}(l) = \emptyset\} \ .$$

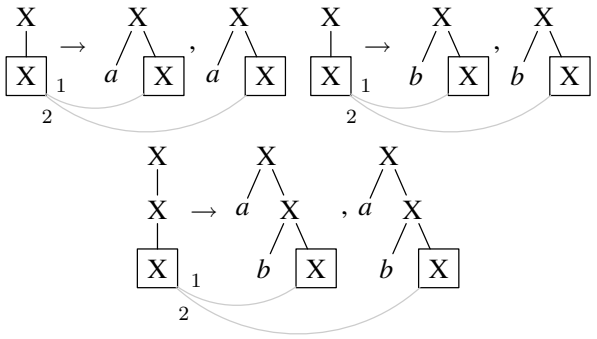

Figure 3: Composed rule.

The composition of the rules, which is illustrated in Figure 3, in the second item of Definition 4 could also be represented as $\rho(\rho_1, \ldots, \rho_k)$ where $\rho_1, \ldots, \rho_k$ is an enumeration of the rules $\{\rho_p \mid p \in \downarrow_{\mathrm{NT}}(l)\}$ used in the item. The following theorem is easy to prove.

Theorem 5 *The bottom-up and top-down semantics coincide; i.e.,* $\tau(R) = \tau_{\mathrm{t}}(R)$.

Chiang (2005) and Graehl et al. (2008) argue that STSG have sufficient expressive power for syntax-based machine translation, but Zhang et al. (2008a) show that the additional expressive power of tree-sequences helps the translation process. This is mostly due to the fact that smaller (and less specific) rules can be extracted from bi-parsed word-aligned training data. A detailed overview that focusses on STSG is presented by Knight (2007).

Theorem 6 *For every* LMBOT*, an equivalent* STSSG *can be constructed in linear time.*

4 Rule extraction and training

In this section, we will show how to automatically obtain an LMBOT from a bi-parsed, word-aligned parallel corpus. Essentially, the process has two steps: *rule extraction* and *training*. In the rule extraction step, an (unweighted) LMBOT is extracted from the corpus. The rule weights are then set in the training procedure.

The two main inspirations for our rule extraction are the corresponding procedures for STSG (Galley et al., 2004; Graehl et al., 2008) and for STSSG (Sun et al., 2009). STSG are always contiguous in both the left- and right-hand side, which means that they (completely) cover a single span of input or output words. On the contrary, STSSG rules can be non-contiguous on both sides, but the extraction procedure of Sun et al. (2009) only extracts rules that are contiguous on the left- or right-hand side. We can adjust its 1^{st} phase that extracts rules with (potentially) non-contiguous right-hand sides. The adjustment is necessary because LMBOT rules cannot have (contiguous) tree sequences in their left-hand sides. Overall, the rule extraction process is sketched in Algorithm 1.

Algorithm 1 Rule extraction for LMBOT

Require: word-aligned tree pair (t, u)
Return: LMBOT rules R such that $(t, u) \in \tau(R)$
  while there exists a maximal non-leaf node $p \in \mathrm{pos}(t)$ and minimal $p_1, \ldots, p_k \in \mathrm{pos}(u)$ such that $t|_p$ and $(u|_{p_1}, \ldots, u|_{p_k})$ have a consistent alignment (i.e., no alignments from within $t|_p$ to a leaf outside $(u|_{p_1}, \ldots, u|_{p_k})$ and vice versa)
  do
2: add rule $\rho = t|_p \to_\psi (u_{p_1}, \ldots, u_{p_k})$ to R with the nonterminal alignments ψ
      `// excise rule` ρ `from` (t, u)
4: $t \leftarrow t[p \leftarrow t(p)]$
    $u \leftarrow u[p_i \leftarrow u(p_i) \mid i \in \{1, \ldots, k\}]$
6: establish alignments according to position
  end while

The requirement that we can only have one input tree in LMBOT rules indeed might cause the extraction of bigger and less useful rules (when compared to the corresponding STSSG rules) as demonstrated in (Sun et al., 2009). However, the stricter rule shape preserves the good algorithmic properties of LMBOT. The more powerful STSSG rules can cause nonclosure under composition (Raoult, 1997; Radmacher, 2008) and parsing to be less efficient.

Figure 4 shows an example of biparsed aligned parallel text. According to the method of Galley et al. (2004) we can extract the (minimal) STSG rule displayed in Figure 5. Using the more liberal format of LMBOT rules, we can decompose the STSG rule of Figure 5 further into the rules displayed in Figure 1. The method of Sun et al. (2009) would also extract the rule displayed in Figure 6.

Let us reconsider Figures 1 and 2. Let ρ_1 be the top left rule of Figure 2 and ρ_2 and ρ_3 be the

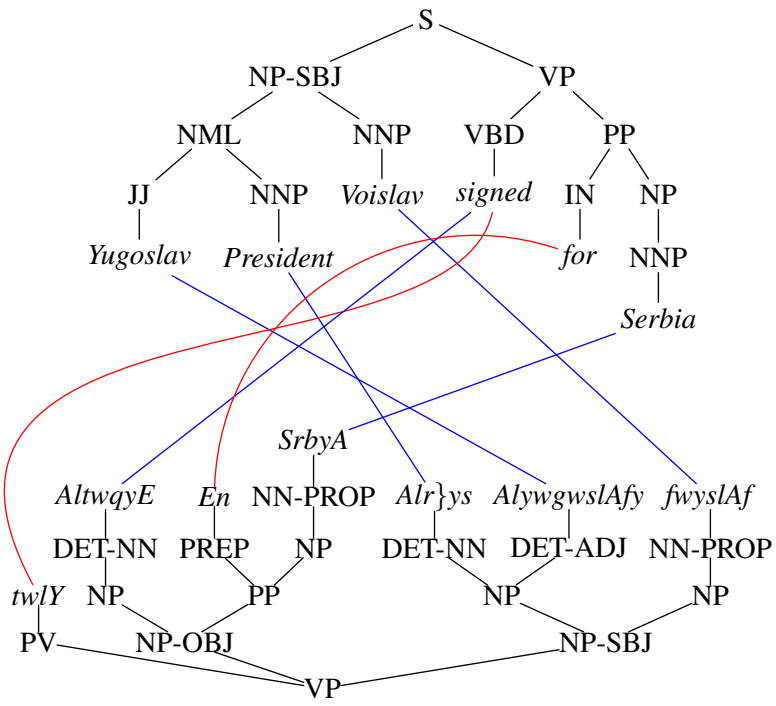

Figure 4: Biparsed aligned parallel text.

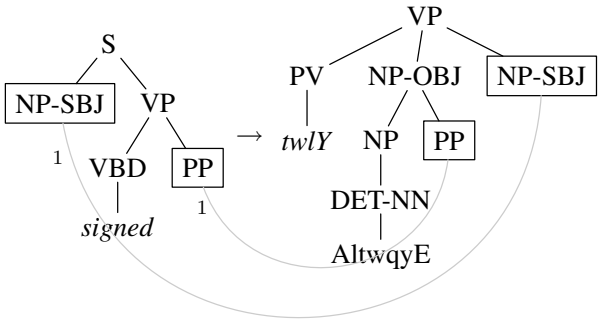

Figure 5: Minimal STSG rule.

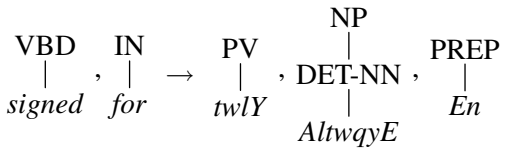

Figure 6: Sample STSSG rule.

left and right rule of Figure 1, respectively. We can represent the lower pre-translation of Figure 2 by $\rho_3(\cdots, \rho_2(\rho_1))$, where $\rho_2(\rho_1)$ represents the upper right pre-translation of Figure 2. If we name all rules of R, then we can represent each pre-translation of $\tau(R)$ symbolically by a tree containing rule names. Such trees containing rule names are often called *derivation trees*. Overall, we obtain the following result, for which details can be found in (Arnold and Dauchet, 1982).

Theorem 7 *The set $D(R)$ is a regular tree language for every* LMBOT *R, and the set of derivations is also regular for every* MBOT.

Moreover, using the input and output product constructions of Maletti (2010) we obtain that even the set $D_{t,u}(R)$ of derivations for a specific input tree t and output tree u is regular. Since $D_{t,u}(R)$ is regular, we can compute the inside and outside weight of each (weighted) rule of R following the method of Graehl et al. (2008). Similarly, we can adjust the training procedure of Graehl et al. (2008), which yields that we can automatically obtain a weighted LMBOT from a bi-parsed parallel corpus. Details on the run-time can be found in (Graehl et al., 2008).

5 Preservation of regularity

Clearly, LMBOT are not symmetric. Although, the backwards application of an LMBOT preserves regularity, this property does not hold for forward application. We will focus on forward application here. Given a set $\mathcal{T}$ of pre-translations and a tree language

$L \subseteq T_\Sigma$, we let

$$\mathcal{T}_{\mathrm{c}}(L) = \{u_i \mid (u_1, \ldots, u_k) \in \mathcal{T}(L), i \in [k]\} \ ,$$

which collects all translations of input trees in L. We say that $\mathcal{T}$ *preserves regularity* if $\mathcal{T}_{\mathrm{c}}(L)$ is regular for every regular tree language $L \subseteq T_\Sigma$. Correspondingly, an LMBOT R preserves regularity if its set $\tau(R)$ of pre-translations preserves regularity.

As mentioned, an LMBOT does not necessarily preserve regularity. The rules of an LMBOT have only alignments between the left-hand side (input tree) and the right-hand side (output tree), which are also called *inter-tree* alignments. However, several alignments to a single nonterminal in the left-hand side can transitively relate two different nonterminals in the output side and thus simulate an *intra-tree* alignment. For example, the right rule of Figure 1 relates a 'PV' and an 'NP-OBJ' node to a single 'VP' node in the left-hand side. This could lead to an intra-tree alignment (synchronization) between the 'PV' and 'NP-OBJ' nodes in the right-hand side.

Figure 7 displays the rules R of an LMBOT that does not preserve regularity. This can easily be seen on the leaf (word) languages because the LMBOT can translate the word x to any element of $L = \{wcwc \mid w \in \{a, b\}^*\}$. Clearly, this word language L is not context-free. Since the leaf language of every regular tree language is context-free and regular tree languages are closed under intersection (needed to single out the translations that have the symbol Y at the root), this also proves that $\tau(R)_{\mathrm{c}}(T_\Sigma)$ is not regular. Since T_Σ is regular, this proves that the LMBOT does not preserve regularity.

Preservation of regularity is an important property for a number of translation model manipulations. For example, the bucket-brigade and the on-the-fly method for the efficient inference described in (May et al., 2010) essentially build on it. Moreover, a regular tree grammar (i.e., a representation of a regular tree language) is an efficient representation. More complex representations such as context-free tree grammars [see, e.g., (Fujiyoshi, 2004)] have worse algorithmic properties (e.g., more complex parsing and problematic intersection).

In this section, we investigate three syntactic restrictions on the set R of rules that guarantees that the obtained LMBOT preserves regularity. Then we shortly discuss how to adjust the rule extraction algorithm, so that the extracted rules automatically have these property. First, we quickly recall the notion of composed rules from Definition 4 because it will play an essential role in all three properties. Figure 3 shows a composition of two rules from Figure 7. Mind that R^2 might not contain all rules of R, but it contains all those without leaf nonterminals.

Definition 8 *An* LMBOT R *is* finitely collapsing *if there is* $n \in \mathbb{N}$ *such that* $\psi\colon \downarrow_{\mathrm{NT}}(\mathbf{r}) \to \downarrow_{\mathrm{NT}}(l) \times \{1\}$ *for every rule* $l \to_\psi \mathbf{r} \in R^n$.

The following statement follows from a more general result of Raoult (1997), which we will introduce with our second property.

Theorem 9 *Every finitely collapsing* LMBOT *preserves regularity.*

Often the simple condition 'finitely collapsing' is fulfilled after rule extraction. In addition, it is automatically fulfilled in an LMBOT that was obtained from an STSG using Theorem 3. It can also be ensured in the rule extraction process by introducing *collapsing points* for output symbols that can appear recursively in the corpus. For example, we could enforce that all extracted rules for clause-level output symbols (assuming that there is no recursion not involving a clause-level output symbols) should have only 1 output tree in the right-hand side.

However, 'finitely collapsing' is a rather strict property. Finitely collapsing LMBOT have only slightly more expressive power than STSG. In fact, they could be called STSG with input desynchronization. This is due to the fact that the alignment in composed rules establishes an injective relation between leaf nonterminals (as in an STSG), but it need not be bijective. Consequently, there can be leaf nonterminals in the left-hand side that have no aligned leaf nonterminal in the right-hand side. In this sense, those leaf nonterminals are desynchronized. This feature is illustrated in Figure 8 and such an LMBOT can compute the transformation $\{(t, a) \mid t \in T_\Sigma\}$, which cannot be computed by an STSG (assuming that T_Σ is suitably rich). Thus STSG with input desynchronization are more expressive than STSG, but they still compute a class of transformations that is not closed under composition.

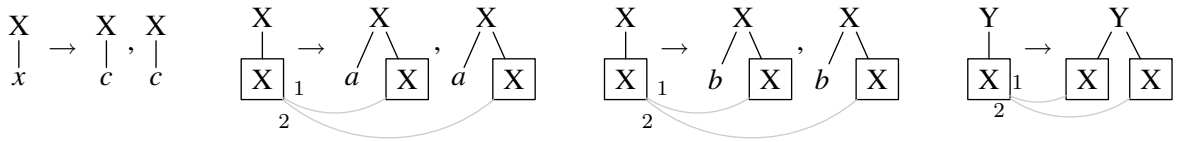

Figure 7: Output subtree synchronization (intra-tree).

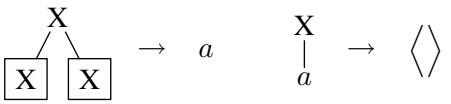

Figure 8: Finitely collapsing LMBOT.

Theorem 10 *For every* STSG*, we can construct an equivalent finitely collapsing* LMBOT *in linear time. Moreover, finitely collapsing* LMBOT *are strictly more expressive than* STSG*.*

Next, we investigate a weaker property by Raoult (1997) that still ensures preservation of regularity.

Definition 11 *An* LMBOT R *has* finite synchronization *if there is* $n \in \mathbb{N}$ *such that for every rule* $l \to_\psi \mathbf{r} \in R^n$ *and* $p \in \downarrow_{\mathrm{NT}}(l)$ *there exists* $i \in \mathbb{N}$ *with* $\psi^{-1}(\{p\} \times \mathbb{N}) \subseteq \{iw \mid w \in \mathbb{N}^*\}$.

In plain terms, multiple alignments to a single leaf nonterminal at p in the left-hand side are allowed, but all leaf nonterminals of the right-hand side that are aligned to p must be in the same tree. Clearly, an LMBOT with finite synchronization is finitely collapsing. Raoult (1997) investigated this restriction in the context of *rational tree relations*, which are a generalization of our LMBOT. Raoult (1997) shows that finite synchronization can be decided. The next theorem follows from the results of Raoult (1997).

Theorem 12 *Every* LMBOT *with finite synchronization preserves regularity.*

MBOT can compute arbitrary compositions of STSG (Maletti, 2010). However, this no longer remains true for MBOT (or LMBOT) with finite synchronization.[2] In Figure 9 we illustrate a translation that can be computed by a composition of two STSG, but that cannot be computed by an MBOT (or LMBOT) with finite synchronization. Intuitively, when processing the chain of 'X's of the transformation depicted in Figure 9, the first and second successor of the 'Z'-node at the root on the output side must be aligned to the 'X'-chain. This is necessary because those two mentioned subtrees must reproduce t_1 and t_2 from the end of the 'X'-chain. We omit the formal proof here, but obtain the following statement.

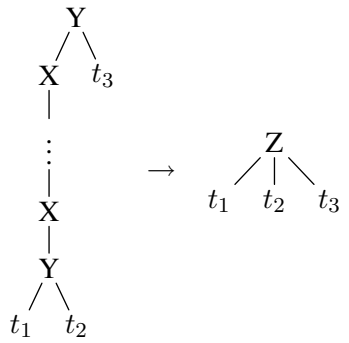

Figure 9: Transformation that cannot be computed by an MBOT with finite synchronization.

Theorem 13 *For every* STSG*, we can construct an equivalent* LMBOT *with finite synchronization in linear time.* LMBOT *and* MBOT *with finite synchronization are strictly more expressive than* STSG *and compute classes that are not closed under composition.*

Again, it is straightforward to adjust the rule extraction algorithm by the introduction of *synchronization points* (for example, for clause level output symbols). We can simply require that rules extracted for those selected output symbols fulfill the condition mentioned in Definition 11.

Finally, we introduce an even weaker version.

Definition 14 *An* LMBOT R *is* copy-free *if there is* $n \in \mathbb{N}$ *such that for every rule* $l \to_\psi \mathbf{r} \in R^n$ *and* $p \in \downarrow_{\mathrm{NT}}(l)$ *we have (i)* $\psi^{-1}(\{p\} \times \mathbb{N}) \subseteq \mathbb{N}$*, or (ii)* $\psi^{-1}(\{p\} \times \mathbb{N}) \subseteq \{iw \mid w \in \mathbb{N}^*\}$ *for an* $i \in \mathbb{N}$*.*

Intuitively, a copy-free LMBOT has rules whose right hand sides may use all leaf nonterminals that are aligned to a given leaf nonterminal in the left-hand side directly at the root (of one of the trees

[2]This assumes a straightforward generalization of the 'finite synchronization' property for MBOT.

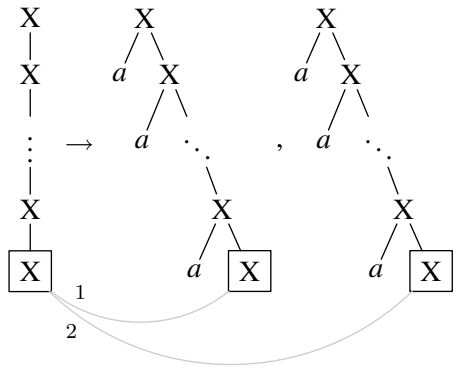

Figure 10: Composed rule that is not copy-free.

in the right-hand side forest) or group all those leaf nonterminals in a single tree in the forest. Clearly, the LMBOT of Figure 7 is not copy-free because the second rule composes with itself (see Figure 10) to a rule that does not fulfill the copy-free condition.

Theorem 15 *Every copy-free* LMBOT *preserves regularity.*

Proof sketch: Let n be the integer of Definition 14. We replace the LMBOT with rules R by the equivalent LMBOT M with rules R^n. Then all rules have the form required in Definition 14. Moreover, let $L \subseteq T_\Sigma$ be a regular tree language. Then we can construct the input product of $\tau(M)$ with L. In this way, we obtain an MBOT M', whose rules still fulfill the requirements (adapted for MBOT) of Definition 14 because the input product does not change the structure of the rules (it only modifies the state behavior). Consequently, we only need to show that the range of the MBOT M' is regular. This can be achieved using a decomposition into a relabeling, which clearly preserves regularity, and a deterministic finite-copying top-down tree transducer (Engelfriet et al., 1980; Engelfriet, 1982). □

Figure 11 shows some relevant rules of a copy-free LMBOT that computes the transformation of Figure 9. Clearly, copy-free LMBOT are more general than LMBOT with finite synchronization, so we again can obtain copy-free LMBOT from STSG. In addition, we can adjust the rule extraction process using synchronization points as for LMBOT with finite synchronization using the restrictions of Definition 14.

Theorem 16 *For every* STSG*, we can construct an equivalent copy-free* LMBOT *in linear time.*

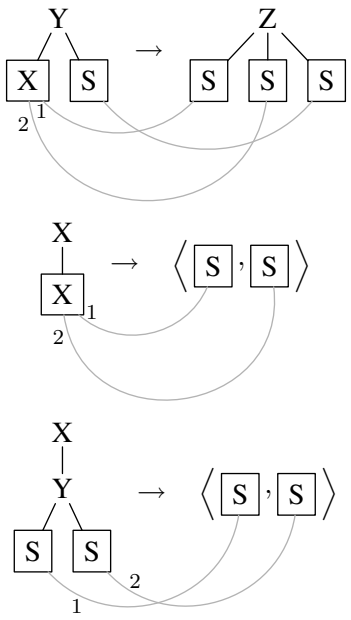

Figure 11: Copy-free LMBOT for the transformation of Figure 9.

Copy-free LMBOT *are strictly more expressive than* LMBOT *with finite synchronization.*

6 Conclusion

We have introduced a simple restriction of multi bottom-up tree transducers. It abstracts from the general state behavior of the general model and only uses the locality tests that are also present in STSG, STSSG, and STAG. Next, we introduced a rule extraction procedure and a corresponding rule weight training procedure for our LMBOT. However, LMBOT allow translations that do not preserve regularity, which is an important property for efficient algorithms. We presented 3 properties that ensure that regularity is preserved. In addition, we shortly discussed how these properties could be enforced in the presented rule extraction procedure.

Acknowledgements

The author gratefully acknowledges the support by KEVIN KNIGHT, who provided the inspiration and the data. JONATHAN MAY helped in many fruitful discussions.

The author was financially supported by the German Research Foundation (DFG) grant MA / 4959 / 1-1.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice Hall.

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

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. Mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*, pages 263–270. Association for Computational Linguistics.

David Chiang. 2006. An introduction to synchronous grammars. In *Proc. ACL*. Association for Computational Linguistics. Part of a tutorial given with Kevin Knight.

Jason Eisner. 2003. Simpler and more general minimization for weighted finite-state automata. In *Proc. NAACL*, pages 64–71. Association for Computational Linguistics.

Joost Engelfriet, Grzegorz Rozenberg, and Giora Slutzki. 1980. Tree transducers, L systems, and two-way machines. *J. Comput. System Sci.*, 20(2):150–202.

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

Joost Engelfriet. 1982. The copying power of one-state tree transducers. *J. Comput. System Sci.*, 25(3):418–435.

Akio Fujiyoshi. 2004. Restrictions on monadic context-free tree grammars. In *Proc. CoLing*, pages 78–84. Association for Computational Linguistics.

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

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

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

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

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

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.

Frank G. Radmacher. 2008. An automata theoretic approach to rational tree relations. In *Proc. SOFSEM*, volume 4910 of LNCS, pages 424–435. Springer.

Jean-Claude Raoult. 1997. Rational tree relations. *Bull. Belg. Math. Soc. Simon Stevin*, 4(1):149–176.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proc. CoLing*, volume 3, pages 253–258. Association for Computational Linguistics.

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

Stuart M. Shieber. 2007. Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In *Proc. SSST*, pages 88–95. Association for Computational Linguistics.

Jun Sun, Min Zhang, and Chew Lim Tan. 2009. A non-contiguous tree sequence alignment-based model for statistical machine translation. In *Proc. ACL*, pages 914–922. Association for Computational Linguistics.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008a. A tree sequence alignment-based tree-to-tree translation model. In *Proc. ACL*, pages 559–567. Association for Computational Linguistics.

Min Zhang, Hongfei Jiang, Haizhou Li, Aiti Aw, and Sheng Li. 2008b. Grammar comparison study for translational equivalence modeling and statistical machine translation. In *Proc. CoLing*, pages 1097–1104. Association for Computational Linguistics.