On the String Translations Produced by Multi Bottom–Up Tree Transducers

Daniel Gildea*
University of Rochester

Tree transducers are defined as relations between trees, but in syntax-based machine translation, we are ultimately concerned with the relations between the strings at the yields of the input and output trees. We examine the formal power of Multi Bottom-Up Tree Transducers from this point of view.

1. Introduction

Many current approaches to syntax-based statistical machine translation fall under the theoretical framework of synchronous tree substitution grammars (STSGs). Tree substitution grammars (TSGs) generalize context-free grammars (CFGs) in that each rule expands a nonterminal to produce an arbitrarily large tree fragment, rather than a fragment of depth one as in a CFG. Synchronous TSGs generate tree fragments in the source and target languages in parallel, with each rule producing a tree fragment in either language. Systems such as that of Galley et al. (2006) extract STSG rules from parallel bilingual text that has been automatically parsed in one language, and the STSG nonterminals correspond to nonterminals in these parse trees. Chiang's (2007) Hiero system produces simpler STSGs with a single nonterminal.

STSGs have the advantage that they can naturally express many re-ordering and restructuring operations necessary for machine translation (MT). They have the disadvantage, however, that they are not closed under composition (Maletti et al. 2009). Therefore, if one wishes to construct an MT system as a pipeline of STSG operations, the result may not be expressible as an STSG. Recently, Maletti (2010) has argued that multi bottom–up tree transducers (MBOTs) (Lilin 1981; Arnold and Dauchet 1982; Engelfriet, Lilin, and Maletti 2009) provide a useful representation for natural language processing applications because they generalize STSGs, but have the added advantage of being closed under composition. MBOTs generalize traditional bottom–up tree transducers in that they allow transducer states to pass more than one output subtree up to subsequent transducer operations. The number of subtrees taken by a state is called its **rank**. MBOTs are linear and non-deleting; that is, operations cannot copy or delete arbitrarily large tree fragments.

Although STSGs and MBOTs both perform operations on trees, it is important to note that, in MT, we are primarily interested in translational relations between strings. Tree operations such as those provided by STSGs are ultimately tools to translate a string

* Computer Science Department, University of Rochester, Rochester NY 14627.
E-mail: gildea@cs.rochester.edu.

Submission received: 3 May 2011; revised submission received: 1 October 2011; accepted for publication: 28 October 2011.

in one natural language into a string in another. Whereas MBOTs originate in the tree transducer literature and are defined to take a tree as input, MT systems such as those of Galley et al. (2006) and Chiang (2007) find a parse of the source language sentence as part of the translation process, and the decoding algorithm, introduced by Yamada and Knight (2002), has more in common with CYK parsing than with simulating a tree transducer.

In this article, we investigate the power of MBOTs, and of compositions of STSGs in particular, in terms of the set of *string* translations that they generate. We relate MBOTs and compositions of STSGs to existing grammatical formalisms defined on strings through five main results, which we outline subsequently. The first four results serve to situate general MBOTs among string formalisms, and the fifth result addresses MBOTs resulting from compositions of STSGs in particular.

Our first result is that the translations produced by MBOTs are a subset of those produced by linear context-free rewriting systems (LCFRSs) (Vijay-Shankar, Weir, and Joshi 1987). LCFRS provides a very general framework that subsumes CFG, tree adjoining grammar (TAG; Joshi, Levy, and Takahashi 1975; Joshi and Schabes 1997), and more complex systems, as well as synchronous context-free grammar (SCFG) (Aho and Ullman 1972) and synchronous tree adjoining grammar (STAG) (Shieber and Schabes 1990; Schabes and Shieber 1994) in the context of translation. LCFRS allows grammar nonterminals to generate more than one span in the final string; the number of spans produced by an LCFRS nonterminal corresponds to the rank of an MBOT state. Our second result states that the translations produced by MBOTs are equivalent to a specific restricted form of LCFRS, which we call 1-m-LCFRS. From the construction relating MBOTs and 1-m-LCFRSs follow results about the source and target sides of the translations produced by MBOTs. In particular, our third result is that the translations produced by MBOTs are context-free within the source language, and hence are strictly less powerful than LCFRSs. This implies that MBOTs are not as general as STAGs, for example. Similarly, MBOTs are not as general as the generalized multitext grammars proposed for machine translation by Melamed (2003), which retain the full power of LCFRSs in each language (Melamed, Satta, and Wellington 2004). Our fourth result is that the output of an MBOT, when viewed as a string language, does retain the full power of LCFRSs. This fact is mentioned by Engelfriet, Lilin, and Maletti (2009, page 586), although no explicit construction is given.

Our final result specifically addresses the string translations that result from compositions of STSGs, with the goal of better understanding the complexity of using such compositions in machine translation systems. We show that the translations produced by compositions of STSGs are more powerful than those produced by single STSGs, or, equivalently, by SCFGs. Although it is known that STSGs are not closed under composition, the proofs used previously in the literature rely on differences in tree structure, and do not generate string translations that cannot be generated by STSG. Our result implies that current approaches to machine translation decoding will need to be extended to handle arbitrary compositions of STSGs.

We now turn to give definitions of MBOTs and LCFRSs in the next section, before presenting our results on general MBOTs in Section 3, and our result on compositions of STSGs in Section 4.

2. Preliminaries

A **ranked alphabet** is an alphabet where each symbol has an integer **rank**, denoting the number of children the symbol takes in a tree. T_Σ denotes the set of trees constructed

from ranked alphabet Σ. We use parentheses to write trees: for example, $a(b, c, d)$ is an element of T_Σ if a is an element of Σ with rank 3, and b, c, and d are elements of Σ with rank 0. Similarly, given a ranked alphabet Σ and a set X, $\Sigma(X)$ denotes the set of trees consisting of a single symbol of Σ of rank k dominating a sequence of k elements from X. We use $T_\Sigma(X)$ to denote the set of arbitrarily sized trees constructed from ranked alphabet Σ having items from set X at some leaf positions. That is, $T_\Sigma(X)$ is the smallest set such that $X \subset T_\Sigma(X)$ and $\sigma(t_1, \ldots, t_k) \in T_\Sigma(X)$ if σ is an element of Σ with rank k, and $t_1, \ldots, t_k \in T_\Sigma(X)$. A **multi bottom–up tree transducer** (MBOT) (Lilin 1981; Arnold and Dauchet 1982; Engelfriet, Lilin, and Maletti 2009; Maletti 2010) is a system $(S, \Sigma, \Delta, F, R)$ where:

- S, Σ, and Δ are ranked alphabets of states, input symbols, and output symbols, respectively.
- $F \subset S$ is a set of accepting states.
- R is a finite set of rules $l \to r$ where, using a set of variables X, $l \in T_\Sigma(S(X))$, and $r \in S(T_\Delta(X))$ such that:
  - every $x \in X$ that occurs in l occurs exactly once in r and vice versa, and
  - $l \notin S(X)$ or $r \notin S(X)$.

One step in an MBOT transduction is performed by rewriting a local tree fragment as specified by one of the rules in R. We replace the fragment l with r, copying the subtree under each variable in l to the location of the corresponding variable in r. Transducer rules apply bottom–up from the leaves of the input tree, as shown in Figure 1, and must terminate in an accepting state. We use underlined symbols for the transducer states, in order to distinguish them from the symbols of the input and output alphabets.

We define a **translation** to be a set of string pairs, and we define the **yield** of an MBOT M to be the set of string pairs (s, t) such that there exist: a tree $s' \in T_\Sigma$ having s as its yield, a tree $t' \in T_\Delta$ having t as its yield, and a transduction from s' to t' that is accepted by M. We refer to s as the **source side** and t as the **target side** of the translation. We use the notation source(T) to denote the set of source strings of a translation T, source(T) $= \{s \mid (s, t) \in T\}$, and we use the notation target(T) to denote the set of target strings. We use the notation yield(MBOT) to denote the set of translations produced by the set of all MBOTs.

A **linear context-free rewriting system** (LCFRS) is defined as a system (V_N, V_T, P, S), where V_N is a set of nonterminal symbols, V_T is a set of terminal symbols, P is a set of productions, and $S \in V_N$ is a distinguished start symbol. Associated with each nonterminal B is a **fan-out** $\varphi(B)$, which tells how many spans B covers in the final string. Productions $p \in P$ take the form: $p : A \to g(B_1, B_2, \ldots, B_r)$, where $A, B_1, \ldots, B_r \in V_N$, and g is a function $g : (V_T^*)^{\varphi(B_1)} \times \cdots \times (V_T^*)^{\varphi(B_r)} \to (V_T^*)^{\varphi(A)}$, which specifies how to assemble the $\sum_{i=1}^{r} \varphi(B_i)$ spans of the righthand side nonterminals into the $\varphi(A)$ spans of the lefthand side nonterminal. The function g must be **linear** and **non-erasing**, which means that if we write

$$g(\langle x_{1,1}, \ldots, x_{1,\varphi(B_1)} \rangle, \ldots, \langle x_{r,1}, \ldots, x_{r,\varphi(B_r)} \rangle) = \langle t_1, \ldots, t_{\varphi(A)} \rangle$$

the tuple of strings $\langle t_1, \ldots, t_{\varphi(A)} \rangle$ on the right-hand side contains each variable $x_{i,j}$ from the left-hand side exactly once, and may also contain terminals from V_T. The process of generating a string from an LCFRS grammar consists of first choosing, top–down, a production to expand each nonterminal, and then, bottom–up, applying the functions

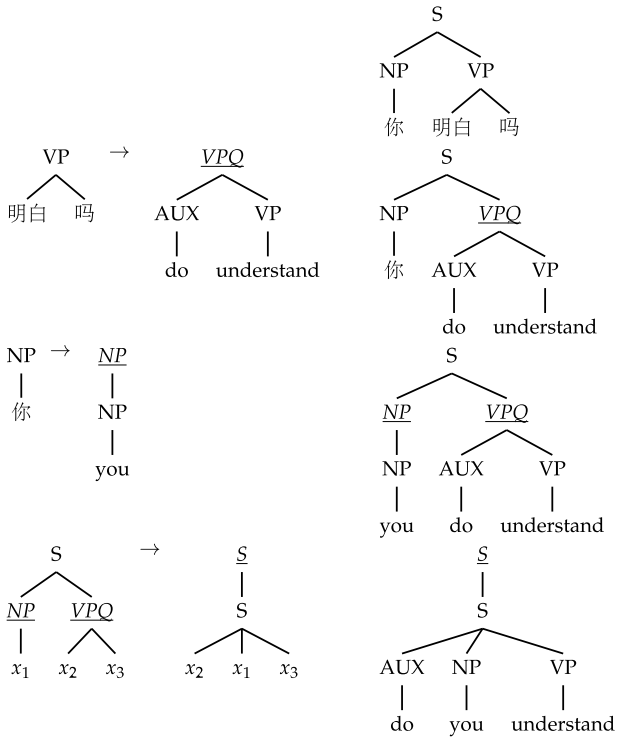


Figure 1
Step-by-step example of an MBOT tree transduction. The left column shows the transducer rule applied at each step; only the last rule contains variables, whereas the others contain alphabet symbols of rank zero at their leaves. State $\underline{VPQ}$ has rank two, and states $\underline{NP}$ and $\underline{S}$ have rank one.

associated with each production to build the string. We refer to the tree induced by top–down nonterminal expansions of an LCFRS as the **derivation tree**, or sometimes simply as a derivation.

As an example of how the LCFRS framework subsumes grammatical formalisms such as CFG, consider the following CFG:

$$S \to A\,B$$
$$A \to a$$
$$B \to b$$

This grammar corresponds to the following grammar in LCFRS notation:

$$S \to g_S(A, B) \qquad g_S(\langle s_A \rangle, \langle s_B \rangle) = \langle s_A s_B \rangle$$
$$A \to g_A() \qquad g_A() = \langle a \rangle$$
$$B \to g_B() \qquad g_B() = \langle b \rangle$$

Here, all nonterminals have fan-out one, reflected in the fact that all tuples defining the productions' functions contain just one string. Just as CFG is equivalent to LCFRS with fan-out 1, SCFG and TAG can be represented as LCFRS with fan-out 2. Higher values of fan-out allow strictly more powerful grammars (Rambow and Satta 1999). Polynomial-time parsing is possible for any fixed LCFRS grammar, but the degree of

the polynomial depends on the grammar. Parsing general LCFRS grammars, where the grammar is considered part of the input, is NP-complete (Satta 1992).

Following Melamed, Satta, and Wellington (2004), we represent translation in LCFRS by using a special symbol # to separate the strings of the two languages. Our LCFRS grammars will only generate strings of the form $s\#t$, where s and t are strings not containing the symbol #, and we will identify s as the source string and t as the target string. We use the notation trans(LCFRS) to denote the set of translations that can be produced by taking the string language of some LCFRS and splitting each string into a pair at the location of the # symbol.

3. Translations Produced by General MBOTs

In this section, we relate the yield of general MBOTs to string rewriting systems.

To begin, we show that the translation produced by any MBOT is also produced by an LCFRS by giving a straightforward construction for converting MBOT rules to LCFRS rules.

We first consider MBOT rules having only variables, as opposed to alphabet symbols of rank zero, at their leaves. For an MBOT rule $l \to r$ with $l \in T_\Sigma(S(X))$, let $S_1, S_2, \ldots, S_k$ be the sequence of states appearing from left to right immediately above the leaves of l. Without loss of generality, we will name the variables such that $x_{i,j}$ is the jth child of the ith state, S_i, and the sequence of variables at the leaves of l, read from left to right, is: $x_{1,1}, \ldots, x_{1,d(S_1)}, \ldots, x_{k,1}, \ldots, x_{k,d(S_k)}$, where $d(S_i)$ is the rank of state S_i. Let S_0 be the state symbol at the root of the right-hand-side (r.h.s.) tree $r \in S(T_\Delta(X))$. Let π and μ be functions such that $x_{\pi(1),\mu(1)}, x_{\pi(2),\mu(2)}, \ldots, x_{\pi(n),\mu(n)}$ is the sequence of variables at the leaves of r read from left to right. We will call this sequence the **yield** of r. Finally, let $p(i)$ for $1 \leq i \leq d(S_0)$ be the position in the yield of r of the rightmost leaf of S_0's ith child. Thus, for all i, $1 \leq p(i) \leq n$.

Given this notation, the LCFRS rule corresponding to the MBOT rule $l \to r$ is constructed as $S_0 \to g(S_1, S_2, \ldots, S_k)$. The LCFRS nonterminal S_i has fan-out equal to the corresponding MBOT state's rank plus one: $\varphi(S_i) = d(S_i) + 1$. This is because the LCFRS nonterminal has one span in the source language, and $d(S_i)$ spans in the target language of the translation. The combination function for the LCFRS rule $S_0 \to g(S_1, S_2, \ldots, S_k)$ is:

$$
\begin{aligned}
g(\langle e_1, f_{1,1}, \ldots, f_{1,d(S_1)}\rangle, \ldots, \langle e_k, f_{k,1}, \ldots, f_{k,d(S_k)}\rangle) = \\
\langle\, e_1 \cdots e_k, \quad f_{\pi(1),\mu(1)} \cdots f_{\pi(p(1)),\mu(p(1))}, \\
f_{\pi(p(1)+1),\mu(p(1)+1)} \cdots f_{\pi(p(2)),\mu(p(2))}, \\
\ldots, \\
f_{\pi(p(d(S_0)-1)+1),\mu(p(d(S_0)-1)+1)} \cdots f_{\pi(p(d(S_0))),\mu(p(d(S_0)))} \,\rangle
\end{aligned}
$$

Here we use e_i for the variables in the LCFRS rule corresponding to spans in the input tree of the MBOT, and $f_{i,j}$ for variables corresponding to the output tree. The pattern in which these spans fit together is specified by the functions π and μ that were read off of the MBOT rule.

Examples of the conversion of an MBOT rule to an LCFRS rule are shown in Figure 2. The first example shows an MBOT rule derived from an STSG rule, in this case converting SVO (as in English) to VSO (as in Arabic) word order. The states of an MBOT rule derived from an STSG rule always have rank 1. In the resulting LCFRS

rule, this means that every nonterminal in the grammar has fan-out 2, corresponding to one span in the source language string and one span in the target language string of the translation. This is what we would expect, given that, in terms of the translations produced, STSG is equivalent to SCFG (because the internal tree structure of the rules is irrelevant), and SCFG falls within the class of LCFRS grammars of fan-out 2. Figure 2b shows a more general example, where the states of the MBOT rule have rank > 1.

Now we extend this construction to handle tree symbols of rank zero, which correspond to terminal symbols in the LCFRS. Let α_0 be the sequence of rank zero symbols appearing at the leaves of l to the left of $x_{1,1}$, and let α_i for $1 \leq i \leq k$ be the sequence of rank zero symbols to the right of $x_{i,d(S_i)}$, and to the left of $x_{i+1,1}$ if $i < k$. Let $\beta_{i,j}$ be the sequence of symbols of rank zero at the leaves of r appearing in the subtree under the ith child of S_0 after the jth variable in this subtree and before the $j+1$th variable, with $\beta_{i,0}$ being to the left of the first variable, and $\beta_{i,p(i)}$ being to the right of the last variable. We can add these sequences of terminal symbols to the LCFRS rule as follows:

$$g(\langle e_1, f_{1,1}, \ldots, f_{1,d(S_1)}\rangle, \ldots, \langle e_k, f_{k,1}, \ldots, f_{k,d(S_k)}\rangle) =$$
$$\langle\ \alpha_0 e_1 \alpha_1 \cdots e_k \alpha_k,$$
$$\beta_{1,0} f_{\pi(1),\mu(1)} \beta_{1,1} \cdots f_{\pi(p(1)),\mu(p(1))} \beta_{1,p(1)},$$
$$\beta_{2,0} f_{\pi(p(1)+1),\mu(p(1)+1)} \beta_{2,1} \cdots f_{\pi(p(2)),\mu(p(2))} \beta_{2,p(2)},$$
$$\ldots,$$
$$\beta_{d(S_0),0} f_{\pi(p(d(S_0)-1)+1),\mu(p(d(S_0)-1)+1)} \beta_{d(S_0),1} \cdots f_{\pi(p(d(S_0))),\mu(p(d(S_0))} \beta_{d(S_0),p(d(S_0))}\ \rangle$$

An example of this conversion is shown in Figure 3. In this example, $\alpha_1 = of$, $\beta_{1,1} =$ 的, all other α and β values are the empty string, and $d(S_0) = 1$. We refer to the LCFRS rule constructed from MBOT rule $l \rightarrow r$ as $p_{l \rightarrow r}$.

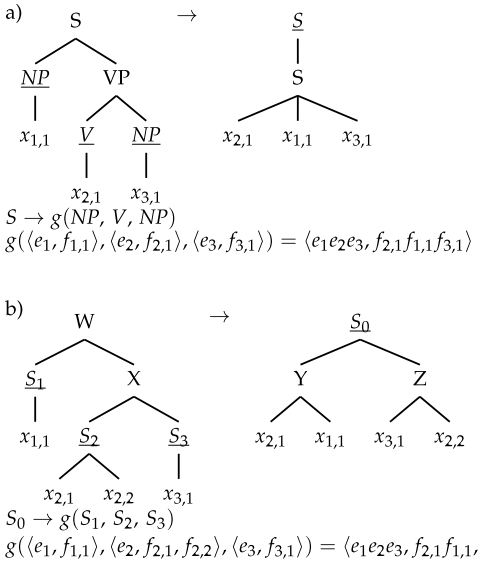


Figure 2
Examples of the conversion of an MBOT rule to an LCFRS rule.

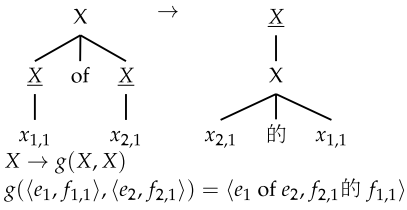


Figure 3
Conversion of an MBOT rule with symbols of rank zero to an LCFRS production with terminals.

Finally, we add a **start rule** rule $S \rightarrow g(S_i)$, $g(\langle e,f\rangle) = \langle e\#f\rangle$ for each $S_i \in F$ to generate all final states S_i of the MBOT from the start symbol S of the LCFRS.

We now show that the language of the LCFRS constructed from a given MBOT is identical to the yield of the MBOT. We represent MBOT transductions as derivation trees, where each node is labeled with an MBOT rule, and each node's children are the rules used to produce the subtrees matched by any variables in the rule. We can construct an LCFRS derivation tree by simply relabeling each node with the LCFRS rule constructed from the node's MBOT rule. Because, in the MBOT derivation tree, each node has children which produce the states required by the the MBOT rule's left-hand side (l.h.s.), it also holds that, in the LCFRS derivation tree, each node has as its children rules which expand the set of nonterminals appearing in the parent's r.h.s. Therefore the LCFRS tree constitutes a valid derivation.

Given the mapping from MBOT derivations to LCFRS derivations, the following lemma relates the strings produced by the derivations:

Lemma 1
Let T_{MBOT} be an MBOT derivation tree with I as its input tree and O as its output tree, and construct T_{LCFRS} by mapping each node n_{MBOT} in T_{MBOT} to a node n_{LCFRS} labeled with the LCFRS production constructed from the rule at n_{MBOT}. Let $\langle t_0, t_1, \ldots, t_k\rangle$ be the string tuple returned by the LCFRS combination function at any node n_{LCFRS} in T_{LCFRS}. The string t_0 contains the yield of the node of I at which the MBOT rule at the node of T_{MBOT} corresponding to n_{LCFRS} was applied. Furthermore, the strings $t_1, \ldots, t_k$ contain the k yields of the k MBOT output subtrees (subtrees of O) that are found as children of the root (state symbol) of the MBOT rule's right-hand side.

Proof
When we apply the LCFRS combination functions to build the string produced by the LCFRS derivation, the sequence of function applications corresponds exactly to the bottom–up application of MBOT rules to the input tree. Let us refer to the tuple returned by one LCFRS combination function g as $\langle t_0, t_1, \ldots, t_k\rangle$. An MBOT rule applying at the bottom of the input tree cannot contain any variables, and for MBOT rules of this type, our construction produces an LCFRS rule with a combination function of the form:

$$g() = \langle \alpha_0, \beta_{1,0}, \ldots, \beta_{k,0}\rangle$$

taking no arguments and returning string constants equal to the yield of the MBOT rule's l.h.s, and the sequence of yields of the k subtrees under the r.h.s.'s root. Now we consider how further rules in the LCFRS derivation make use of the tuple $\langle t_0, t_1, \ldots, t_k\rangle$. Our LCFRS combination functions always concatenate the first elements of the input

tuple in order, adding any terminals present in the portion of the input tree matched by the MBOT's l.h.s. Thus the combination functions maintain the property that the first element in the resulting tuple, t_0, contains the yield of the subtree of the input tree where the corresponding MBOT rule applied. The combination functions combine the remaining elements in their input tuples in the same order given by the MBOT rule's r.h.s., again adding any terminals added to the output tree by the MBOT rule. Thus, at each step, the strings $t_1, \ldots, t_k$ returned by LCFRS combination functions contain the k yields of the k MBOT output subtrees found as children of the root (state symbol) of the MBOT rule's r.h.s. By induction, the lemma holds at each node in the derivation tree. ■

The correspondence between LCFRS string tuples and MBOT tree yields gives us our first result:

Theorem 1
yield(MBOT) $\subset$ trans(LCFRS).

Proof
From a given MBOT, construct an LCFRS as described previously. For any transduction of the MBOT, from Lemma 1, there exists an LCFRS derivation which produces a string consisting of the yield of the MBOT's input and output trees joined by the # symbol. In the other direction, we note that any valid derivation of the LCFRS corresponds to an MBOT transduction on some input tree; this input tree can be constructed by assembling the left-hand sides of the MBOT rules from which the LCFRS rules of the LCFRS derivation were originally constructed. Because there is a one-to-one correspondence between LCFRS and MBOT derivations, the translation produced by the LCFRS and the yield of the MBOT are identical.

Because we can construct an LCFRS generating the same translation as the yield of any given MBOT, we see that yield(MBOT) $\subset$ trans(LCFRS). ■

The translations produced by MBOTs are equivalent to the translations produced by a certain restricted class of LCFRS grammars, which we now specify precisely.

Theorem 2
The class of translations yield(MBOT) is equivalent to yield(1-m-LCFRS), where 1-m-LCFRS is defined to be the class of LCFRS grammars where each rule either is a start rule of the form $S \rightarrow g(S_i)$, $g(\langle e,f \rangle) = \langle e\#f \rangle$, or meets both of the following conditions:

- The combination function keeps the two sides of the translation separate. That is, it must be possible to write

$$g(\langle e_1, f_{1,1}, \ldots, f_{1,\varphi_1} \rangle, \ldots, \langle e_r, f_{r,1}, \ldots, f_{r,\varphi_r} \rangle)$$

  as

$$g_1(\langle e_1 \rangle, \ldots, \langle e_r \rangle) + g_2(\langle f_{1,1}, \ldots, f_{1,\varphi_1} \rangle, \ldots, \langle f_{r,1}, \ldots, f_{r,\varphi_r} \rangle)$$

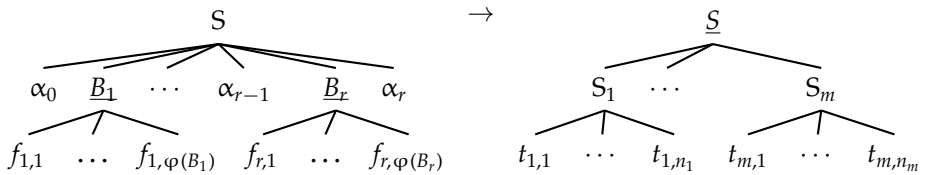
  where $+$ represents tuple concatenation, for some functions g_1 and g_2.
- The function g_1 returns a tuple of length 1.

Proof
Our construction for transforming an MBOT to an LCFRS produces LCFRS grammars satisfying the given constraints, so yield(MBOT) $\subset$ trans(1-m-LCFRS).

To show the other direction, we will construct an MBOT from a 1-m-LCFRS. For each 1-m-LCFRS rule of the form

$$S \rightarrow g(B_1, \ldots, B_r)$$
$$g(\langle e_1, f_{1,1}, \ldots, f_{1,\varphi_1}\rangle, \ldots, \langle e_r, f_{r,1}, \ldots, f_{r,\varphi_r}\rangle) =$$
$$g_1(\langle e_1\rangle, \ldots, \langle e_r\rangle) + g_2(\langle f_{1,1}, \ldots, f_{1,\varphi_1}\rangle, \ldots, \langle f_{r,1}, \ldots, f_{r,\varphi_r}\rangle)$$
$$g_1(\langle e_1\rangle, \ldots, \langle e_r\rangle) = \langle \alpha_0 e_1 \cdots \alpha_{r-1} e_r \alpha_r\rangle$$
$$g_2(\langle f_{1,1}, \ldots, f_{1,\varphi_1}\rangle, \ldots, \langle f_{r,1}, \ldots, f_{r,\varphi_r}\rangle) = \langle t_{1,1} \cdots t_{1,n_1}, \ldots, t_{m,1} \cdots t_{m,n_m}\rangle$$

where each α_i is a string of terminals, and each symbol $t_{i,j}$ is either a variable $f_{i',j'}$, or a single terminal, we construct the MBOT rule:



By the same reasoning used for our construction of LCFRS grammars from MBOTs, there is a one-to-one correspondence between derivation trees of the 1-m-LCFRS and the constructed MBOT, and the yield strings also correspond at each node in the derivation trees. Therefore, yield(1-m-LCFRS) $\subset$ yield(MBOT).

Because we have containment in both directions, yield(MBOT) = trans(1-m-LCFRS). ■

We now move on to consider the languages formed by the source and target projections of MBOT translations.

Grammars of the class 1-m-LCFRS have the property that, for any nonterminal A (other than the start symbol S) having fan-out $\varphi(A)$, one span is always realized in the source string (to the left of the # separator), and $\varphi(A) - 1$ spans are always realized in the target language (to the right of the separator). This property is introduced by the start rules $S \rightarrow g(S_i)$, $g(\langle e, f\rangle) = \langle e\#f\rangle$ and is maintained by all further productions because of the condition on 1-m-LCFRS that the combination function must keep the two sides of translation separate. For a 1-m-LCFRS rule constructed from an MBOT, we define the rule's source language projection to be the rule obtained by discarding all the target language spans, as well as the separator symbol # in the case of the start productions. The definition of 1-m-LCFRS guarantees that the combination function returning a rule's l.h.s. source span needs to have only the r.h.s. source spans available as arguments.

For an LCFRS G, we define $L(G)$ to be the language produced by G. We define source(G) to be the LCFRS obtained by projecting each rule in G. Because more than one rule may have the same projection, we label the rules of source(G) with their origin rule, preserving a one-to-one correspondence between rules in the two grammars. Similarly, we obtain a rule's target language projection by discarding the source language spans, and define target(G) to be the resulting grammar.

Lemma 2
For an LCFRS G constructed from an MBOT M by the given construction, $L(\text{source}(G)) = \text{source}(\text{trans}(M))$, and $L(\text{target}(G)) = \text{target}(\text{trans}(M))$.

Proof
There is a valid derivation tree in the source language projection for each valid derivation tree in the full LCFRS, because for any expansion rewriting a nonterminal of fan-out $\varphi(A)$ in the full grammar, we can apply the projected rule to the corresponding nonterminal of fan-out 1 in the projected derivation. In the other direction, for any expansion in a derivation of the source projection, a nonterminal of fan-out $\varphi(A)$ will be available for expansion in the corresponding derivation of the full LCFRS. Because there is a one-to-one correspondence between derivations in the full LCFRS and its source projection, the language generated by the source projection is the source of the translation generated by the original LCFRS. By the same reasoning, there is a one-to-one correspondence between derivations in the target projection and the full LCFRS, and the language produced by the target projection is the target side of the translation of the full LCFRS. ■

Lemma 2 implies that it is safe to evaluate the power of the source and target projections of the LCFRS independently. This fact leads to our next result.

Theorem 3
$\text{yield}(\text{MBOT}) \subsetneq \text{trans}(\text{LCFRS})$.

Proof
In the LCFRS generated by our construction, all nonterminals have fan-out 1 in the source side of the translation. Therefore, the source side of the translation is a context-free language, and an MBOT cannot represent the following translation:

$$\{(\mathrm{a}^n\mathrm{b}^n\mathrm{c}^n\mathrm{d}^n,\ \mathrm{a}^n\mathrm{b}^n\mathrm{c}^n\mathrm{d}^n) \mid n \geq 1\}$$

which is produced by an STAG (Shieber and Schabes 1990; Schabes and Shieber 1994). Because STAG is a type of LCFRS, $\text{yield}(\text{MBOT}) \subsetneq \text{trans}(\text{LCFRS})$. ■

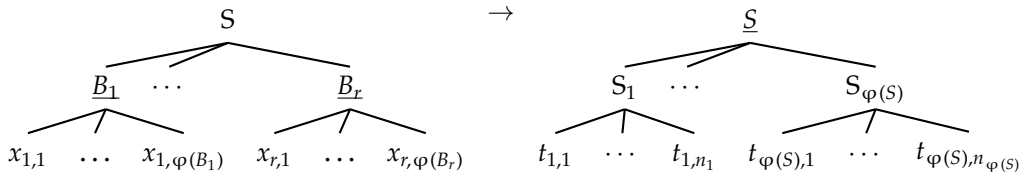
Although the source side of the translation produced by an MBOT must be a context-free language, we now show that the target side can be any language produced by an LCFRS.

Theorem 4
$\text{target}(\text{yield}(\text{MBOT})) = \text{LCFRS}$

Proof
Given an input LCFRS, we can construct an MBOT whose target side corresponds to the rules in the original LCFRS, and whose source simply accepts derivation trees of the LCFRS. To make this precise, given an LCFRS rule in the general form:

$$S \rightarrow g(B_1, \ldots, B_r)$$
$$g(\langle x_{1,1}, \ldots, x_{1,\varphi(B_1)}\rangle, \ldots, \langle x_{1,1}, \ldots, x_{1,\varphi(B_r)}\rangle) = \langle t_{1,1} \cdots t_{1,n_1}, \ldots, t_{\varphi(S),1} \cdots t_{\varphi(S),n_{\varphi(S)}}\rangle$$

where each symbol $t_{i,j}$ is either some variable $x_{i',j'}$ or a terminal from the alphabet of the LCFRS, we construct the MBOT rule:



where the MBOT's input alphabet contains a symbol S for each LCFRS nonterminal S, and the MBOT's output alphabet contains $\varphi(S)$ symbols S_i for each LCFRS nonterminal S. This construction for converting an LCFRS to an MBOT shows that LCFRS $\subset$ target(yield(MBOT)).

Given our earlier construction for generating the target projection of the LCFRS derived from an MBOT, we know that target(yield(MBOT)) $\subset$ LCFRS. Combining these two facts yields the theorem. ■

4. Composition of STSGs

Maletti et al. (2009) discuss the composition of extended top–down tree transducers, which are equivalent to STSGs, as shown by Maletti (2010). They show that this formalism is not closed under composition in terms of the tree transformations that are possible. In this article, we focus on the string yields of the formalisms under discussion, and from this point of view we now examine the question of whether the *yield* of the composition of two STSGs is itself the yield of an STSG in general. It is important to note that, although we focus on the yield of the composition, in our notion of STSG composition, the tree structure output by the first STSG still serves as input to the second STSG.

Maletti et al. (2009) give two tree transformations as counterexamples to the compositionality of STSG, shown in Figure 4. From the point of view of string yield, both of the transformations are equivalent to an STSG rule that simply copies the three variables with no re-ordering. Thus, these counterexamples are not sufficient to show that the yield of the composition of two STSGs is not the yield of an STSG.

We now present two STSGs, shown in MBOT notation in Figures 5 and 6, whose composition is not a translation produced by an STSG. The essence of this counterexample, explained in more detail subsequently, is that rules from the two STSGs apply in an overlapping manner to unboundedly long sequences, as in the example of Arnold

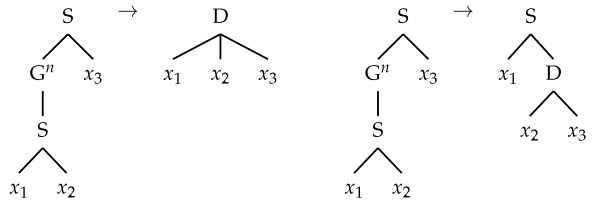


Figure 4
Examples of tree transformations not contained in STSG, from Maletti et al. (2009). Here G^n denotes a unary chain of G's of arbitrary length.

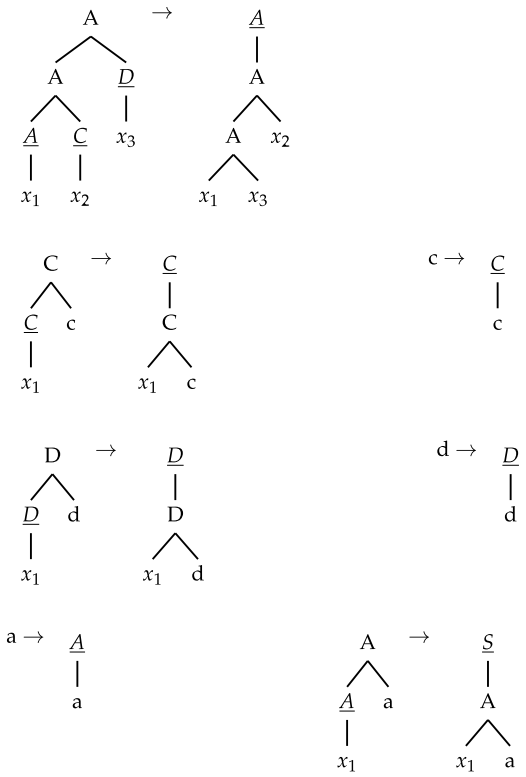


Figure 5
First MBOT in composition.

and Dauchet (1982, section 3.4). To this approach we add a re-ordering pattern which results in a translation that we will show not to be possible with STSG.

The heart of each MBOT is the first rule, which reverses the order of adjacent sequences of c's and d's. The MBOT of Figure 5 generates the translation:

$$\{(\mathrm{a}[\mathrm{c}^{n_{2i-1}}\mathrm{d}^{n_{2i}}]_{i=1}^{\ell}\mathrm{a},\ \mathrm{a}[\mathrm{d}^{n_{2i}}\mathrm{c}^{n_{2i-1}}]_{i=1}^{\ell}\mathrm{a}) \mid n_i \geq 1, \ell \geq 1\}$$

where ℓ is the number of times the first rule of the transducer is applied, and the notation $[x_i]_{i=1}^{n}$ indicates the string concatenation $x_1 x_2 \cdots x_n$. Here we have 2ℓ repeated sequences of characters c and d, each occurring n_i times, with each integer n_i for $1 \leq i \leq 2\ell$ varying freely.

The second MBOT reverses sequences of c's and d's in a pattern that is offset by one from the pattern of the first MBOT. It produces the translation:

$$\{(\mathrm{ad}^{n_1}[\mathrm{c}^{n_{2i}}\mathrm{d}^{n_{2i+1}}]_{i=1}^{\ell-1}\mathrm{c}^{n_{2\ell}}\mathrm{a},\ \mathrm{ad}^{n_1}[\mathrm{d}^{n_{2i+1}}\mathrm{c}^{n_{2i}}]_{i=1}^{\ell-1}\mathrm{c}^{n_{2\ell}}\mathrm{a}) \mid n_i \geq 1, \ell \geq 1\}$$

When we compose the two MBOTs, the yield of the resulting transducer is the translation:

$$T_{crisscross} = \bigcup_{\ell=1}^{\infty} T^{\ell}_{crisscross} \tag{1}$$

where

$$T^{\ell}_{crisscross} = \{(\mathrm{a}[\mathrm{c}^{n_{2i-1}}\mathrm{d}^{n_{2i}}]_{i=1}^{\ell}\mathrm{a},\ \mathrm{ad}^{n_2}[\mathrm{d}^{n_{2i+2}}\mathrm{c}^{n_{2i-1}}]_{i=1}^{\ell-1}\mathrm{c}^{n_{2\ell-1}}\mathrm{a}) \mid n_i \geq 1\} \tag{2}$$

A visualization of the alignment pattern of this translation is shown in Figure 7. We will show that $T_{crisscross}$ cannot be produced by any SCFG.

We define an SCFG to be a system $(V, \Sigma, \Delta, P, S)$ where V is a set of nonterminals, Σ and Δ are the terminal alphabets of the source and target language respectively, $S \in V$ is a distinguished start symbol, and P is a set of productions of the following general form:

$$X_0 \rightarrow X_1^{\boxed{1}} \cdots X_n^{\boxed{n}},\ X_{\pi(1)}^{\boxed{\pi(1)}} \cdots X_{\pi(n)}^{\boxed{\pi(n)}}$$

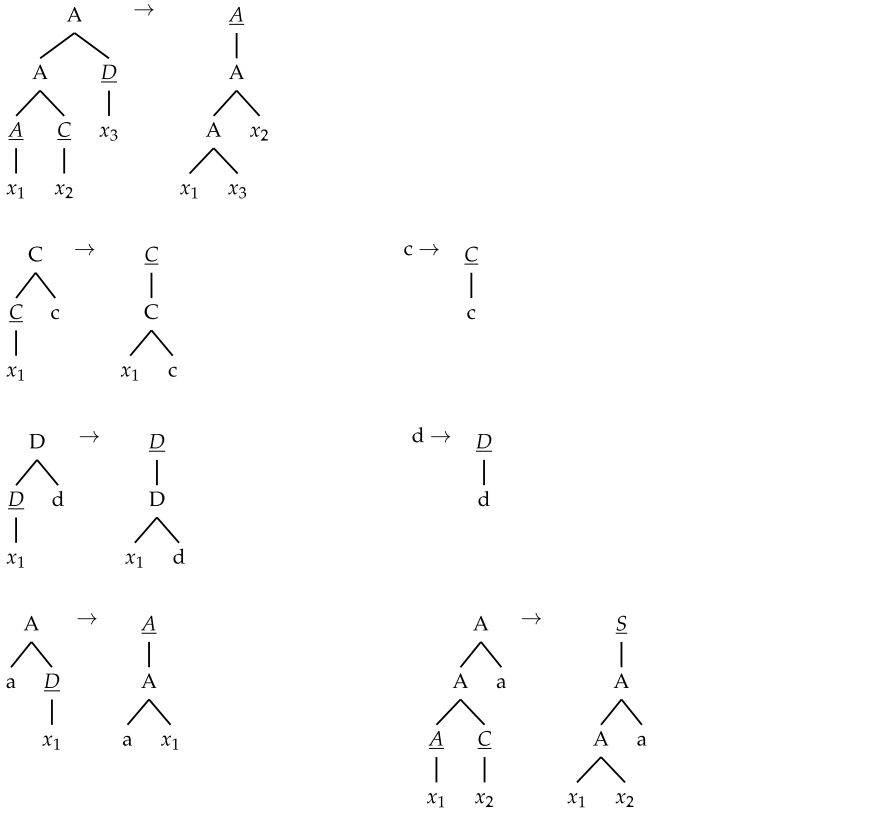


Figure 6
Second MBOT in composition.

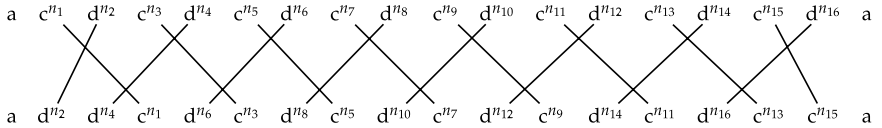


Figure 7
Translation resulting from MBOT composition with $\ell = 8$.

where π is a permutation of length n, and the variables X_i for $0 \leq i \leq n$ range over nonterminal symbols (for example, X_1 and X_2 may both stand for nonterminal A). In SCFG productions, the l.h.s. nonterminal rewrites into a string of terminals and nonterminals in both the source and target languages, and pairs of r.h.s. nonterminals that are linked by the same superscript index must be further rewritten by the same rule.

In terms of string translations, STSGs and SCFGs are equivalent, because any SCFG is also an STSG with rules of depth 1, and any STSG can be converted to an SCFG with the same string translation by simply removing the internal tree nodes in each rule. We will adopt SCFG terminology for our proof because the internal structure of STSG rules is not relevant to our result.

For a fixed value of ℓ, the translation $T^{\ell}_{crisscross}$ can be produced by an SCFG of rank 2ℓ, shown in Figure 8, because one rule can produce 2ℓ nonterminals arranged in the permutation of Figure 7. (In the context of SCFGs, **rank** refers to the maximum number of nonterminals on the r.h.s. of a rule.) We will show that strings of this form cannot be produced by any SCFG of rank less than 2ℓ. Intuitively, factoring the alignment pattern of Figure 7 into smaller SCFG rules would require identifying subsequences in the two languages that are consistently aligned to one another, and, as can be seen from the figure, no such subsequences exist. Because ℓ can be unboundedly large in our translation, the translation cannot be produced by any SCFG of fixed rank.

We will assume, without loss of generality, that any SCFG is written in a normal form such that each rule's r.h.s. either contains only terminals in each language, or contains only nonterminals. An SCFG can be transformed into this normal form by applying the following procedure to each rule:

1. Associate each sequence of terminals with the preceding nonterminal, or the following nonterminal in the case of initial terminals.
2. Replace each group consisting of a nonterminal and its associated terminals with a fresh nonterminal A, and add a rule rewriting A as the group in source and target. (Nonterminals with no associated terminals may be left intact.)
3. In each rule created in the previous step, replace each sequence of terminals with another fresh nonterminal B, and add a rule rewriting B as the terminal sequence in source and target.

Figure 9 shows an example of this grammar transformation. Because we do not change the rank of existing rules, and we add rules of rank no greater than 3, the transformation does not increase the rank of any grammar having rank at least 3.

$S \rightarrow \mathrm{a} \left[C^{\boxed{2i-1}} D^{\boxed{2i}}\right]_{i=1}^{\ell} \mathrm{a},\ \mathrm{a} D^{\boxed{2}} \left[D^{\boxed{2i+2}} C^{\boxed{2i-1}}\right]_{i=1}^{\ell-1} C^{\boxed{2\ell-1}} \mathrm{a}$
$C \rightarrow C^{\boxed{1}} \mathrm{c},\ C^{\boxed{1}} \mathrm{c}$
$C \rightarrow \mathrm{c},\ \mathrm{c}$
$D \rightarrow D^{\boxed{1}} \mathrm{d},\ D^{\boxed{1}} \mathrm{d}$
$D \rightarrow \mathrm{d},\ \mathrm{d}$

Figure 8
An SCFG producing translation $T^{\ell}_{crisscross}$ for fixed ℓ.

a) $X \rightarrow X^{[1]}$ 的 $X^{[2]}$, $X^{[2]}$ of $X^{[1]}$
$X \rightarrow$ 生命, life
$X \rightarrow$ 意义, meaning

b) $X \rightarrow Y^{[1]}X^{[2]}$, $X^{[2]}Y^{[1]}$
$Y \rightarrow X^{[1]}Z^{[2]}$, $Z^{[2]}X^{[1]}$
$Z \rightarrow$ 的, of
$X \rightarrow$ 生命, life
$X \rightarrow$ 意义, meaning

Figure 9
Conversion of grammar (a) to normal form (b) in which each rule has only nonterminals or only terminals on the r.h.s.

$X^{[1]}$, $X^{[1]}$
$Y^{[1]}X^{[2]}$, $X^{[2]}Y^{[1]}$
$X^{[1]}Z^{[2]}X^{[3]}$, $X^{[3]}Z^{[2]}X^{[1]}$
$X^{[1]}$ of $X^{[2]}$, $X^{[2]}$ 的 $X^{[1]}$
$X^{[1]}$ of life , 生命 的 $X^{[1]}$
meaning of life , 生命 的 意义

Figure 10
An SCFG derivation produced by applying each rule in Figure 9b once, in the order given in Figure 9b. Indices of linked nonterminals are renumbered after each step to be monotonically increasing in the English side of the derivation. The preterminal permutation of the derivation, (3,2,1), is the sequence of indices on the Chinese side in the last step before any terminals are produced.

In an SCFG derivation, nonterminals in either language are linked as shown in Figure 10. We restrict derivations to apply rules producing terminals after applying all other rules. We refer to nonterminals at the last step in which the sentential form consists exclusively of nonterminals as **preterminals**, and we refer to a pair of linked preterminals as an **aligned preterminal pair**. Assuming that aligned preterminal pairs are indexed consecutively in the source side of the sentential form, we refer to the sequence of indices in the target side as the **preterminal permutation** of a derivation. For example, the preterminal permutation of the derivation in Figure 10 is (3,2,1). The permutation of any sentential form of an SCFG of rank r can be produced by composing permutations of length no greater than r, by induction over the length of the derivation. Thus, while the permutation (3,2,1) of our example can be produced by composing permutations of length 2, the preterminal permutation (2,4,1,3) can never be produced by an SCFG of rank 2 (Wu 1997). In fact, this restriction also applies to subsequences of the preterminal permutation.

Lemma 3
Let π be a preterminal permutation produced by an SCFG derivation containing rules of maximum rank r, and let π' be a permutation obtained from π by removing some elements and renumbering the remaining elements with a strictly increasing function. Then π' falls within the class of compositions of permutations of length r.

Proof
From each rule in the derivation producing preterminal permutation π, construct a new rule by removing any nonterminals whose indices were removed from π. The resulting sequence of rules produces preterminal permutation π' and contains rules of rank no greater than r. ■
As an example of Lemma 3, removing any element from the permutation (3,2,1) results in the permutation (2,1), which can still (trivially) be produced by an SCFG of rank 2.

We will make use of another general fact about SCFGs, which we derive by applying Ogden's Lemma (Ogden 1968), a generalized pumping lemma for context-free languages, to the source language of an SCFG.

Lemma 4 (Ogden's Lemma)
For each context-free grammar $G = (V, \Sigma, P, S)$ there is an integer k such that for any word ξ in $L(G)$, if any k or more distinct positions in ξ are designated as distinguished, then there is some A in V and there are words α, β, γ, δ, and μ in Σ^* such that:

- $S \Rightarrow^* \alpha A \mu \Rightarrow^* \alpha\beta A \delta\mu \Rightarrow^* \alpha\beta\gamma\delta\mu = \xi$, and hence $\alpha\beta^m\gamma\delta^m\mu \in L(G)$ for all $m \geq 0$.
- γ contains at least one of the distinguished positions.
- Either α and β both contain distinguished positions, or δ and μ both contain distinguished positions.
- $\beta\gamma\delta$ contains at most k distinguished positions.

Ogden's lemma can be extended as follows to apply to SCFGs.

Lemma 5
For each SCFG $G = (V, \Sigma, \Delta, P, S)$ having source alphabet Σ and target alphabet Δ, there is an integer k such that for any string pair (ξ, ξ') in $L(G)$, if any k or more distinct positions in ξ are designated as distinguished, then there is some A in V and there are words α, β, γ, δ, and μ in Σ^* and α', β', γ', δ', and μ' in Δ^* such that:

- $(S^{\boxed{1}}, S^{\boxed{1}}) \Rightarrow^* (\alpha A^{\boxed{1}} \mu,\ \alpha' A^{\boxed{1}} \mu') \Rightarrow^* (\alpha\beta A^{\boxed{1}} \delta\mu,\ \alpha'\beta' A^{\boxed{1}} \delta'\mu') \Rightarrow^* (\alpha\beta\gamma\delta\mu,\ \alpha'\beta'\gamma'\delta'\mu') = (\xi, \xi')$, and hence $(\alpha\beta^m\gamma\delta^m\mu,\ \alpha'(\beta')^m\gamma'(\delta')^m\mu') \in L(G)$ for all $m \geq 0$.
- γ contains at least one of the distinguished positions.
- Either α and β both contain distinguished positions, or δ and μ both contain distinguished positions.
- $\beta\gamma\delta$ contains at most k distinguished positions.

Note that there are no guarantees on the form of α', β', γ', δ', and μ', and indeed these may all be the empty string.

Proof
There must exist some sequence of rules in the source projection of G which licenses the derivation $A \Rightarrow^* \beta A \delta$. If we write the jth rule in this sequence as $A_j \rightarrow \nu_j$, there must exist a synchronous rule in G of the form $A_j \rightarrow \nu_j, \nu'_j$ that rewrites the same nonterminal. Thus G licenses a synchronous derivation $(A^{\boxed{1}}, A^{\boxed{1}}) \Rightarrow^* (\beta A^{\boxed{1}} \delta,\ \beta' A^{\boxed{1}} \delta')$ for some β' and δ'. Similarly, the source derivation $S \Rightarrow^* \alpha A \mu$ has a synchronous counterpart $(S^{\boxed{1}}, S^{\boxed{1}}) \Rightarrow^* (\alpha A^{\boxed{1}} \mu,\ \alpha' A^{\boxed{1}} \mu')$ for some α' and μ', and the source derivation $A \Rightarrow \gamma$ has a synchronous counterpart $(A^{\boxed{1}}, A^{\boxed{1}}) \Rightarrow (\gamma, \gamma')$ for some γ'. Because the synchronous

derivation $(A^{\boxed{1}}, A^{\boxed{1}}) \Rightarrow^* (\beta A^{\boxed{1}} \delta, \beta' A^{\boxed{1}} \delta')$ can be repeated any number of times, the string pairs

$$(\alpha \beta^m \gamma \delta^m \mu,\ \alpha' (\beta')^m \gamma' (\delta')^m \mu') \tag{3}$$

are generated by the SCFG for all $m \geq 0$. The further conditions on α, β, γ, δ, and μ follow directly from Ogden's Lemma. ■

We refer to a substring arising from a term c^{n_i} or d^{n_i} in the definition of $T^{\ell}_{crisscross}$ (Equation (2)) as a **run**. In order to distinguish runs, we refer the run arising from c^{n_i} or d^{n_i} as the ith run. We refer to the pair (c^{n_i}, c^{n_i}) or (d^{n_i}, d^{n_i}) consisting of the ith run in the source and target strings as the ith **aligned run**. We now use Lemma 5 to show that aligned runs must be generated from aligned preterminal pairs.

Lemma 6
Assume that some SCFG G' generates the translation $T^{\ell}_{crisscross}$ for some fixed ℓ. There exists a constant k such that, in any derivation of grammar G' having each $n_i > k$, for any $i, 1 \leq i \leq 2\ell$, there exists at least one aligned preterminal pair among the subsequences of source and target preterminals generating the ith aligned run.

Proof
We consider a source string ξ, $(\xi, \xi') \in T^{\ell}_{crisscross}$, such that the length n_i of each run is greater than the constant k of Lemma 5. For a fixed i, $1 \leq i \leq 2\ell$, we consider the distinguished positions to be all and only the terminals in the ith run. This implies that the run can be pumped to be arbitrarily long; indeed, this follows from the definition of the language itself.

Because our distinguished positions are within the ith run, and because Lemma 5 guarantees that either α, β, and γ all contain distinguished positions or γ, δ, and μ all contain distinguished positions, we are guaranteed that either β or δ lies entirely within the ith run. Consider the case where β lies within the run. We must consider three possibilities for the location of δ in the string:

Case 1. The string δ also lies entirely within the ith run.

Case 2. The string δ contains substrings of more than one run. This cannot occur, because pumped strings of the form $\alpha \beta^m \gamma \delta^m \mu$ would contain more than 2ℓ runs, which is not allowed under the definition of $T^{\ell}_{crisscross}$.

Case 3. The string δ lies entirely within the jth run, where $j \neq i$. The strings $\alpha \beta^m \gamma \delta^m \mu$ have the same form as $\alpha \beta \gamma \delta \mu$, with the exception that the ith and jth runs are extended from lengths n_i and n_j to some greater lengths n_i' and n_j'. By the definition of $T^{\ell}_{crisscross}$, for each source string, only one target string is permitted. For string pairs of the form of Equation (3) to belong to $T^{\ell}_{crisscross}$, β' and δ' must lie within the ith and jth aligned runs in the target side. Because the permutation of Figure 7 cannot be decomposed, there must exist some k such that the kth aligned run lies between the ith and jth aligned runs in one side of the translation, and outside the ith and jth aligned runs in the other side of the translation. If this were not the case, we would be able to decompose the permutation by factoring out the subsequence between the ith and jth runs on both sides of the translation. Consider the case where the kth aligned run lies between the ith and jth aligned runs in the source side, and therefore is a substring of γ in the source,

and a substring of either α' or μ' in the target. We apply Lemma 5 a second time, with all terminals of the kth run as the distinguished positions, to the derivation $(A, A) \Rightarrow^* (\gamma, \gamma')$ by taking A as the start symbol of the grammar. This implies that there exist $\hat{\beta}$, $\hat{\gamma}$, $\hat{\delta}$, $\hat{\beta}'$, $\hat{\gamma}'$, and $\hat{\delta}'$ such that

$$(\xi, \xi') = (\alpha\beta\hat{\beta}\hat{\gamma}\hat{\delta}\delta\mu,\ \alpha'\beta'\hat{\beta}'\hat{\gamma}'\hat{\delta}'\delta'\mu')$$

and all strings

$$(\alpha\beta^m\hat{\beta}^n\hat{\gamma}\hat{\delta}^n\delta^m\mu,\ \alpha'(\beta')^m(\hat{\beta}')^n\hat{\gamma}'(\hat{\delta}')^n(\delta')^m\mu')$$

are members of the translation $T^{\ell}_{crisscross}$. Either $\hat{\beta}$ or $\hat{\delta}$ is a substring of source side of the kth aligned run, so the kth aligned run can be pumped to be arbitrarily long in the source without changing its length in the target. This contradicts the definition of $T^{\ell}_{crisscross}$. Similarly, the case where the kth aligned run lies between β' and δ' in the target leads to a contradiction. Thus the assumption that $j \neq i$ must be false.

Because Cases 2 and 3 are impossible, δ must lie entirely within the ith run. Similarly, in the case where δ contains distinguished positions, β must lie within the ith run. Thus both β and δ always lie entirely within the ith aligned run.

Because the β and δ lie within the ith aligned run, the strings $\alpha\beta^m\gamma\delta^m\mu$ have the same form as $\alpha\beta\gamma\delta\mu$, with the exception that the ith run is extended from length n_i to some greater length n_i'. For the pairs of Equation (3) to be members of the translation, β' and δ' must be substrings of the ith aligned run in the target. Because $\beta^m\gamma\delta^m$ and $(\beta')^m\gamma'(\delta')^m$ were derived from the same nonterminal, the two sequences of preterminals generating these two strings consist of aligned preterminal pairs. Because both $\beta^m\gamma\delta^m$ and $(\beta')^m\gamma'(\delta')^m$ are substrings of the ith aligned run, we have at least one aligned preterminal pair among the source and target preterminal sequences generating the ith aligned run. ■

Lemma 7
Assume that some SCFG G' generates the translation $T^{\ell}_{crisscross}$ for some fixed ℓ. There exists a constant k such that, if (ξ, ξ') is a string pair generated by G' having each $n_i > k$, any derivation of (ξ, ξ') with grammar G' must contain a rule of rank at least 2ℓ.

Proof
Because the choice of i in Lemma 6 was arbitrary, each aligned run must contain at least one aligned preterminal pair. If we select one such preterminal pair from each run, the associated permutation is that of Figure 7. This permutation cannot be decomposed, so, by Lemma 3, it cannot be generated by an SCFG derivation containing only rules of rank less than 2ℓ. ■

We will use one more general fact about SCFGs to prove our main result.

Lemma 8
Let G be an SCFG and let $T = L(G)$ be the translation it produces. Let F be a finite state machine, and let $R = L(F)$ be the regular language it accepts. Let T' be the translation derived by intersecting the source strings of T with R

$$T' = \{(s, t) \mid (s, t) \in T, s \in R\}$$

Then there exists an SCFG G' such that $T' = L(G')$.

Proof
Let V be the nonterminal set of G, and let S be the state set of F. Construct the SCFG G' with nonterminal set $V \times S \times S$ by applying the construction of Bar-Hillel, Perles, and Shamir (1961) for intersection of a CFG and finite state machine to the source side of each rule in G. ■

Now we are ready for our main result.

Theorem 5
SCFG = yield(STSG) $\subsetneq$ yield(STSG;STSG), where the semicolon denotes composition.

Proof
Assume that some SCFG G generates $T_{crisscross}$. Note that $T^{\ell}_{crisscross}$ is the result of intersecting the source of $T_{crisscross}$ with the regular language $\mathrm{a[c^{+}d^{+}]^{\ell}a}$. By Lemma 8, we can construct an SCFG G^{ℓ} generating $T^{\ell}_{crisscross}$. By Lemma 7, for each ℓ, G^{ℓ} has rank at least 2ℓ. The intersection construction does not increase the rank of the grammar, so G has rank at least 2ℓ. Because ℓ is unbounded in the definition of $T_{crisscross}$, and because any SCFG has a finite maximum rank, $T_{crisscross}$ cannot be produced by any SCFG. ■

4.1 Implications for Machine Translation

The ability of MBOTs to represent the composition of STSGs is given as a motivation for the MBOT formalism by Maletti (2010), but this raises the issue of whether synchronous parsing and machine translation decoding can be undertaken efficiently for MBOTs resulting from the composition of STSGs.

In discussing the complexity of synchronous parsing problems, we distinguish the case where the grammar is considered part of the input, and the case where the grammar is fixed, and only the source and target strings are considered part of the input. For SCFGs, synchronous parsing is NP-complete when the grammar is considered part of the input and can have arbitrary rank. For any fixed grammar, however, synchronous parsing is possible in time polynomial in the lengths of the source and target strings, with the degree of the polynomial depending on the rank of the fixed SCFG (Satta and Peserico 2005). Because MBOTs subsume SCFGs, the problem of recognizing whether a string pair belongs to the translation produced by an arbitrary MBOT, when the MBOT is considered part of the input, is also NP-complete.

Given our construction for converting an MBOT to an LCFRS, we can use standard LCFRS tabular parsing techniques to determine whether a given string pair belongs to the translation defined by the yield of a fixed MBOT. As with arbitrary-rank SCFG, LCFRS parsing is polynomial in the length of the input string pair, but the degree of the polynomial depends on the complexity of the MBOT. To be precise, the degree of the polynomial for LCFRS parsing is $\sum_{i=0}^{r} \varphi(S_i)$ (Seki et al. 1991), which yields $\sum_{i=0}^{r}(1 + d(S_i))$ when applied to MBOTs.

If we restrict ourselves to MBOTs that are derived from the composition of STSGs, synchronous parsing is NP-complete if the STSGs to compose are part of the input, because a single STSG suffices. For a composition of fixed STSGs, we obtain a fixed MBOT, and polynomial time parsing is possible. Theorem 5 indicates that we cannot apply SCFG parsing techniques off the shelf, but rather that we must implement some type of more general parsing system. Either of the STSGs used in our proof of Theorem 5

can be binarized and synchronously parsed in time $O(n^6)$, but tabular parsing for the LCFRS resulting from composition has higher complexity. Thus, composing STSGs generally increases the complexity of synchronous parsing.

The problem of language-model–integrated decoding with synchronous grammars is closely related to that of synchronous parsing; both problems can be seen as intersecting the grammar with a fixed source-language string and a finite-state machine constraining the target-language string. The widely used decoding algorithms for SCFG (Yamada and Knight 2002; Zollmann and Venugopal 2006; Huang et al. 2009) search for the highest-scoring translation when combining scores from a weighted SCFG and a weighted finite-state language model. As with SCFG, language-model–integrated decoding for weighted MBOTs can be performed by adding n-gram language model state to each candidate target language span. This, as with synchronous parsing, gives an algorithm which is polynomial in the length of the input sentence for a fixed MBOT, but with an exponent that depends on the complexity of the MBOT. Furthermore, Theorem 5 indicates that SCFG-based decoding techniques cannot be applied off the shelf to compositions of STSGs, and that composition of STSGs in general increases decoding complexity.

Finally, we note that finding the highest-scoring translation without incorporating a language model is equivalent to parsing with the source or target projection of the MBOT used to model translation. For the source language of the MBOT, this implies time $O(n^3)$ because the problem reduces to CFG parsing. For the target language of the MBOT, this implies polynomial-time parsing, where the degree of the polynomial depends on the MBOT, as a result of Theorem 4.

5. Conclusion

MBOTs are desirable for natural language processing applications because they are closed under composition and can be used to represent sequences of transformations of the type performed by STSGs. However, the string translations produced by MBOTs representing compositions of STSGs are strictly more powerful than the string translations produced by STSGs, which are equivalent to the translations produced by SCFGs. From the point of view of machine translation, because parsing with general LCFRS is NP-complete, restrictions on the power of MBOTs will be necessary in order to achieve polynomial–time algorithms for synchronous parsing and language-model–integrated decoding. Our result on the string translations produced by compositions of STSGs implies that algorithms for SCFG-based synchronous parsing or language-model-integrated decoding cannot be applied directly to these problems, and that composing STSGs generally increases the complexity of these problems. Developing parsing algorithms specific to compositions of STSGs, as well as possible restrictions on the STSGs to be composed, presents an interesting area for future work.

Acknowledgments

We are grateful for extensive feedback on earlier versions of this work from Giorgio Satta, Andreas Maletti, Adam Purtee, and three anonymous reviewers. This work was partially funded by NSF grant IIS-0910611.

References

Aho, Albert V. and Jeffery D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

Arnold, André and Max Dauchet. 1982. Morphismes et bimorphismes

d'arbres. *Theoretical Computer Science*, 20:33–93.

Bar-Hillel, Y., M. Perles, and E. Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172. Reprinted in Y. Bar-Hillel. 1964. *Language and Information: Selected Essays on their Theory and Application*, Addison-Wesley, Boston, MA, pages 116–150.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Engelfriet, J., E. Lilin, and A. Maletti. 2009. Extended multi bottom–up tree transducers. *Acta Informatica*, 46(8):561–590.

Galley, Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the International Conference on Computational Linguistics/Association for Computational Linguistics (COLING/ACL-06)*, pages 961–968, Sydney.

Huang, Liang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2009. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4):559–595.

Joshi, A. K., L. S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10:136–163.

Joshi, A. K. and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer, Berlin, pages 69–124.

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

Maletti, Andreas. 2010. Why synchronous tree substitution grammars? In *Proceedings of the 2010 Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-10)*, pages 876–884, Los Angeles, CA.

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

Melamed, I. Dan. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the 2003 Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-03)*, pages 158–165, Edmonton.

Melamed, I. Dan, Giorgio Satta, and Ben Wellington. 2004. Generalized multitext grammars. In *Proceedings of the 42nd Annual Conference of the Association for Computational Linguistics (ACL-04)*, pages 661–668, Barcelona.

Ogden, William F. 1968. A helpful result for proving inherent ambiguity. *Mathematical Systems Theory*, 2(3):191–194.

Rambow, Owen and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223(1-2):87–120.

Satta, Giorgio. 1992. Recognition of Linear Context-Free Rewriting Systems. In *Proceedings of the 30th Annual Conference of the Association for Computational Linguistics (ACL-92)*, pages 89–95, Newark, DE.

Satta, Giorgio and Enoch Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 803–810, Vancouver.

Schabes, Yves and Stuart M. Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20:91–124.

Seki, H., T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

Shieber, Stuart and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING-90)*, volume III, pages 253–258, Helsinki.

Vijay-Shankar, K., D. L. Weir, and A. K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Conference of the Association for Computational Linguistics (ACL-87)*, pages 104–111, Stanford, CA.

Wu, Dekai. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Yamada, Kenji and Kevin Knight. 2002. A decoder for syntax-based statistical MT. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02)*, pages 303–310, Philadelphia, PA.

Zollmann, Andreas and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 138–141, New York, NY.