Generalized Multitext Grammars

I. Dan Melamed
Computer Science Department
New York University
715 Broadway, 7th Floor
New York, NY, 10003, USA
{lastname}@cs.nyu.edu

Giorgio Satta
Dept. of Information Eng'g
University of Padua
via Gradenigo 6/A
I-35131 Padova, Italy
{lastname}@dei.unipd.it

Benjamin Wellington
Computer Science Department
New York University
715 Broadway, 7th Floor
New York, NY, 10003, USA
{lastname}@cs.nyu.edu

Abstract

Generalized Multitext Grammar (GMTG) is a synchronous grammar formalism that is weakly equivalent to Linear Context-Free Rewriting Systems (LCFRS), but retains much of the notational and intuitive simplicity of Context-Free Grammar (CFG). GMTG allows both synchronous and independent rewriting. Such flexibility facilitates more perspicuous modeling of parallel text than what is possible with other synchronous formalisms. This paper investigates the generative capacity of GMTG, proves that each component grammar of a GMTG retains its generative power, and proposes a generalization of Chomsky Normal Form, which is necessary for synchronous CKY-style parsing.

1 Introduction

Synchronous grammars have been proposed for the formal description of parallel texts representing translations of the same document. As shown by Melamed (2003), a plausible model of parallel text must be able to express discontinuous constituents. Since linguistic expressions can vanish in translation, a good model must be able to express independent (in addition to synchronous) rewriting. Inversion Transduction Grammar (ITG) (Wu, 1997) and Syntax-Directed Translation Schema (SDTS) (Aho and Ullman, 1969) lack both of these properties. Synchronous Tree Adjoining Grammar (STAG) (Shieber, 1994) lacks the latter and allows only limited discontinuities in each tree.

Generalized Multitext Grammar (GMTG) offers a way to synchronize Mildly Context-Sensitive Grammar (MCSG), while satisfying both of the above criteria. The move to MCSG is motivated by our desire to more perspicuously account for certain syntactic phenomena that cannot be easily captured by context-free grammars, such as clitic climbing, extraposition, and other types of long-distance movement (Becker et al., 1991). On the other hand, MCSG still observes some restrictions that make the set of languages it generates less expensive to analyze than the languages generated by (properly) context-sensitive formalisms.

More technically, our proposal starts from Multitext Grammar (MTG), a formalism for synchronizing context-free grammars recently proposed by Melamed (2003). In MTG, synchronous rewriting is implemented by means of an indexing relation that is maintained over occurrences of nonterminals in a sentential form, using essentially the same machinery as SDTS. Unlike SDTS, MTG can extend the dimensionality of the translation relation beyond two, and it can implement independent rewriting by means of partial deletion of syntactic structures. Our proposal generalizes MTG by moving from component grammars that generate context-free languages to component grammars whose generative power is equivalent to Linear Context-Free Rewriting Systems (LCFRS), a formalism for describing a class of MCSGs. The generalization is achieved by allowing context-free productions to rewrite tuples of strings, rather than single strings. Thus, we retain the intuitive top-down definition of synchronous derivation original in SDTS and MTG but not found in LCFRS, while extending the generative power to linear context-free rewriting languages. In this respect, GMTG has also been inspired by the class of Local Unordered Scattered Context Grammars (Rambow and Satta, 1999). A syntactically very different synchronous formalism involving LCFRS has been presented by Bertsch and Nederhof (2001).

This paper begins with an informal description of GMTG. It continues with an investigation of this formalism's generative capacity. Next, we prove that in GMTG each component grammar retains its generative power, a requirement for synchronous formalisms that Rambow and Satta (1996) called the "weak language preservation property." Lastly, we propose a synchronous generalization of Chomsky Normal Form, which lays the groundwork for synchronous parsing under GMTG using a CKY-style algorithm (Younger, 1967; Melamed, 2004).

2 Informal Description and Comparisons

GMTG is a generalization of MTG, which is itself a generalization of CFG to the synchronous case. Here we present MTG in a new notation that shows the relation to CFG more clearly. For example, the following MTG productions can generate the multitext [(*I fed the cat*), (*ya kota kormil*)]:[1]

$$[(\mathrm{S}),(\mathrm{S})] \rightarrow [(\mathrm{PN}^1\mathrm{VP}^2),(\mathrm{PN}^1\mathrm{VP}^2)] \quad (1)$$
$$[(\mathrm{PN}),(\mathrm{PN})] \rightarrow [(\mathrm{I}),(\mathrm{ya})] \quad (2)$$
$$[(\mathrm{VP}),(\mathrm{VP})] \rightarrow [(\mathrm{V}^1\mathrm{NP}^2),(\mathrm{NP}^2\mathrm{V}^1)] \quad (3)$$
$$[(\mathrm{V}),(\mathrm{V})] \rightarrow [(\mathrm{fed}),(\mathrm{kormil})] \quad (4)$$
$$[(\mathrm{NP}),(\mathrm{NP})] \rightarrow [(\mathrm{D}^1\mathrm{N}^2),(\mathrm{N}^2)] \quad (5)$$
$$[(\mathrm{D}),()] \rightarrow [(\mathrm{the}),()] \quad (6)$$
$$[(\mathrm{N}),(\mathrm{N})] \rightarrow [(\mathrm{cat}),(\mathrm{kota})] \quad (7)$$

Each production in this example has two components, the first modeling English and the second (transliterated) Russian. Nonterminals with the same index must be rewritten together (synchronous rewriting). One strength of MTG, and thus also GMTG, is shown in Productions (5) and (6). There is a determiner in English, but not in Russian, so Production (5) does not have the nonterminal D in the Russian component and (6) applies only to the English component (independent rewriting). Formalisms that do not allow independent rewriting require a corresponding D to appear in the second component on the right-hand side (RHS) of Production (5), and this D would eventually generate the empty string. This approach has the disadvantage that it introduces spurious ambiguity about the position of the "empty" nonterminal with respect to the other nonterminals in its component. Spurious ambiguity leads to wasted effort during parsing.

GMTG's implementation of independent rewriting through the empty tuple () serves a very different function from the empty string. Consider the following GMTG:

$$[(S),(S)] \rightarrow [(a),(\varepsilon)] \quad (8)$$
$$[(S),(S)] \rightarrow [(X^1),(Y^2)] \quad (9)$$
$$[(X),()] \rightarrow [(b),()] \mid [(c),()] \mid [(d),()] \quad (10)$$
$$[(),(Y)] \rightarrow [(),(e)] \mid [(),(f)] \mid [(),(g)] \quad (11)$$

Production (8) asserts that symbol a vanishes in translation. Its application removes both of the nonterminals on the left-hand side (LHS), pre-empting any other production. In contrast, Production (9) explicitly relaxes the synchronization constraint, so that the two components can be rewritten independently. The other six productions make assertions about only one component and are agnostic about the other component. Incidentally, generating the same language with only fully synchronized productions would raise the number of required productions to 11, so independent rewriting also helps to reduce grammar size.

Independent rewriting is also useful for modeling paraphrasing. Take, for example, [(*Tim got a pink slip*), (*Tim got laid off*)]. While the two sentences have the same meaning, the objects of their verb phrases are structured very differently. GMTG can express their relationships as follows:

$$[(\mathrm{S}),(\mathrm{S})] \rightarrow [(\mathrm{NP}^1\mathrm{VP}^2),(\mathrm{NP}^1\mathrm{VP}^2)] \quad (12)$$
$$[(\mathrm{VP}),(\mathrm{VP})] \rightarrow [(\mathrm{V}^1\mathrm{NP}^2),(\mathrm{V}^1\mathrm{PP}^2)] \quad (13)$$
$$[(\mathrm{NP}),(\mathrm{PP})] \rightarrow [(\mathrm{DT}^1\mathrm{A}^2,\mathrm{N}^3),(\mathrm{VB}^4,\mathrm{R}^5)] \quad (14)$$
$$[(\mathrm{NP}),(\mathrm{NP})] \rightarrow [(\mathrm{Tim}),(\ \mathrm{Tim})] \quad (15)$$
$$[(\mathrm{V}),(\mathrm{V})] \rightarrow [(\mathrm{got}),(\mathrm{got})] \quad (16)$$
$$[(\mathrm{DT}),()] \rightarrow [(\mathrm{a}),()] \quad (17)$$
$$[(\mathrm{A}),()] \rightarrow [(\mathrm{pink}),()] \quad (18)$$
$$[(\mathrm{N}),()] \rightarrow [(\mathrm{slip}),()] \quad (19)$$
$$[(),(\mathrm{VB})] \rightarrow [(),(\mathrm{laid})] \quad (20)$$
$$[(),(\mathrm{R})] \rightarrow [(),(\mathrm{off})] \quad (21)$$

As described by Melamed (2003), MTG requires production components to be contiguous, except after binarization. GMTG removes this restriction. Take, for example, the sentence pair [(*The doctor treats his teeth*), (*El médico le examino los dientes*)] (Dras and Bleam, 2000). The Spanish clitic *le* and the NP *los dientes* should both be paired with the English NP *his teeth*, giving rise to a discontinuous constituent in the Spanish component. A GMTG fragment for the sentence is shown below:

$$[(\mathrm{S}),(\mathrm{S})] \rightarrow [(\mathrm{NP}^1\mathrm{VP}^2),(\mathrm{NP}^1\mathrm{VP}^2)]$$
$$[(\mathrm{VP}),(\mathrm{VP})] \rightarrow [(\mathrm{V}^1\mathrm{NP}^2),(\mathrm{NP}^2\mathrm{V}^1\mathrm{NP}^2)]$$
$$[(\mathrm{NP}),(\mathrm{NP})] \rightarrow [(\mathrm{The\ doctor}),(\mathrm{El\ médico})]$$
$$[(\mathrm{V}),(\mathrm{V})] \rightarrow [(\mathrm{treats}),(\mathrm{examino})]$$
$$[(\mathrm{NP}),(\mathrm{NP},\mathrm{NP})] \rightarrow [(\mathrm{his\ teeth}),(\mathrm{le},\mathrm{los\ dientes})]$$

Note the discontinuity between *le* and *los dientes*. Such discontinuities are marked by commas on both the LHS and the RHS of the relevant component.

GMTG's flexibility allows it to deal with many complex syntactic phenomena. For example, Becker et al. (1991) point out that TAG does not have the generative capacity to model certain kinds of scrambling in German, when the so-called "co-occurrence constraint" is imposed, requiring the

[1] We write production components both side by side and one above another to save space, but each component is always in parentheses.

derivational pairing between verbs and their complements. They examine the English/German sentence fragment [(*... that the detective has promised the client to indict the suspect of the crime*), (*... daß des Verbrechens der Detektiv den Verdächtigen dem Klienten zu überführen versprochen hat*)]. The verbs *versprochen* and *überführen* both have two noun phrases as arguments. In German, these noun phrases can appear to the left of the verbs in any order. The following is a GMTG fragment for the above sentence pair[2]:

$$\begin{bmatrix}(\mathrm{S})\\(\mathrm{S})\end{bmatrix} \rightarrow \begin{bmatrix}(\mathrm{N}^1_{det}\text{ has promised }\mathrm{N}^2_{client}\hat{\mathrm{S}}^3)\\(\hat{\mathrm{S}}^3\mathrm{N}^1_{Det}\hat{\mathrm{S}}^3\mathrm{N}^2_{Klien}\hat{\mathrm{S}}^3\text{versprochen hat})\end{bmatrix} \quad (22)$$

$$\begin{bmatrix}(\hat{\mathrm{S}})\\(\hat{\mathrm{S}},\hat{\mathrm{S}},\hat{\mathrm{S}})\end{bmatrix} \rightarrow \begin{bmatrix}(\text{to indict }\mathrm{N}^1_{suspect}\mathrm{N}^2_{crime})\\(\mathrm{N}^2_{Verb},\mathrm{N}^1_{Verd},\text{ zu überführen})\end{bmatrix} \quad (23)$$

The discontinuities allow the noun arguments of *versprochen* to be placed in any order with the noun arguments of *überführen*. Rambow (1995) gives a similar analysis.

3 Formal Definitions

Let V_N be a finite set of nonterminal symbols and let $\mathbb{Z}$ be the set of integers.[3] We define $\mathcal{I}(V_N) = \{A^{(t)} \mid A \in V_N,\ t \in \mathbb{Z}\}$.[4] Elements of $\mathcal{I}(V_N)$ will be called **indexed nonterminal symbols**. In what follows we also consider a finite set of terminal symbols V_T, disjoint from V_N, and work with strings in V_I^*, where $V_I = \mathcal{I}(V_N) \cup V_T$. For $\gamma \in V_I^*$, we define $\mathsf{index}(\gamma) = \{t \mid \gamma = \gamma' A^{(t)} \gamma'',\ \gamma', \gamma'' \in V_I^*,\ A^{(t)} \in \mathcal{I}(V_N)\}$, i.e. the set of indexes that appear in γ.

An **indexed tuple vector**, or ITV, is a vector of tuples of strings over V_I, having the form

$$\overline{\gamma} = [(\gamma_{11}, \ldots, \gamma_{1q_1}), \ldots, (\gamma_{D1}, \ldots, \gamma_{Dq_D})]$$

where $D \geq 1$, $q_i \geq 0$ and $\gamma_{ij} \in V_I^*$ for $1 \leq i \leq D$, $1 \leq j \leq q_i$. We write $\overline{\gamma}[i]$, $1 \leq i \leq D$, to denote the i-th component of $\overline{\gamma}$ and $\varphi(\overline{\gamma}[i])$ to denote the arity of such a tuple, which is q_i. When $\varphi(\overline{\gamma}[i]) = 0$, $\overline{\gamma}[i]$ is the empty tuple, written $()$. This should not be confused with (ε), that is the tuple of arity one containing the empty string. A **link** is an ITV where each γ_{ij} consists of one indexed nonterminal and all of these nonterminals are coindexed. As we shall see, the notion of a link generalizes the notion of nonterminal in context-free grammars: each production rewrites a single link.

Definition 1 *Let $D \geq 1$ be some integer constant. A* **generalized multitext grammar** *with D dimensions (D-GMTG for short) is a tuple $G = (V_N, V_T, P, S)$ where V_N, V_T are finite, disjoint sets of nonterminal and terminal symbols, respectively, $S \in V_N$ is the start symbol and P is a finite set of productions. Each production has the form $\overline{\alpha} \rightarrow \overline{\beta}$, where $\overline{\alpha}$ is a D-dimensional link and $\overline{\beta}$ is a D-dimensional ITV such that $\varphi(\overline{\alpha}[i]) = \varphi(\overline{\beta}[i])$ for $1 \leq i \leq D$. If $\overline{\alpha}[i]$ contains S, then $\varphi(\overline{\alpha}[i]) = 1$.*

We omit symbol D from D-GMTG whenever it is not relevant. To simplify notation, we write productions as $\overline{p} = [p_1, \ldots, p_D]$, with each $p_i = (A_{i1}, \ldots, A_{iq_i}) \rightarrow (\alpha_{i1}, \ldots, \alpha_{iq_i})$, $A_{ij} \in V_N$. I.e. we omit the unique index appearing on the LHS of $\overline{p}$. Each p_i is called a **production component**. The production component $() \rightarrow ()$ is called the **inactive** production component. All other production components are called **active** and we set $\mathsf{active}(\overline{p}) = \{i \mid q_i > 0\}$. Inactive production components are used to relax synchronous rewriting on some dimensions, that is to implement rewriting on $d < D$ components. When $d = 1$, rewriting is licensed on one component, independently of all the others.

Two grammar parameters play an important role in this paper. Let $\overline{p} = [p_1, \ldots, p_D] \in P$ and $p_i = (A_{i1}, \ldots, A_{iq_i}) \rightarrow (\alpha_{i1}, \ldots, \alpha_{iq_i})$.

Definition 2 *The* **rank** *ρ of a production $\overline{p}$ is the number of links on its RHS: $\rho(\overline{p}) = |\mathsf{index}(\alpha_{11} \cdots \alpha_{1q_1}\alpha_{21} \cdots \alpha_{Dq_D})|$. The* **rank** *of a GMTG G is $\rho(G) = \max_{\overline{p} \in P} \rho(\overline{p})$.*

Definition 3 *The* **fan-out** *of p_i, $\overline{p}$ and G are, respectively, $\varphi(p_i) = q_i$, $\varphi(\overline{p}) = \sum_{i=1}^{D} \varphi(p_i)$ and $\varphi(G) = \max_{\overline{p} \in P} \varphi(\overline{p})$.*

For example, the rank of Production (23) is two and its fan-out is four.

In GMTG, the derives relation is defined over ITVs. GMTG derivation proceeds by synchronous application of all the active components in some production. The indexed nonterminals to be rewritten simultaneously must all have the same index t, and all nonterminals indexed with t in the ITV must be rewritten simultaneously. Some additional notation will help us to define rewriting precisely. A **reindexing** is a one-to-one function on $\mathbb{Z}$, and is extended to V_I by letting $f(a) = a$ for $a \in V_T$

[2] These are only a small subset of the necessary productions. The subscripts on the nonterminals indicate what terminals they will eventually yield; the terminal productions have been left out to save space.

[3] Any other infinite set of indexes would suit too.

[4] The parentheses around indexes distinguish them from other uses of superscripts in formal language theory. However, we shall omit the parentheses when the context is unambiguous.

and $f(A^{(t)}) = A^{(f(t))}$ for $A^{(t)} \in \mathcal{I}(V_N)$. We also extend f to strings in V_I^* analogously. We say that $\alpha, \alpha' \in V_I^*$ are **independent** if $\mathsf{index}(\alpha) \cap \mathsf{index}(\alpha') = \emptyset$.

Definition 4 *Let $G = (V_N, V_T, P, S)$ be a D-GMTG and let $\overline{p} = [p_1, \ldots, p_D]$ with $\overline{p} \in P$ and $p_i = (A_{i1}, \ldots, A_{iq_i}) \rightarrow (\alpha_{i1}, \ldots, \alpha_{iq_i})$. Let $\overline{\gamma}$ and $\overline{\delta}$ be two ITVs with $\overline{\gamma}[i] = (\gamma_{i1}, \ldots, \gamma_{iq_i})$ and $\overline{\delta}[i] = (\delta_{i1}, \ldots, \delta_{iq_i})$. Assume that α is some concatenation of all α_{ij} and that γ is some concatenation of all γ_{ij}, $1 \leq i \leq D$, $1 \leq j \leq q_i$, and let f be some reindexing such that strings $f(\alpha)$ and γ are independent. The derives relation $\overline{\gamma} \Rightarrow_G^{\overline{p}} \overline{\delta}$ holds whenever there exists an index $t \in \mathbb{Z}$ such that the following two conditions are satisfied:*

(i) *for each $i \in \mathsf{active}(\overline{p})$ we have $\gamma_{i1} \cdots \gamma_{iq_i} = \gamma'_{i0} A_{i1}^{(t)} \gamma'_{i1} A_{i2}^{(t)} \cdots \gamma'_{iq_i-1} A_{iq_i}^{(t)} \gamma'_{iq_i}$ such that $t \notin \mathsf{index}(\gamma'_{i0}\gamma'_{i1}\cdots\gamma'_{iq_i})$, and each δ_{ij} is obtained from γ_{ij} by replacing each $A_{ij'}^{(t)}$ with $f(\alpha_{ij'})$;*

(ii) *for each $i \notin \mathsf{active}(\overline{p})$ we have $t \notin \mathsf{index}(\gamma_{i1} \cdots \gamma_{iq_i})$ and $\overline{\gamma}[i] = \overline{\delta}[i]$.*

We generalize the $\Rightarrow_G^{\overline{p}}$ relation to $\Rightarrow_G$ and $\Rightarrow_G^*$ in the usual way, to represent derivations.

We can now introduce the notion of generated language (or generated relation). A **start link** of a D-GMTG is a D-dimensional link where at least one component is $(S^{(1)})$, S the start symbol, and the rest of the components are $()$. Thus, there are $2^D - 1$ start links. The **language** generated by a D-GMTG G is $L(G) = \{\overline{\gamma}_w \mid \overline{\gamma}_S \Rightarrow_G^* \overline{\gamma}_w,\ \overline{\gamma}_S$ a start link, $\overline{\gamma}_w[i] = ()or\overline{\gamma}_w[i] = (w_i)$ with $w_i \in V_T^*, 1 \leq i \leq D\}$. Each ITV in $L(G)$ is called a **multitext**. For every D-GMTG G, $L(G)$ can be partitioned into $2^D - 1$ subsets, each containing multitexts derived from a different start link. These subsets are disjoint, since every non-empty tuple of a start link is eventually rewritten as a string, either empty or not.[5]

A **start production** is a production whose LHS is a start link. A GMTG writer can choose the combinations of components in which the grammar can generate, by including start productions with the desired combinations of active components. If a grammar contains no start productions with a certain combination of active components, then the corresponding subset of $L(G)$ will be empty. Allowing a single GMTG G to generate multitexts with some empty tuples corresponds to modeling relations of different dimensionalities. This capability enables a synchronous grammar to govern lower-dimensional sublanguages/translations. For example, an English/Italian GMTG can include Production (9), an English CFG, and an Italian CFG. A single GMTG can then govern both translingual and monolingual information in applications. Furthermore, this capability simplifies the normalization procedure described in Section 6. Otherwise, this procedure would require exceptions to be made when eliminating epsilons from start productions.

[5] We are assuming that there are no useless nonterminals.

4 Generative Capacity

In this section we compare the generative capacity of GMTG with that of mildly context-sensitive grammars. We focus on LCFRS, using the notational variant introduced by Rambow and Satta (1999), briefly summarized below. Throughout this section, strings $w \in V_T^*$ and vectors of the form $[(w)]$ will be identified. For lack of space, some proofs are only sketched, or entirely omitted when relatively intuitive: Melamed et al. (2004) provide more details.

Let V_T be some terminal alphabet. A function g has rank $r \geq 0$ if it is defined on $(V_T^*)^{f_1} \times (V_T^*)^{f_2} \times \cdots \times (V_T^*)^{f_r}$, for integers $f_i \geq 1$, $1 \leq i \leq r$. Also, g has fan-out $f \geq 1$ if its range is a subset of $(V_T^*)^f$. Let y_h, x_{ij}, $1 \leq h \leq f$, $1 \leq i \leq r$ and $1 \leq j \leq f_i$, be string-valued variables. Function g is **linear regular** if it is defined by an equation of the form

$$g(\langle x_{11}, \ldots, x_{1f_1}\rangle, \ldots, \langle x_{r1}, \ldots, x_{rf_r}\rangle) = \langle y_1, \ldots, y_f\rangle \quad (24)$$

where $\langle y_1, \ldots, y_f\rangle$ represents some grouping into f strings of all and only the variables appearing in the left-hand side, possibly with some additional terminal symbols. (Symbols ρ, φ and $\Rightarrow_G$ are overloaded below.)

Definition 5 *A* **Linear Context-Free Rewriting System** *(LCFRS) is a quadruple $G = (V_N, V_T, P, S)$ where V_N, V_T and S are as in GMTGs, every $A \in V_N$ is associated with an integer $\varphi(A) \geq 1$ with $\varphi(S) = 1$, and P is a finite set of productions of the form $A \rightarrow g(B_1, B_2, \ldots, B_{\rho(g)})$, where $\rho(g) \geq 0$, $A, B_i \in V_N$, $1 \leq i \leq \rho(g)$ and where g is a linear regular function having rank $\rho(g)$ and fan-out $\varphi(A)$, defined on $(V_T^*)^{\varphi(B_1)} \times \cdots \times (V_T^*)^{\varphi(B_{\rho(g)})}$.*

For every $A \in V_N$ and $\tau \in (V_T^*)^{\varphi(A)}$, we write $A \Rightarrow_G \tau$ if

(i) *$A \rightarrow g() \in P$ and $g() = \tau$; or else*

(ii) $A \to g(B_1, \ldots, B_{\rho(g)}) \in P$, $B_i \Rightarrow_G \tau_i \in (V_T^*)^{\varphi(B_i)}$ *for every* $1 \leq i \leq \rho(g)$, *and* $g(\tau_1, \ldots, \tau_{\rho(g)}) = \tau$.

The language generated by G is defined as $L(G) = \{w \mid S \Rightarrow_G (w),\ w \in V_T^*\}$. Let $p \in P$, $p = A \to g(B_1, B_2, \ldots, B_{\rho(g)})$. The **rank** of p and G are, respectively, $\rho(p) = \rho(g)$ and $\rho(G) = \max_{p \in P} \rho(p)$. The **fan-out** of p and G are, respectively, $\varphi(p) = \varphi(A)$ and $\varphi(G) = \max_{p \in P} \varphi(p)$.

The proof of the following theorem is relatively intuitive and therefore omitted.

Theorem 1 *For any LCFRS G, there exists some 1-GMTG G' with $\rho(G') = \rho(G)$ and $\varphi(G') = \varphi(G)$ such that $L(G') = L(G)$.*

Next, we show that the generative capacity of GMTG does not exceed that of LCFRS. In order to compare string tuples with bare strings, we introduce two special functions ranging over multitexts. Assume two fresh symbols $\#, \Diamond \notin (V_T \cup V_N)$. For a multitext $\overline{\gamma}$ we write $\mathsf{pad}(\overline{\gamma}) = \overline{\gamma}'$, where $\overline{\gamma}'[i] = (\Diamond)$ if $\overline{\gamma}[i] = ()$ and $\overline{\gamma}'[i] = \overline{\gamma}[i]$ otherwise, $1 \leq i \leq D$. For a multitext $[(w_1), (w_2), \ldots, (w_D)]$ with no empty tuple, we write $\mathsf{glue}([(w_1), (w_2), \ldots, (w_D)]) = w_1 \# w_2 \# \cdots \# w_D$. We extend both functions to sets of multitexts in the obvious way: $\mathsf{glue}(L) = \{\mathsf{glue}(l) \mid l \in L\}$ and $\mathsf{pad}(L) = \{\mathsf{pad}(l) \mid l \in L\}$.

In a D-GMTG, a production with d active components, $1 \leq d \leq D$, is said to be d-**active**. A D-GMTG whose start productions are all D-active is called **properly synchronous**.

Lemma 1 *For any properly synchronous D-GMTG G, there exists some LCFRS G' with $\rho(G') = \rho(G)$ and $\varphi(G') = \varphi(G)$ such that $L(G') = \mathsf{glue}(L(G))$.*

Outline of the proof. We set $G' = (V'_N, V_T, P', [S])$, where $V'_N = \{[p, t] \mid p \in P,\ t \in \mathsf{index}(G)\} \cup \{[S]\}$, $\mathsf{index}(G)$ is the set of all indexes appearing in the productions of G, and P' is constructed as follows. Let $\overline{p}, \overline{p}' \in P$ with $\overline{p} = [p_1, \ldots, p_D]$, $\overline{p}' = [p'_1, \ldots, p'_D]$, $p_i = (A_{i1}, \ldots, A_{iD}) \to (\alpha_{i1}, \ldots, \alpha_{iq_i})$, and $p'_i = (B_{i1}, \ldots, B_{iD}) \to (\beta_{i1}, \ldots, \beta_{iq'_i})$. Assume that $\overline{p}$ can rewrite the right-hand side of $\overline{p}'$, that is

$$[(\beta_{11}, \ldots, \beta_{1q'_1}), \ldots, (\beta_{D1}, \ldots, \beta_{Dq'_D})] \Rightarrow^{\overline{p}}_G [(\delta_{11}, \ldots, \delta_{1q_1}), \ldots, (\delta_{D1}, \ldots, \delta_{Dq_D})].$$

Then there must be at least one index t such that for each $i \in \mathsf{active}(\overline{p})$, $(\beta_{i1}, \ldots, \beta_{iq'_i})$ contains exactly q_i occurrences of t.

Let $\alpha_{\overline{p}} = \alpha_{11} \cdots \alpha_{1q_1} \alpha_{21} \cdots \alpha_{Dq_D}$. Also let $\mathsf{index}(\alpha_{\overline{p}}) = \{t_1, \ldots, t_{\rho(\overline{p})}\}$ and let $\varphi(t_i)$ be the number of occurrences of t_i appearing in $\alpha_{\overline{p}}$. We define an alphabet $X_{\overline{p}} = \{x_{ij} \mid 1 \leq i \leq \rho(\overline{p}),\ 1 \leq j \leq \varphi(t_i)\}$. For each i and j with $1 \leq i \leq D$, $i \in \mathsf{active}(\overline{p})$ and $1 \leq j \leq q_i$, we define a string $h(\overline{p}, i, j)$ over $X_{\overline{p}} \cup V_T$ as follows. Let $\alpha_{ij} = Y_1 Y_2 \cdots Y_c$, each $Y_k \in V_I$. Then $h(\overline{p}, i, j) = Y'_1 Y'_2 \cdots Y'_c$, where

- $Y'_k = Y_k$ in case $Y_k \in V_T$; and
- $Y'_k = x_{t,m}$ in case $Y_k \in \mathcal{I}(V_N)$, where t is the index of Y_k and the indicated occurrence of Y_k is the m-th occurrence of such symbol appearing from left to right in string $\alpha_{\overline{p}}$.

Next, for every possible $\overline{p}$, $\overline{p}'$, and t as above, we add to P' a production

$$p_t = [\overline{p}', t] \to g([\overline{p}, t_1], \ldots, [\overline{p}, t_{\rho(\overline{p})}]),$$

where

$$g(\langle x_{11}, \ldots, x_{1\varphi(t_1)} \rangle, \ldots, \langle x_{\rho(\overline{p})1}, \ldots, x_{\rho(\overline{p})\varphi(t_{\rho(\overline{p})})} \rangle) = \langle h(\overline{p}, 1, 1), \ldots, h(\overline{p}, D, q_D) \rangle$$

(each $h(\overline{p}, i, j)$ above satisfies $i \in \mathsf{active}(\overline{p})$). Note that g is a function with rank $\rho(\overline{p})$ and fan-out $\sum_{i=1}^{D} q_i = \varphi(\overline{p})$. Thus we have $\rho(p_t) = \rho(\overline{p})$ and $\varphi(p_t) = \varphi(\overline{p})$. Without loss of generality, we assume that G contains only one production with S appearing on the left-hand side, having the form $\overline{p}_S = [(S), \ldots, (S)] \to [(A^1), \ldots, (A^1)]$. To complete the construction of P', we then add a last production $[S] \to g([\overline{p}_S, 1])$ where $g(\langle x_{11}, x_{12}, \ldots, x_{1D} \rangle) = \langle x_{11} \# x_{12} \# \cdots \# x_{1D} \rangle$. We claim that, for each $\overline{p}$, $\overline{p}'$ and t as above

$$[(A^1_{11}, \ldots, A^1_{1q_1}), \ldots, (A^1_{D1}, \ldots, A^1_{Dq_D})] \Rightarrow^*_G [(u_{11}, \ldots, u_{1q_1}), \ldots, (u_{D1}, \ldots, u_{Dq_D})]$$

iff $[\overline{p}', t] \Rightarrow_{G'} \langle u_{11}, \ldots, u_{1q_1}, u_{21}, \ldots, w_{Dq_D} \rangle$. The lemma follows from this claim. ∎

The proof of the next lemma is relatively intuitive and therefore omitted.

Lemma 2 *For any D-GMTG G, there exists a properly synchronous D-GMTG G' such that $\rho(G') = \rho(G)$, $\varphi(G') = \max\{\varphi(G), D\}$, and $L(G') = \mathsf{pad}(L(G))$.*

Combining Lemmas 1 and 2, we have

Theorem 2 *For any D-GMTG G, there exists some LCFRS G' with $\rho(G') = \rho(G)$ and $\varphi(G') = \max\{\varphi(G), D\}$ such that $L(G') = \mathsf{glue}(\mathsf{pad}(L(G)))$.*

5 Weak Language Preservation Property

GMTGs have the weak language preservation property, which is one of the defining requirements of synchronous rewriting systems (Rambow and Satta, 1996). Informally stated, the generative capacity of the class of all component grammars of a GMTG exactly corresponds to the class of all projected languages. In other words, the interaction among different grammar components in the rewriting process of GMTG does not increase the generative power beyond the above mentioned class. The next result states this property more formally.

Let G be a D-GMTG with production set P. For $1 \leq i \leq D$, the i-th **component grammar** of G, written $\mathsf{proj}(G, i)$, is the 1-GMTG with productions $P_i = \{p_i \mid [p_1, \ldots, p_D] \in P,\ p_i \neq () \rightarrow ()\}$. Similarly, the i-th **projected language** of $L(G)$ is $\mathsf{proj}(L(G), i) = \{w_i \mid [(w_1), \ldots, (w_D)] \in L(G),\ (w_i) \neq ()\}$. In general $L(\mathsf{proj}(G, i)) \neq \mathsf{proj}(L(G), i)$, because component grammars $\mathsf{proj}(G, i)$ interact with each other in the rewriting process of G. To give a simple example, consider the 2-GMTG G with productions $[(S), (S)] \rightarrow [(\varepsilon), (\varepsilon)]$, $[(S), (S)] \rightarrow [(aA^{(1)}), (aS^{(1)})]$ and $[(A), (S)] \rightarrow [(S^{(1)}), (S^{(1)}b)]$. Then $L(G) = \{[(a^n), (a^n b^n)] \mid n \geq 0\}$, and thus $\mathsf{proj}(L(G), 2) = \{a^n b^n \mid n \geq 0\}$. On the other hand, $L(\mathsf{proj}(G, 2)) = \{a^n b^m \mid n, m \geq 0\}$. Let $\mathcal{L}(\text{LCFRS})$ be the class of all languages generated by LCFRSs. Also let $\mathcal{L}_{p(G)}$ and $\mathcal{L}_{p(L)}$ be the classes of languages $L(\mathsf{proj}(G, d))$ and $\mathsf{proj}(L(G), d))$, respectively, for every $D \geq 1$, every D-GMTG G and every d with $1 \leq d \leq D$.

Theorem 3 *$\mathcal{L}_{p(G)} = \mathcal{L}(LCFRS)$ and $\mathcal{L}_{p(L)} = \mathcal{L}(LCFRS)$.*

Proof. The $\supseteq$ cases directly follow from Theorem 1.

Let G be some D-GMTG and let d be an integer such that $1 \leq d \leq D$. It is not difficult to see that $\mathsf{glue}(\mathsf{pad}(L(\mathsf{proj}(G, d)))) = L(\mathsf{proj}(G, d))$. Hence $L(\mathsf{proj}(G, d))$ can be generated by some LCFRS, by Theorem 2.

We now define a LCFRS G' such that $L(G') = \mathsf{proj}(\mathsf{pad}(L(G)), d))$. Assume $G'' = (V_N, V_T, P, S)$ is a properly synchronous D-GMTG generating $\mathsf{pad}(L(G))$ (Lemma 2). Let $G' = (V'_N, V_T, P', [S])$, where V'_N and P' are constructed from G'' almost as in the proof of Lemma 1. The only difference is in the definition of strings $h(\overline{p}, i, j)$ and the production rewriting $[S]$, specified as follows (we use the same notation as in the proof of Lemma 1). $h(\overline{p}, i, j) = Y'_1 Y'_2 \cdots Y'_c$, where for each k: (i) $Y'_k = Y_k$ if $Y_k \in V_T$ and $i = d$; (ii) $Y'_k = \varepsilon$ if $Y_k \in V_T$ and $i \neq d$; (iii) $Y'_k = x_{t,m}$ if $Y_k \in \mathcal{I}(V_N)$, with t, m as in the original proof. Finally, the production rewriting $[S]$ has the form $[S] \rightarrow g([\overline{p}_S, 1])$, where $g(\langle x_{11}, x_{12}, \ldots, x_{1D}\rangle) = \langle x_{11} x_{12} \cdots x_{1D}\rangle$. To conclude the proof, note that $\mathsf{proj}(L(G), d))$ and $\mathsf{proj}(\mathsf{pad}(L(G)), d))$ can differ only with respect to string $\Diamond$. The theorem then follows from the fact that LCFRS is closed under intersection with regular languages (Weir, 1988). ∎

6 Generalized Chomsky Normal Form

Certain kinds of text analysis require a grammar in a convenient normal form. The prototypical example for CFG is Chomsky Normal Form (CNF), which is required for CKY-style parsing. A D-GMTG is in **Generalized Chomsky Normal Form** (GCNF) if it has no useless links or useless terminals, and every production is in one of two forms:

(i) A **nonterminal production** has rank = 2 and no terminals or ε's on the RHS.

(ii) A **terminal production** has exactly one component of the form $A \rightarrow a$, where $A \in V_N$ and $a \in V_T$. The other components are inactive.

The algorithm to convert a GMTG to GCNF has the following steps: (1) add a new start-symbol (2) isolate terminals, (3) binarize productions, (4) remove ε's, (5) eliminate useless links and terminals, and (6) eliminate unit productions. The steps are generalizations of those presented by Hopcroft et al. (2001) to the multidimensional case with discontinuities. The ordering of these steps is important, as some steps can restore conditions that others eliminate. Traditionally, the terminal isolation and binarization steps came last, but the alternative order reduces the number of productions that can be created during ε-elimination. Steps (1), (2), (5) and (6) are the same for CFG and GMTG, except that the notion of nonterminal in CFG is replaced with links in GMTG. Some complications arise, however, in the generalization of steps (3) and (4).

6.1 Step 3: Binarize

The third step of converting to GCNF is binarization of the productions, making the rank of the grammar two. For $r \geq 0$ and $f \geq 1$, we write D-GMTG$_f^{(r)}$ to represent the class of all D-GMTGs with rank r and fan-out f. A CFG can always be binarized into another CFG: two adjacent nonterminals are replaced with a single nonterminal that yields them. In contrast, it can be impossible to binarize a D-GMTG$_f^{(r)}$ into an equivalent D-GMTG$_f^2$. From results presented by Rambow and Satta (1999) it follows that,

$$\begin{bmatrix}(\text{S})\\(\text{S})\end{bmatrix} \rightarrow \begin{bmatrix}(\text{N}^1_{\text{Pat}}\text{V}^2_{\text{went}}\text{P}^3_{\text{home}}\text{A}^4_{\text{early}})\\(\text{P}^3_{\text{damoy}}\text{N}^1_{\text{Pat}}\text{A}^4_{\text{rano}}\text{V}^2_{\text{pashol}})\end{bmatrix}$$

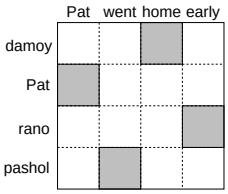

Figure 1: A production that requires an increased fan-out to binarize, and its 2D illustration.

for every fan-out $f \geq 2$ and rank $r \geq 4$, there are some index orderings that can be generated by D-GMTG$_f^{(r)}$ but not D-GMTG$_f^{(r-1)}$. The distinguishing characteristic of such index orderings is apparent in Figure 1, which shows a production in a grammar with fan-out two, and a graph that illustrates which nonterminals are coindexed. No two nonterminals are adjacent in both components, so replacing any two nonterminals with a single nonterminal causes a discontinuity. Increasing the fan-out of the grammar allows a single nonterminal to rewrite as non-adjacent nonterminals in the same string. Increasing the fan-out can be necessary even for binarizing a 1-GMTG production such as:

$$[(\text{S,S})] \rightarrow [(\text{N}^1\text{V}^2\text{P}^3\text{A}^4, \text{P}^3\text{N}^1\text{A}^4\text{V}^2)] \qquad (25)$$

To binarize, we nondeterministically split each nonterminal production p_0 of rank $r > 2$ into two nonterminal productions p_1 and p_2 of rank $< r$, but possibly with higher fan-out. Since this algorithm replaces r with two productions that have rank $< r$, recursively applying the algorithm to productions of rank greater than two will reduce the rank of the grammar to two. The algorithm follows:

(i) Nondeterministically chose n links to be removed from p_0 and replaced with a single link to make p_1, where $2 \leq n \leq r-1$. We call these links the **m-links**.

(ii) Create a new ITV $\overline{\gamma}$. Two nonterminals are **neighbors** if they are adjacent in the same string in a production RHS. For each set of m-link neighbors in component d in p_0, place that set of neighbors into the d'th component of $\overline{\gamma}$ in the order in which they appeared in p_0, so that each set of neighbors becomes a different string, for $1 \leq d \leq D$.

(iii) Create a new unique nonterminal, say B, and replace each set of neighbors in production p_0 with B, to create p_1. The production p_2 is $[B, \ldots, B] \rightarrow \overline{\gamma}$

For example, binarization of the productions for the English/Russian multitext [(*Pat went home early*), (*damoy Pat rano pashol*)][6] in Figure 1 requires that we increase the fan-out of the language to three. The binarized productions are as follows:

$$\begin{bmatrix}(\text{S})\\(\text{S})\end{bmatrix} \rightarrow \begin{bmatrix}(\text{N}^1_{\text{Pat}}\text{VP}^2)\\(\text{VP}^2\text{N}^1_{\text{Pat}}\text{VP}^2)\end{bmatrix} \qquad (26)$$

$$\begin{bmatrix}(\text{VP})\\(\text{VP, VP})\end{bmatrix} \rightarrow \begin{bmatrix}(\text{V}^1\text{A}^2_{\text{early}})\\(\text{V}^1, \text{A}^2_{\text{rano}}\text{V}^1)\end{bmatrix} \qquad (27)$$

$$\begin{bmatrix}(\text{V})\\(\text{V, V})\end{bmatrix} \rightarrow \begin{bmatrix}(\text{V}^1_{\text{went}}\text{P}^2_{\text{home}})\\(\text{P}^2_{\text{damoy}}, \text{V}^1_{\text{pashol}})\end{bmatrix} \qquad (28)$$

6.2 Step 4: Eliminate ε's

Grammars in GCNF cannot have ε's in their productions. Thus, GCNF is a more restrictive normal form than those used by Wu (1997) and Melamed (2003). The absence of ε's simplifies parsers for GMTG (Melamed, 2004). Given a GMTG G with ε in some productions, we give the construction of a weakly equivalent grammar G' without any ε's. First, determine all nullable links and associated strings in G. A link $\overline{\lambda} = [(A_1, \ldots, A_1), \ldots, (A_D, \ldots, A_D)]$ is **nullable** if $\overline{\lambda} \stackrel{*}{\Rightarrow} \overline{\gamma}$, where $\overline{\gamma} = [(\alpha_{11}, \ldots, \alpha_{1q_1}), \ldots, (\alpha_{D1}, \ldots, \alpha_{Dq_D})]$ is an ITV where at least one α_{ij} is ε. We say the link $\overline{\lambda}$ is nullable and the string at address (d, q) in $\overline{\lambda}$ is nullable. For each nullable link, we create 2^n versions of the link, where n is the number of nullable strings of that link. There is one version for each of the possible combinations of the nullable strings being present or absent. The version of the link with all strings present is its original version. Each non-original version of the link (except in the case of start links) gets a unique subscript, which is applied to all the nonterminals in the link, so that each link is unique in the grammar. We construct a new grammar G' whose set of productions P' is determined as follows: for each production, we identify the nullable links on the RHS and replace them with each combination of the non-original versions found earlier. If a string is left empty during this process, that string is removed from the RHS and the fan-out of the production component is reduced by one. The link on the LHS is replaced with its appropriate matching non-original link. There is one exception to the replacements. If a production consists of all nullable strings, do not include this case. Lastly, we remove all strings on the RHS of productions that have ε's, and reduce the fan-out of the productions accordingly. Once

[6] The Russian is topicalized but grammatically correct.

again, we replace the LHS link with the appropriate version.

Consider the example grammar:

$$[(S),(S)] \rightarrow [(A^1B^2A^1),(B^2A^1)] \quad (29)$$
$$[(A,A),(A)] \rightarrow [(a,B^1),(B^1)] \quad (30)$$
$$[(B),(B)] \rightarrow [(b),(\varepsilon)] \quad (31)$$
$$[(B),(B)] \rightarrow [(b),(bb)] \quad (32)$$

We first identify which links are nullable. In this case $[(A,A),(A)]$ and $[(B),(B)]$ are nullable so we create a new version of both links: $[(A_1,A_1),()]$ and $[(B_1),()]$. We then alter the productions. Production (31) gets replaced by (40). A new production based on (30) is Production (38). Lastly, Production (29) has two nullable strings on the RHS, so it gets altered to add three new productions, (34), (35) and (36). The altered set of productions are the following:

$$[(S),(S)] \rightarrow [(A^1B^2A^1),(B^2A^1)] \quad (33)$$
$$[(S),(S)] \rightarrow [(A^1B_1^2A^1),(A^1)] \quad (34)$$
$$[(S),(S)] \rightarrow [(A_1^1B^2A_1^1),(B^2)] \quad (35)$$
$$[(S),()] \rightarrow [(A_1^1B_1^2A_1^1),()] \quad (36)$$
$$(A,A),(A)] \rightarrow [(a,B^1),(B^1) \quad (37)$$
$$[(A_1,A_1),()] \rightarrow [(a,B_1^1),()] \quad (38)$$
$$[(B),(B)] \rightarrow [(b),(bb)] \quad (39)$$
$$[(B_1),()] \rightarrow [(b),()] \quad (40)$$

Melamed et al. (2004) give more details about conversion to GCNF, as well as the full proof of our final theorem:

Theorem 4 *For each GMTG G there exists a GMTG G' in GCNF generating the same set of multitexts as G but with each (ε) component in a multitext replaced by $()$.*

7 Conclusions

Generalized Multitext Grammar is a convenient and intuitive model of parallel text. In this paper, we have presented some formal properties of GMTG, including proofs that the generative capacity of GMTG is comparable to ordinary LCFRS, and that GMTG has the weak language preservation property. We also proposed a synchronous generalization of Chomsky Normal Form, laying the foundation for synchronous CKY parsing under GMTG. In future work, we shall explore the empirical properties of GMTG, by inducing stochastic GMTGs from real multitexts.

Acknowledgments

Thanks to Owen Rambow and the anonymous reviewers for valuable feedback. This research was supported by an NSF CAREER Award, the DARPA TIDES program, the Italian MIUR under project PRIN No. 2003091149_005, and an equipment gift from Sun Microsystems.

References

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

T. Becker, A. Joshi, and O. Rambow. 1991. Long-distance scrambling and tree adjoining grammars. In *Proceedings of the 5th Meeting of the European Chapter of the Association for Computational Linguistics (EACL)*, Berlin, Germany.

E. Bertsch and M. J. Nederhof. 2001. On the complexity of some extensions of RCG parsing. In *Proceedings of the 7th International Workshop on Parsing Technologies (IWPT)*, pages 66–77, Beijing, China.

M. Dras and T. Bleam. 2000. How problematic are clitics for S-TAG translation? In *Proceedings of the 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, Paris, France.

J. Hopcroft, R. Motwani, and J. Ullman. 2001. *Introduction to Automota Theory, Languages and Computation*. Addison-Wesley, USA.

I. Dan Melamed, G. Satta, and B. Wellington. 2004. Generalized multitext grammars. Technical Report 04-003, NYU Proteus Project. `http://nlp.cs.nyu.edu/pubs/`.

I. Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 158–165, Edmonton, Canada.

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain.

O. Rambow and G. Satta. 1996. Synchronous models of language. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, Santa Cruz, USA.

O. Rambow and G. Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223:87–120, July.

O. Rambow. 1995. *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

S. Shieber. 1994. Restricting the weak-generative capactiy of synchronous tree-adjoining grammars. *Computational Intelligence*, 10(4):371–386.

D. J. Weir. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

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

D. H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, February.