Synchronous Context-Free Tree Grammars

Mark-Jan Nederhof
School of Computer Science
University of St Andrews
KY16 9SX, UK

Heiko Vogler
Department of Computer Science
Technische Universität Dresden
D-01062 Dresden, Germany

Abstract

We consider pairs of context-free tree grammars combined through synchronous rewriting. The resulting formalism is at least as powerful as synchronous tree adjoining grammars and linear, nondeleting macro tree transducers, while the parsing complexity remains polynomial. Its power is subsumed by context-free hypergraph grammars. The new formalism has an alternative characterization in terms of bimorphisms. An advantage over synchronous variants of linear context-free rewriting systems is the ability to specify tree-to-tree transductions.

1 Introduction

Machine translation involves mappings between strings in two languages, formalized as *string transductions*. Early models of string transductions include syntax-directed translation schemata (Lewis II and Stearns, 1968; Aho and Ullman, 1969b; Aho and Ullman, 1969a). These are precursors of more recent models of translation, such as inversion transduction grammars (Wu, 1997), and models in the Hiero system (Chiang, 2007). The underlying assumption in such models is that source and target languages are context-free, which is often too restrictive for practical applications. Therefore, more powerful models have been investigated, such as synchronous tree adjoining grammars (STAGs) (Shieber and Schabes, 1990), which assume that the translation to be modelled is between two tree adjoining languages. Such grammars offer an *extended domain of locality*, beyond the power of context-free grammars.

All of the above models translate between string pairs via a hierarchical structure (i.e. a parse tree) imposed on the source string and another such structure imposed on the target string. These formalisms therefore involve a mapping between parse trees, in addition to a mapping between strings. STAGs also involve derivation trees next to parse trees.

Translations between trees, formalized as *tree transductions*, are the main focus of formalisms such as top-down tree transducers (Rounds, 1970; Thatcher, 1970) and bottom-up tree transducers (Thatcher, 1973). These have attracted much interest in the area of statistical machine translation (SMT) (Knight and Graehl, 2005). Recent developments include (Engelfriet et al., 2009; Maletti, 2011; Maletti, 2012).

The rationale for treating tree transductions as an isolated issue in machine translation is one of modularity: parsing a source sentence to produce a parse tree is challenging enough to be investigated as a separate task, next to the problem of transferring the source-language structure to the target-language structure.

The awareness that phrase structure may be discontinuous, and hence exceeds the power of context-free formalisms, has been growing steadily over the past few years, owing to treebanks for many different languages. See for example (Kallmeyer et al., 2009) for evidence that synchronous rewriting cannot be avoided. The 'gap degree' found in some treebanks in fact even exceeds the power of tree adjoining grammars (Gómez-Rodríguez et al., 2011). This suggests that more powerful formalisms such as linear context-free rewriting systems (LCFRSs) (Vijay-Shanker et al., 1987) may be needed.

While LCFRSs induce derivation trees, they lack a natural notion of derived trees. As a consequence, transduction between strings via synchronous LCFRSs do not, in any obvious way, involve source-language and target-language parse trees. This complicates modular design of machine translation systems, in which parsing/generation of the source/target languages is separated from transfer of structures across the two languages.

The purpose of the present paper is to remedy this by introducing a formalism that combines the flexibility of synchronous context-free and synchronous tree adjoining grammars, with some of the additional generative capacity offered by LCFRSs. The new formalism consists of pairs of simple context-free tree grammars (sCFTGs) (Rounds, 1970; Engelfriet and Schmidt, 1977; Engelfriet and Schmidt, 1978), which are coupled through synchronous rewriting. The relevance of sCFTG to natural language processing is suggested by recent findings involving lexicalization of tree adjoining grammars (Maletti and Engelfriet, 2012).

Among the properties that make the new formalism suitable for applications in machine translation are the following. First, it is based on tree transductions, but indirectly also describes string transductions. It can therefore be used to translate strings to strings, but also trees to trees, allowing separate modules to handle parsing/generation. Second, its generative capacity contains that of synchronous tree adjoining grammars, offering the potential to handle some difficult cases of non-projective linguistic structures. Third, parsing complexity is polynomial in the size of the input string or the input tree. Fourth, the formalism can be straightforwardly extended to assign probabilities to rules, whereby probability distributions can be defined, both on the set of pairs of trees, and on the set of pairs of strings.

2 Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ and $\mathbb{N}^+ = \mathbb{N} \backslash \{0\}$. For each $n \in \mathbb{N}$, we let $[n]$ stand for the set $\{1, \ldots, n\}$, with $[0] = \emptyset$.

A *ranked alphabet* is a finite set Σ of symbols, associated with a rank function assigning a number $\mathrm{rk}_\Sigma(\sigma) \in \mathbb{N}$ to each symbol $\sigma \in \Sigma$. We write rk for rk_Σ when the alphabet Σ is understood. We let $\Sigma^{(k)}$ denote $\{\sigma \in \Sigma \mid \mathrm{rk}_\Sigma(\sigma) = k\}$.

We fix an infinite list $x_1, x_2, \ldots$ of pairwise distinct *variables*. We write $X = \{x_1, x_2, x_3, \ldots\}$ and $X_k = \{x_1, \ldots, x_k\}$. We denote the set of all ordered, labelled *trees* over ranked alphabet Σ, with variables in set $Y \subseteq X$, by $T_\Sigma(Y)$ We define T_Σ to be $T_\Sigma(\emptyset)$. If $\sigma \in \Sigma^{(0)}$, we may abbreviate $\sigma()$ to σ. Very often we will deal with sequences of variables such as $x_1, \ldots, x_k$, which we may then write in the abbreviated notation $x_{1,k}$. The same hold for sequences of trees; e.g. $t_{1,k} = t_1, \ldots, t_k$.

The *yield* of a tree t is the string of symbols in t that have rank 0, that is, the leaves, read from left to right. Positions in trees are identified by Gorn addresses, represented as strings of natural numbers as usual. The set of all positions in a tree t is denoted by $\mathrm{pos}(t)$. The *label* at position p of a tree $t \in T_\Sigma(Y)$ is denoted by $t(p)$ and the subtree of t at p is denoted by $t|_p$. The expression $t[s]_p$ denotes the tree obtained from t by replacing the subtree at position p by $s \in T_\Sigma(Y)$.

The set of positions in a tree t labelled by a symbol $a \in \Sigma \cup X$ is defined as $\mathrm{pos}_a(t) = \{p \mid t(p) = a\}$. For finite Y, the subset of $T_\Sigma(Y)$ consisting of those trees in which every variable in Y occurs precisely once is denoted by $C_\Sigma(Y)$.

If $t \in T_\Sigma(X_k)$ and $t_i \in T_\Sigma$ ($i \in [k]$), then the *first-order substitution* $t[t_{1,k}]$ denotes the tree t in which each occurrence of the variable x_i has been replaced by the corresponding tree t_i

If $t \in T_\Sigma(Y)$, $t(p) \in \Sigma^{(k)}$ and $s \in T_\Sigma(X_k)$, then the *second-order substitution* $t[\![s]\!]_p$ denotes the tree obtained from t in which the subtree at position p has been replaced by s, with the variables in s replaced by the corresponding immediate subtrees of $t|_p$, or formally $t[\![s]\!]_p = t[s[t|_{p1}, \ldots, t|_{pk}]]_p$.

3 CFTGs

A *context-free tree grammar (with states)* (CFTG) is a tuple $G = (Q, q_0, \Sigma, R)$, where:

- Q is a ranked alphabet (of *states*),
- $q_0 \in Q^{(0)}$ (*initial state*),
- Σ is a ranked alphabet (of *terminals*), such that $Q \cap \Sigma = \emptyset$, and
- R is a finite set (of *rules*), each of the form $q(x_{1,k}) \to \tau$, where $q \in Q^{(k)}$ and $\tau \in T_{Q\cup\Sigma}(X_k)$.

We write $\Rightarrow_G^{p,r}$ for the ‘derives’ relation, using rule $r = q(x_{1,k}) \to \tau$ at position p of a tree. Formally, we write $t \Rightarrow_G^{p,r} t'$ if $t \in T_{Q\cup\Sigma}$, $t(p) = q$

and $t' = t[\![\tau]\!]_p$. We write $t \Rightarrow_G t'$ if $t \Rightarrow_G^{p,r} t'$ for some p and r, and $\Rightarrow_G^*$ is the reflexive, transitive closure of $\Rightarrow_G$. The tree language induced by CFTG G is $[\![G]\!] = \{t \in T_\Sigma \mid q_0 \Rightarrow_G^* t\}$. The string language induced by G is $[G] = \{\mathrm{yield}(t) \mid t \in [\![G]\!]\}$.

In the sequel we will focus our attention on CFTGs where every rule is linear and nondeleting. Formally, a *simple* CFTG (sCFTG) is a CFTG where $\tau \in C_{Q\cup\Sigma}(X_k)$ for each rule $q(x_{1,k}) \to \tau$.

A CFTG G is a *regular tree grammar* (RTG) if $Q = Q^{(0)}$. We assume a normal form for RTG in which right-hand side trees contain precisely one terminal. The tree languages induced by RTGs are called *regular tree languages*.

Example 1 Fig. 1 shows a sCFTG allowing conjunctions, under the assumption that both parts share the same structure. The tree language contains:
$S(NP(John), VP(loves, and, eats))$, and
$S(NP(John), VP(VP(loves, haggis), and, VP(eats, it)))$,
but not for example:
$S(NP(John), VP(VP(loves, haggis), and, eats))$, nor
$S(NP(John), VP(loves, and, VP(eats, it)))$.

Note that if we modify the grammar to be recursive, for example by changing the first two occurrences of q_3 into q_2, then the string language is related to the copy language $\{ww \mid w \in \{a, b\}^*\}$. It is well-known that the copy language is induced by a tree adjoining grammar. However, the structure of the corresponding trees would be very different from the trees induced by our example sCFTG, and the latter arguably have a more direct linguistic interpretation. □

$$
\begin{aligned}
& q_0 \to S(NP(John), q_1(loves, eats)) \\
& q_1(x_1, x_2) \to q_2(VP(x_1, haggis), VP(x_2, it)) \\
& q_1(x_1, x_2) \to q_2(x_1, x_2) \\
& q_2(x_1, x_2) \to \\
& \qquad q_3(VP(x_1, dearly), VP(x_2, often)) \\
& q_2(x_1, x_2) \to \\
& \qquad q_3(VP(x_1, truly), VP(x_2, seldom)) \\
& q_2(x_1, x_2) \to q_3(x_1, x_2) \\
& q_3(x_1, x_2) \to VP(x_1, and, x_2)
\end{aligned}
$$

Figure 1: Rules of an example sCFTG modelling two parts of a conjunction being developed in tandem, where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{S, NP, VP, John, loves, \ldots\}$.

4 Synchronous CFTGs

We now take a pair of simple CFTGs and synchronize their derivations. For this, we need to represent bijections between occurrences of states in two trees. This is realized by annotating states with *indices*. More precisely, we define $I(Q) = \{q^{\boxed{u}} \mid q \in Q, u \in \mathbb{N}^+\}$. For $t \in C_{I(Q)\cup\Sigma}(Y)$ and $u \in \mathbb{N}^+$, we let $\mathrm{pos}_u(t)$ denote the set of positions where u occurs as index of a state, or formally, $\mathrm{pos}_u(t) = \{p \mid \exists q[t(p) = q^{\boxed{u}}]\}$. For $n \in \mathbb{N}$, we define $I^n_{Q,\Sigma}(Y)$ to be the set of trees where each index from 1 to n occurs precisely once and no other indices are present, or formally:

$$
\begin{aligned}
I^n_{Q,\Sigma}(Y) = \{t \in C_{I(Q)\cup\Sigma}(Y) \mid \\
\forall u[u \le n \implies |\mathrm{pos}_u(t)| = 1, \\
u > n \implies |\mathrm{pos}_u(t)| = 0]\}
\end{aligned}
$$

We let $I^n_{Q,\Sigma}$ denote $I^n_{Q,\Sigma}(\emptyset)$.

A pair $[t_1, t_2]$ of trees is called *synchronous* if each contains unique occurrences of all indices from 1 to n and no others, or formally, $t_1 \in I^n_{Q,\Sigma}(Y_1)$ and $t_2 \in I^n_{Q,\Sigma}(Y_2)$ for the same value of n. We call n the *synchronization breadth* of $[t_1, t_2]$.

A *synchronous (simple) CFTG* (SCFTG) is a tuple $G = (Q, q_0, \Sigma, R)$, where Q, q_0, and Σ are as for CFTGs, and R is a set of *synchronous rules*, each of which is of the form:

$$[q(x_{1,k}) \to \tau_1,\ q'(x_{1,m}) \to \tau_2] \qquad (1)$$

where $q \in Q^{(k)}$, $q' \in Q^{(m)}$, and $\tau_1 \in I^n_{Q,\Sigma}(X_k)$ and $\tau_2 \in I^n_{Q,\Sigma}(X_m)$ for some n. We note that $[\tau_1, \tau_2]$ is a synchronous tree pair. The *synchronization breadth* of a rule of the form (1) is the synchronization breadth of $[\tau_1, \tau_2]$.

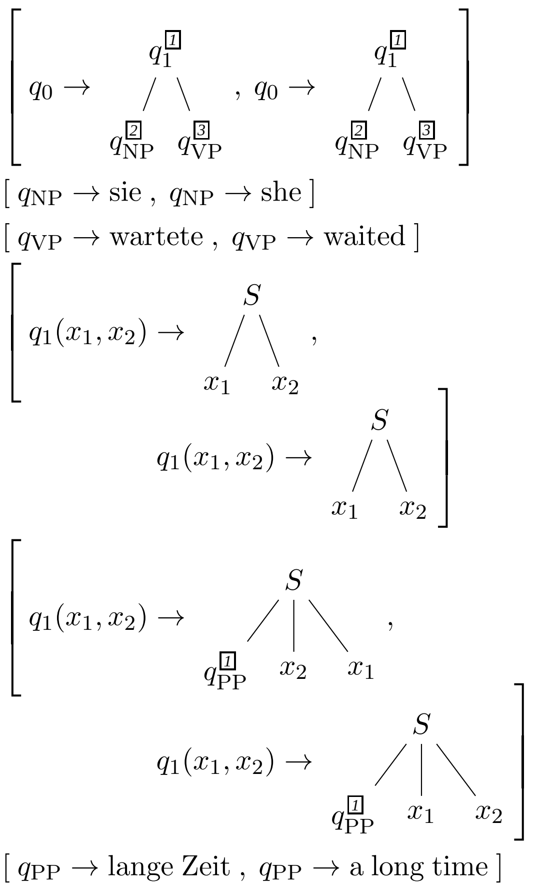
In order to define the binary 'derives' relation $\Rightarrow_G^{u,r}$ between synchronous pairs of trees, we need the additional notion of *reindexing*. This is an injective function that replaces each existing index in the synchronous pair by another, making sure the new indices do not clash with those of a chosen rule r. More precisely, let t_1 and t_2 be two synchronous trees in $I^{n'}_{Q,\Sigma}$. Choose an index $u \in [n']$ and determine the unique positions p and p' such that $t_1(p) = q^{\boxed{u}}$ and $t_2(p') = q'^{\boxed{u}}$, for some q and q'. Further, choose a synchronous rule r of the form (1). Depending on u, we define the reindexing function f as follows:

- $f(v) = n' + v$ if $v < u$,
- $f(v) = n' + v - 1$ if $v > u$,
- the value of $f(u)$ can be arbitrarily chosen (it will be ignored in the rewriting step).

For $i = 1, 2$, let $f(t_i)$ be t_i in which every index v is replaced by $f(v)$. We can now formally define $[t_1, t_2] \Rightarrow_G^{u,r} [t'_1, t'_2]$ to hold if and only if $t'_1 = f(t_1)[\![\tau_1]\!]_p$ and $t'_2 = f(t_2)[\![\tau_2]\!]_{p'}$. It is easy to show that $t'_1, t'_2 \in I_{Q,\Sigma}^{n+n'-1}$. In other words, one derivation step turns a synchronous tree pair $[t_1, t_2]$ into another.

For SCFTG G, we write $[t_1, t_2] \Rightarrow_G [t'_1, t'_2]$ if $[t_1, t_2] \Rightarrow_G^{u,r} [t'_1, t'_2]$ for some u and r, and $\Rightarrow_G^*$ is the reflexive, transitive closure of $\Rightarrow_G$. The tree transduction induced by SCFTG G is $[\![G]\!] = \{[t_1, t_2] \in T_\Sigma \times T_\Sigma \mid [q_0, q_0] \Rightarrow_G^* [t_1, t_2]\}$. The string transduction induced by G is $[G] = \{[\text{yield}(t_1), \text{yield}(t_2)] \mid [t_1, t_2] \in [\![G]\!]\}$.

Example 2 Fig. 2 shows a SCFTG. On the input side it models inversion of subject and main verb following an adverbial phrase in German. □



$q_0 \rightarrow$
$q_1^{\boxed{1}}$
$q_{\text{NP}}^{\boxed{2}}$ $q_{\text{VP}}^{\boxed{3}}$
$q_0 \rightarrow$
$q_1^{\boxed{1}}$
$q_{\text{NP}}^{\boxed{2}}$ $q_{\text{VP}}^{\boxed{3}}$
[$q_{\text{NP}} \rightarrow$ sie , $q_{\text{NP}} \rightarrow$ she]
[$q_{\text{VP}} \rightarrow$ wartete , $q_{\text{VP}} \rightarrow$ waited]
$q_1(x_1, x_2) \rightarrow$
S
x_1 x_2
$q_1(x_1, x_2) \rightarrow$
S
x_1 x_2
$q_1(x_1, x_2) \rightarrow$
S
$q_{\text{PP}}^{\boxed{1}}$ x_2 x_1
$q_1(x_1, x_2) \rightarrow$
S
$q_{\text{PP}}^{\boxed{1}}$ x_1 x_2
[$q_{\text{PP}} \rightarrow$ lange Zeit , $q_{\text{PP}} \rightarrow$ a long time]

Figure 2: Rules of an SCFTG.

5 Bimorphism characterization

Next we investigate a characterization of SCFTG in terms of generalized bimorphisms (Arnold and Dauchet, 1976; Arnold and Dauchet, 1982). A *bitransformation* (BT) is a tuple $B = (g, L, h)$ where:

- $L \subseteq T_\Delta$ is a regular tree language (*center language*), and
- $g \subseteq T_\Delta \times T_\Sigma$ (*input transformation*) and $h \subseteq T_\Delta \times T_\Sigma$ (*output transformation*) are tree transformations.

The BT B computes the tree transformation $[\![B]\!] \subseteq T_\Sigma \times T_\Sigma$, which is defined by:

$$[\![B]\!] = g^{-1} \,;\, \text{id}_L \,;\, h$$

where id_L is the binary identity relation on L and the semicolon denotes (left to right) composition of binary relations.

If the input and output tree transformation are tree homomorphisms, then the BT is a bimorphism in the sense of (Arnold and Dauchet, 1976; Arnold and Dauchet, 1982) For our characterization of SCFTG in terms of bitransformations we need stronger input/output transformations however. For this we recall the concept of macro tree transducer (Engelfriet, 1980; Courcelle and Franchi-Zannettacci, 1982). It can be seen as the combination of the concepts of top-down tree transducer and context-free tree grammar, and serves as formal model for syntax-directed semantics (Engelfriet, 1982) in which context can be handled.

Formally, a *macro tree transducer* (MAC) is a tuple $N = (Q, q_0, \Delta, \Sigma, R)$ where Q is a ranked alphabet (of *states*) with $Q^{(0)} = \emptyset$, $q_0 \in Q^{(1)}$ (*initial state*), Δ and Σ are ranked alphabets (of *input symbols* and *output symbols*, resp.) with $Q \cap (\Delta \cup \Sigma) = \emptyset$, and R is a finite set of rules of the form:

$$q(\delta(y_{1,n}), x_{1,k}) \rightarrow \zeta \qquad (2)$$

where $n, k \geq 0$, $q \in Q^{(k+1)}$, $\delta \in \Delta^{(n)}$, $y_1, \ldots, y_n$ and $x_1, \ldots, x_k$ are input and output variables ranging over T_Δ and T_Σ, resp., and $\zeta \in \mathrm{RHS}(n,k)$, where $\mathrm{RHS}(n,k)$ is the smallest subset RHS such that (i) $x_i \in \mathrm{RHS}$ for every $i \in [k]$, (ii) $\sigma(\zeta_{1,m}) \in \mathrm{RHS}$ for every $m \in \mathbb{N}$, $\sigma \in \Sigma^{(m)}$, and $\zeta_1, \dots, \zeta_m \in \mathrm{RHS}$, and (iii) $q'(y_j, \zeta_{1,m}) \in \mathrm{RHS}$ for every $j \in [n]$, $q' \in Q^{(m+1)}$, and $\zeta_1, \dots, \zeta_m \in \mathrm{RHS}$.

A MAC M is *linear and nondeleting* if for each rule of the form (2), ζ contains exactly one occurrence of each y_j ($j \in [n]$) and one of each x_i ($i \in [k]$), and contains no other variables. It is *pure* if $|Q^{(m)}| \leq 1$ for every $m \in \mathbb{N}$. It is *monadic* if $Q = Q^{(1)} \cup Q^{(2)}$. It is *total and deterministic* if for each $q \in Q$ and $\delta \in \Delta$ there is exactly one rule with q and δ in its left-hand side. A MAC M is called an *enriched embedded tree transducer* (eEMB) if it is linear and nondeleting, pure, and total and deterministic; an eEMB M is called an *embedded tree transducer* (EMB) (Shieber, 2006) if it is monadic.

Based on the concept of term rewriting, we can define the binary derivation relation $\Rightarrow_N$ of N in the usual way. The tree transformation computed by N is the set $[\![N]\!] = \{[t_1, t_2] \in T_\Delta \times T_\Sigma \mid q_0(t_1) \Rightarrow_N^* t_2\}$.

Theorem 1. *Let $\mathcal{T} \subseteq T_\Delta \times T_\Delta$. Then the following are equivalent.*

1. *There is a SCFTG G such that $\mathcal{T} = [\![G]\!]$.*
2. *There are eEMBs M_1 and M_2 and a regular tree language L such that $\mathcal{T} = [\![([\![M_1]\!], L, [\![M_2]\!])]\!]$.*

Proof. $1 \Rightarrow 2$. Let $G = (Q, q_0, \Sigma, R)$ be a SCFTG. We construct the RTG $H = (Q \times Q, (q_0, q_0), R, R')$ where $\mathrm{rk}_R(r)$ is the synchronization breadth of r for each $r \in R$, and R' is constructed as follows. Let G contain a rule r of the form (1) with synchronization breadth n. Moreover, let $q_1^{\boxed{1}}, \dots, q_n^{\boxed{n}}$ and $q_1'^{\boxed{1}}, \dots, q_n'^{\boxed{n}}$ be all the occurrences of indexed states in τ_1 and τ_2, resp. Then R' contains the rule:

$$(q, q') \to r\big((q_1, q_1'), \dots, (q_n, q_n')\big)$$

We construct the eEMB $M_1 = (Q_1, *_0, R, \Sigma, R_1)$ where $Q_1 = \{*_j \mid Q^{(j)} \neq \emptyset\}$ and $\mathrm{rk}_{Q_1}(*_j) = j + 1$. Let G contain a rule r of the form (1) as in the construction of H. Then R_1 contains the rule:

$$*_k(r(y_{1,n}), x_{1,k}) \to \tau_1'$$

where τ_1' is obtained from τ_1 by recursively replacing every subtree of the form $q_i^{\boxed{i}}(t_{1,\ell})$ by $*_\ell(y_i, t'_{1,\ell})$. In a similar way we can define the eEMB M_2 using τ_2 and m instead of τ_1 and k, resp. We can prove that $[\![G]\!] = [\![([\![M_1]\!], L(H), [\![M_2]\!])]\!]$.

Conversely, let $M_1 = (Q_1, q_{0,1}, \Delta, \Sigma, R_1)$ and $M_2 = (Q_2, q_{0,2}, \Delta, \Sigma, R_2)$ be two eEMBs and $H = (Q, q_0, \Delta, R)$ be a RTG in normal form. We construct the SCFTG $G = (Q', q_0', \Sigma, R')$ where $Q' = \{(q, i) \mid q \in Q, i \in \mathbb{N}^+, Q_1^{(i)} \cup Q_2^{(i)} \neq \emptyset\}$ and $\mathrm{rk}_{Q'}((q, i)) = i$. Now let:

$q \to \delta(q_1, \dots, q_n)$ be a rule in R,
$q'(\delta(y_{1,n}), x_{1,k}) \to \zeta_1$ a rule in R_1, and
$q''(\delta(y_{1,n}), x_{1,m}) \to \zeta_2$ a rule in R_2.

Then R' contains the rule:

$$[(q, k)(x_{1,k}) \to \zeta_1',\ (q, m)(x_{1,m}) \to \zeta_2']$$

where ζ_1' is obtained from ζ_1 by recursively replacing every subtree of the form $\bar{q}(y_j, t_{1,\ell})$ by $(q_j, \ell)^{\boxed{j}}(t'_{1,\ell})$. In a similar way we obtain ζ_2' from ζ_2. We can prove that $[\![G]\!] = [\![([\![M_1]\!], [\![H]\!], [\![M_2]\!])]\!]$. □

6 Parsing

In SMT it has become commonplace to use a combination of relatively powerful syntactic models akin to context-free grammars, and weaker models of finite-state power. The theoretical foundation is the result by (Bar-Hillel et al., 1964), allowing the construction of a context-free grammar inducing the intersection of two languages, one induced by a given context-free grammar and another induced by a given finite automaton. The technique carries over to several other grammatical formalisms, and to tree languages next to string languages. In the realm of synchronous grammars, moreover, the technique generalizes to input products and output products.

The *input product* of a tree transformation $\mathcal{T} \subseteq T_\Sigma \times T_\Sigma$ and a tree language $L \subseteq T_\Sigma$, denoted by $L \lhd \mathcal{T}$, is defined as the tree transformation $\mathrm{id}_L; \mathcal{T}$. Similarly, we define the *output product* as $\mathcal{T} \rhd L = \mathcal{T}; \mathrm{id}_L$.

In this section, we consider application of the technique to SCFTGs and RTGs.

Theorem 2. *If G is a SCFTG and H is a RTG, then there are SCFTGs G' and G'' such that $[\![G']\!] = [\![H]\!] \lhd [\![G]\!]$ and $[\![G'']\!] = [\![G]\!] \rhd [\![H]\!]$.*

Proof. We prove closure under input product; the proof for output product is similar. By Theorem 1 there are eEMBs M_1 and M_2 and a regular tree language L such that $[\![G]\!] = [\![([\![M_1]\!], L, [\![M_2]\!])]\!] = [\![M_1]\!]^{-1}; \mathrm{id}_L; [\![M_2]\!]$. Therefore:

$$\begin{aligned} & [\![H]\!] \lhd [\![G]\!] \\ = \; & \mathrm{id}_{[\![H]\!]}; [\![M_1]\!]^{-1}; \mathrm{id}_L; [\![M_2]\!] \\ = \; & [\![M_1]\!]^{-1}; \mathrm{id}_{L \cap [\![M_1]\!]^{-1}([\![H]\!])}; [\![M_2]\!] \end{aligned}$$

Since the class of regular tree languages is closed under intersection and under the inverse of macro tree transformations (cf. Thm. 7.4 of (Engelfriet and Vogler, 1985)), $L' = L \cap [\![M_1]\!]^{-1}([\![H]\!])$ is a regular tree language. Hence $[\![H]\!] \lhd [\![G]\!] = [\![([\![M_1]\!], L', [\![M_2]\!])]\!]$ is induced by a SCFTG, once more by Theorem 1. □

In the following, we give a direct construction of the SCFTG G' mentioned in Theorem 2 The style of the construction is close to that by (Büchse et al., 2011).

Let $G = (Q, q_0, \Sigma, R)$ be a SCFTG and $H = (Q_H, s_0, \Sigma, R_H)$ be a RTG. The constructed SCFTG G' is of the form $(Q', (q_0, s_0), \Sigma, R')$, where Q' is defined by $\bigcup_k \; Q^{(k)} \times Q_H^{k+1}$ and R' is defined below.

The intuition is that we explore all portions of trees that can be parsed simultaneously by H and by the CFTG that is composed of the input parts of the rules of G. For this purpose, we construct the RTG $H(r, s, \theta) = (Q_H, s, \Sigma \cup Q \cup X_k, R_\theta)$, for each rule $r \in R$ of the form (1), each $s \in Q_H$ and each function θ that maps:

- each indexed state $q^{\boxed{u}}$ in τ_1 to a sequence of $\mathrm{rk}(q) + 1$ states from Q_H, and
- each variable $x \in X_k$ to a state from Q_H.

The rules in R_θ include all rules from R_H and in addition:

- $s' \to q^{\boxed{u}}(s_1 \cdots s_{\mathrm{rk}(q)})$ for each indexed state $q^{\boxed{u}}$ in τ_1 such that $\theta(q^{\boxed{u}}) = s' s_1 \cdots s_{\mathrm{rk}(q)}$, and
- $s' \to x$ for each $x \in X_k$ such that $\theta(x) = s'$.

If τ_1 is in the tree language induced by $H(r, s, \theta)$, then we say that (s, θ) is *input-consistent* for r.

We can now define R' to contain one rule:

$$[q_1'(x_{1,k}) \to \tau_1', \; q_2(x_{1,m}) \to \tau_2]$$

for each rule r from R of the form:

$$[q_1(x_{1,k}) \to \tau_1, \; q_2(x_{1,m}) \to \tau_2] \qquad (3)$$

and each s and θ such that (s, θ) is input-consistent for r, where $q_1' = (q_1, s\theta(x_1) \ldots \theta(x_k))$ and τ_1' results from τ_1 by replacing each $q^{\boxed{u}}$ by $\theta(q^{\boxed{u}})^{\boxed{u}}$.

Let $q_1^{\boxed{1}}, \ldots, q_n^{\boxed{n}}$ be all indexed states in τ_1. Then there are up to $|Q_H|^C$ choices of (s, θ), where $C = 1 + k + \sum_{j \in [n]} 1 + \mathrm{rk}(q_j)$. Let $C_{\max}$ be the maximum value of C over different rules r. For checking whether a choice of (s, θ) is input-consistent for given r, we need to match at most $|R_H|$ rules at each position of $H(r, s, \theta)$ that is labelled with a terminal. Summing over all rules r, this means that R' can be constructed in time $\mathcal{O}(|G|_{in} \cdot |R_H| \cdot |Q_H|^{C_{\max}})$, where $|G|_{in}$ is defined as $\sum_{r \in R} |\mathrm{pos}(\tau_1(r))|$, where $\tau_1(r)$ denotes τ_1 assuming r is of the form (3). Deciding whether the input product is empty amounts to deciding whether all rules are useless. As for context-free grammars (Sippu and Soisalon-Soininen, 1988), this can be decided in linear time in the size of the grammar.

The input product can be used to realize recognition of strings, as follows. Given ranked alphabet Σ, one can construct a RTG H inducing T_Σ. Given a string $w = a_1 \cdots a_n$, with $a_i \in \Sigma^{(0)}$ for $i \in [n]$, one can construct the RTG H_w from H such that $[\![H_w]\!] = \{t \in [\![H]\!] \mid \mathrm{yield}(t) = w\}$, by the usual technique of intersection (Bar-Hillel et al., 1964). The number of rules of H_w is $\mathcal{O}(|\Sigma| \cdot n^{D+1})$, where D is $\max\{\mathrm{rk}(\sigma) \mid \sigma \in \Sigma\}$.

Deciding whether $(w, v) \in [G]$ for some v can now be done by deciding whether the input product of H_w and G is non-empty. By the above analysis, this can be done in polynomial time in n, assuming G and thereby Σ are fixed. As a side-effect of recognition, one obtains a SCFTG G' inducing the tree transduction $\{[t_1, t_2] \in [\![G]\!] \mid \mathrm{yield}(t_1) = w\}$. Appropriate output trees t_2 can subsequently be extracted from G'.

7 Relation to other formalisms

We now relate SCFTG to other formalisms that are relevant for machine translation. First, we return to macro tree transducers, which were discussed before in Section 5.

Theorem 3. *Linear, nondeleting macro tree transducers are strongly equivalent to SCFTGs in* *which rules have the form:*

$$[q \to \sigma(q_1^{\boxed{1}}, \ldots, q_k^{\boxed{k}}),\ q'(x_{1,m}) \to \tau] \qquad (4)$$

Proof. Let $M = (Q, q_0, \Delta, \Sigma, R)$ be a linear, nondeleting MAC. We construct the SCFTG $G = (\bar{Q}, q_{\text{in}}, \Delta \cup \Sigma, R')$ where $\bar{Q} = Q \cup Q' \cup \{q_{\text{in}}\}$, $Q' = \{q' \mid q \in Q\}$, q_{in} is a new state, and $\text{rk}_{\bar{Q}}(q) = \text{rk}_Q(q) - 1$ and $\text{rk}_{\bar{Q}}(q') = 0$ for each $q \in Q$, and $\text{rk}_{\bar{Q}}(q_{\text{in}}) = 0$. Let $r \in R$ be of the form (2). For each $j \in [k]$, let p_j be the unique position such that $\zeta(p_j 1) = y_j$ and let $q_j = \zeta(p_j)$. Then R' contains the rule:

$$[q' \to \delta(q_1'^{\boxed{1}}, \ldots, q_k'^{\boxed{k}}),\ q(x_{1,n}) \to \zeta']$$

where ζ' is obtained from ζ by recursively replacing every subtree of the form $q_j(y_j, \zeta_{1,m})$ by $q_j^{\boxed{j}}(\zeta'_{1,m})$. In addition, R' contains the initial rule $[q_{\text{in}} \to q'_0, q_{\text{in}} \to q_0]$. It can be proven that $[\![M]\!] = [\![G]\!]$.

Conversely, let $G = (Q, q_0, \Sigma, R)$ be a SCFTG in which each rule has the form (4). We construct the MAC $M = (Q', (q_0, q_0), \Sigma, \Sigma, R')$ where $Q' = Q \times Q$ and $\text{rk}_{Q'}((q, q')) = \text{rk}_Q(q') + 1$ for every $(q, q') \in Q'$, and if R contains a rule of the form (4), then R' contains the rule:

$$(q, q')(\sigma(y_{1,k}), x_{1,m}) \to \tau'$$

where τ' is obtained from τ by recursively replacing every subtree of the form $q''^{\boxed{j}}(\tau_{1,n})$ by $(q_j, q'')(y_j, \tau'_{1,n})$. It can be proven that $[\![G]\!] = [\![M]\!]$. $\square$

Synchronous tree-adjoining grammar (STAG) (Shieber and Schabes, 1990) captures mildly context-sensitive phenomena in natural languages. STAGs with states (Büchse et al., 2011; Büchse et al., 2012) are characterized by bitransformations in which the input and output transformations are EMBs (Shieber, 2006). Thus, in view of Theorem 1, every STAG with states can be simulated by a SCFTG.

Synchronous tree-substitution grammar (STSG) (Schabes, 1990) is STAG without adjoining. STSGs with states (Fülöp et al., 2010) are characterized by bitransformations in which the input and output transformations are linear, nondeleting tree homomorphisms (Shieber, 2004) (also cf. Thm. 4 of (Fülöp et al., 2010)).

Extended top-down tree transducers (XTOP) (Rounds, 1970; Arnold and Dauchet, 1976) and *extended bottom-up tree transducers* (XBOT) (Fülöp et al., 2011) are top-down tree transducers and bottom-up tree transducers, resp., in which the input patterns occurring in the left-hand sides of rules may have arbitrary depth. XTOPs have been used to specify e.g. English-Arabic translation (Maletti et al., 2009). The linear, nondeleting restrictions of XTOP and XBOT are denoted by ln-XTOP and ln-XBOT, respectively, and both classes are strongly equivalent (cf. Prop. 3.3 of (Fülöp et al., 2011)). Moreover, nl-XTOP (and hence, nl-XBOT) is strongly equivalent to STSG with states, because these classes have the same bimorphism characterization (Arnold and Dauchet, 1976) (also cf. Thm. 4.2 of (Fülöp et al., 2011)). Hence, the power of nl-XTOP and nl-XBOT is subsumed by SCFTG.

A *linear context-free rewriting system* (LCFRS) (Vijay-Shanker et al., 1987) is a string-generating device that can be thought of a context-free grammar in which each nonterminal has a fixed number of parameter positions, each of which contains a string. Moreover, each rule specifies how to synthesize the strings contained in the parameters on its right-hand side to make up the strings for the parameters on its left-hand side. In fact, LCFRSs are attribute grammars with synthesized attributes only (Knuth, 1968) interpreted over the set of strings with concatenation. LCFRGs are weakly equivalent to multiple context-free grammars (MCFGs) (Seki et al., 1991).

The string languages induced by linear CFTGs are the same as those induced by *well-nested* linear context-free rewriting systems (cf. footnote 3 of (Kanazawa, 2009)). A synchronous variant of well-nested LCFRSs can easily be defined in terms of generalized bimorphisms (see also (Bertsch and Nederhof, 2001)), but the connection to SCFTGs is yet to be clarified.

Context-free hypergraph grammars (CFHG) (Bauderon and Courcelle, 1987; Habel and Kreowski, 1987; Engelfriet and Heyker, 1991) are context-free grammars that generate hypergraphs. Each rule of a CFHG G specifies how a hyperedge, carrying a state and adjacent with n nodes, is replaced by a hypergraph with n port (or interface) nodes. The set of derivation trees of G is a regular tree language. The hypergraph language induced by G is the set of all hypergraphs that only contain hyperedges labelled by terminals.

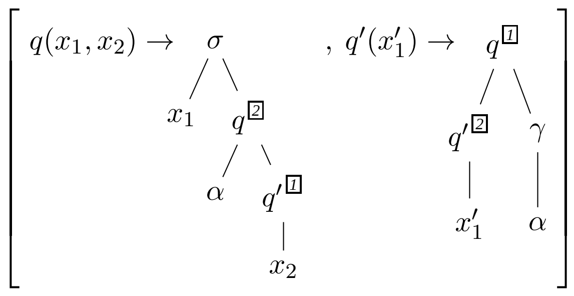


Figure 3: Rule of a SCFTG.

Every SCFTG G can be simulated by a CFHG H. Construction of H out of G is relatively straightforward, but available space does not allow a formal definition. Instead we give an example.

Example 3 Consider the SCFTG rule in Fig. 3, with states q and q' of rank 2 and 1, resp., and terminals σ, γ and α of rank 2, 1 and 0, resp.; the (only) variable in the output part is written x'_1 to distinguish it from x_1 in the input part. Fig. 4 shows the corresponding CFHG rule. A pair of synchronized states together form one hyperedge. Each pair of identically labelled nodes corresponds to a single node in the host graph to which this rule is applied, before and after the application. □

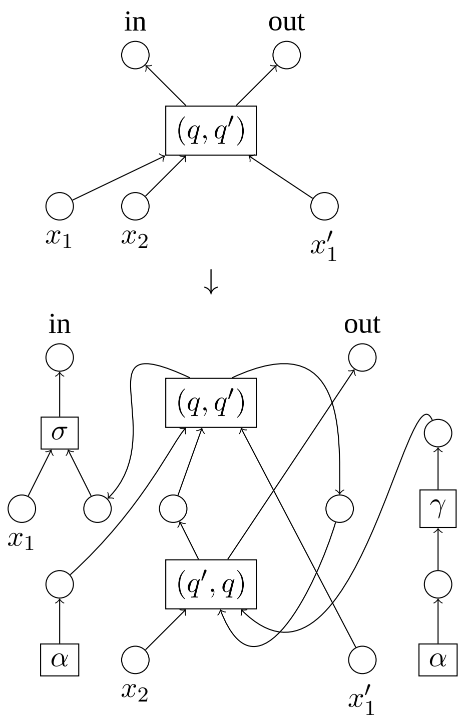


Figure 4: Rule of a CFHG.

References

A.V. Aho and J.D. Ullman. 1969a. Properties of syntax directed translations. *Journal of Computer and System Sciences*, 3:319–334.

A.V. Aho and J.D. Ullman. 1969b. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3:37–56.

A. Arnold and M. Dauchet. 1976. Bi-transduction de forêts. In S. Michaelson and R. Milner, editors, *Proc. 3rd International Colloquium on Automata, Languages and Programming*, pages 74–86. Edinburgh University Press.

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

Y. Bar-Hillel, M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Y. Bar-Hillel, editor, *Language and Information: Selected Essays on their Theory and Application*, chapter 9, pages 116–150. Addison-Wesley, Reading, Massachusetts.

M. Bauderon and B. Courcelle. 1987. Graph expressions and graph rewritings. *Mathematical Systems Theory*, 20:83–127.

E. Bertsch and M.-J. Nederhof. 2001. On the complexity of some extensions of RCG parsing. In *Proceedings of the Seventh International Workshop on Parsing Technologies*, pages 66–77, Beijing, China, October.

M. Büchse, M.-J. Nederhof, and H. Vogler. 2011. Tree parsing with synchronous tree-adjoining grammars. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 14–25, Dublin, Ireland, October.

M. Büchse, A. Maletti, and H. Vogler. 2012. Unidirectional derivation semantics for synchronous tree-adjoining grammars. In *Developments in Language Theory, 16th International Conference*, volume 7410 of *Lecture Notes in Computer Science*, Taipei, Taiwan. Springer-Verlag.

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

B. Courcelle and P. Franchi-Zannettacci. 1982. Attribute grammars and recursive program schemes. *Theoretical Computer Science*, 17:163–191.

J. Engelfriet and L. Heyker. 1991. The string generating power of context-free hypergraph grammars. *Journal of Computer and System Sciences*, 43:328–360.

J. Engelfriet and E.M. Schmidt. 1977. IO and OI. I. *Journal of Computer and System Sciences*, 15:328–353.

J. Engelfriet and E.M. Schmidt. 1978. IO and OI. II. *Journal of Computer and System Sciences*, 16:67–99.

J. Engelfriet and H. Vogler. 1985. Macro tree transducers. *Journal of Computer and System Sciences*, 31:71–146.

J. Engelfriet, E. Lilin, and A. Maletti. 2009. Composition and decomposition of extended multi bottom-up tree transducers. *Acta Informatica*, 46:561–590.

J. Engelfriet. 1980. Some open questions and recent results on tree transducers and tree languages. In R.V. Book, editor, *Formal language theory: perspectives and open problems*, pages 241–286. Academic Press, New York.

J. Engelfriet. 1982. Tree transducers and syntax-directed semantics. In *Proc. of 7th Colloquium on Trees in Algebra and Programming*, pages 82–107, Lille, France.

Z. Fülöp, A. Maletti, and H. Vogler. 2010. Preservation of recognizability for synchronous tree substitution grammars. In *Workshop on Applications of Tree Automata in Natural Language Processing*, pages 1–9, Uppsala, Sweden.

Z. Fülöp, A. Maletti, and H. Vogler. 2011. Weighted extended tree transducers. *Fundamenta Informaticae*, 111:163–202.

C. Gómez-Rodríguez, J. Carroll, and D. Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational Linguistics*, 37(3):541–586.

A. Habel and H.-J. Kreowski. 1987. Some structural aspects of hypergraph languages generated by hyperedge replacement. In *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science*, volume 247 of *Lecture Notes in Computer Science*, pages 207–219. Springer-Verlag.

L. Kallmeyer, W. Maier, and G. Satta. 2009. Synchronous rewriting in treebanks. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 69–72, Paris, France, October.

M. Kanazawa. 2009. The pumping lemma for well-nested multiple context-free languages. In *Developments in Language Theory*, volume 5583 of *Lecture Notes in Computer Science*, pages 312–325, Stuttgart, Germany. Springer-Verlag.

K. Knight and J. Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In A.F. Gelbukh, editor, *Proceedings of the Sixth Conference on Intelligent Text Processing and Computational Linguistics*, volume 3406 of *Lecture Notes in Computer Science*, Mexico City, Mexico.

D.E. Knuth. 1968. Semantics of context-free languages. *Mathematical Systems Theory*, 2:127–145. Corrections in Math. Systems Theory 5 (1971), 95-96.

P.M. Lewis II and R.E. Stearns. 1968. Syntax-directed transduction. *Journal of the ACM*, 15(3):465–488, July.

A. Maletti and J. Engelfriet. 2012. Strong lexicalization of tree adjoining grammars. In *50th Annual Meeting of the ACL*, pages 506–515, Jeju Island, Korea, July.

A. Maletti, J. Graehl, M. Hopkins, and K. Knight. 2009. The power of extended top-down tree transducers. *SIAM Journal on Computing*, 39:410–430.

A. Maletti. 2011. How to train your multi bottom-up tree transducer. In *49th Annual Meeting of the ACL*, pages 825–834, Portland, Oregon, June.

A. Maletti. 2012. Every sensible extended top-down tree transducer is a multi bottom-up tree transducer. In *Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 263–273, Montréal, Canada, June.

W.C. Rounds. 1970. Mappings and grammars on trees. *Mathematical Systems Theory*, 4:257–287.

Yves Schabes. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, University of Pennsylvania.

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

S.M. Shieber and Y. Schabes. 1990. Synchronous tree-adjoining grammars. In *Papers presented to the 13th International Conference on Computational Linguistics*, volume 3, pages 253–258.

S.M. Shieber. 2004. Synchronous grammars as tree transducers. In *Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 88–95, May.

S.M. Shieber. 2006. Unifying synchronous tree adjoining grammars and tree transducers via bimorphisms. In *Proceedings of the 11th Conference of the European Chapter of the ACL*, pages 377–384, Trento, Italy.

S. Sippu and E. Soisalon-Soininen. 1988. *Parsing Theory, Vol. I: Languages and Parsing*, volume 15 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag.

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

J.W. Thatcher. 1973. Tree automata: an informal survey. In A.V. Aho, editor, *Currents in the Theory of Computing*, pages 143–172. Prentice Hall, Englewood Cliffs.

K. Vijay-Shanker, D.J. Weir, and A.K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the ACL*, pages 104–111, Stanford, California, USA, July.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.