Lecture 5

Mildly Context-Sensitive Languages

Last modified 2016/07/08

Multiple context-free grammars

In the previous lecture, we proved the equivalence between TAG and LIG. In fact, there is yet another grammar formalism, namely the *head grammar*, that is equivalent to these two formalisms. A head grammar is a special kind of *multiple context-free grammar* (MCFG), so we introduce the latter formalism first.

The MCFG is a natural generalization of the "bottom-up" view of the CFG. The standard, "top-down" view of the CFG takes a rule

$$A \to X_1 \dots X_n$$

as a permission to rewrite A into $X_1 \dots X_n$. In contrast, the bottom-up view of the CFG interprets the same rule not as a rewriting instruction, but as an implication, which says:

$$A \Rightarrow^* x_1 \dots x_n \quad \text{if} \quad X_1 \Rightarrow^* x_1 \wedge \dots \wedge X_n \Rightarrow^* x_n.$$

In fact, to define the language $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$ of a CFG G, there is no need to define the derivation relation $\Rightarrow^*$ which holds between strings of terminals and nonterminals. All you need is an inductive definition of the subrelation of this relation that holds between single nonterminals and strings of terminals:

$$\Rightarrow^* \cap (N \times \Sigma^*).$$

To express that nonterminal A and terminal string x stand in this relation (i.e., $A \Rightarrow^* x$), we may write $A(x)$, treating nonterminal A as a unary predicate on

terminal strings. Then the bottom-up interpretation of the CFG rule can be written in the form of a *Horn clause*:

$$A(x_1 \dots x_n) \leftarrow X_1(x_1), \dots, X_n(x_n).$$

A context-free grammar now becomes a Horn clause program consisting of rules of the above form.

A *multiple context-free grammar* is a generalization of this Horn clause presentation of a context-free grammar. While in a CFG, all nonterminals are unary predicates, in an MCFG each nonterminal may have arbitrary fixed *arity*; that is to say, a nonterminal is an r-ary relation on terminal strings for some $r \in \mathbb{N}$ (usually $r \geq 1$). (A top-down view of MCFG rules as rewriting instructions is possible (Rambow and Satta 1999), but is more awkward.)

We now proceed to give a formal definition of an MCFG.

A *ranked alphabet* is a finite set $F = \bigcup_{n \in \mathbb{N}} F^{(n)}$, where $F^{(m)} \cap F^{(n)} = \varnothing$ if $m \neq n$. If $f \in F^{(n)}$, n is the *rank* or *arity* of f.

A *multiple context-free grammar* (Seki et al. 1991) is a 4-tuple $G = (N, \Sigma, P, S)$, where

1. N is a ranked alphabet of *nonterminals*,
2. Σ is an (unranked) alphabet of *terminals*, disjoint from N,
3. P is a finite set of *rules* of the form[1]

   $$B(\alpha_1, \dots, \alpha_r) \leftarrow B_1(\boldsymbol{x}_{1,1}, \dots, \boldsymbol{x}_{1,r_1}), \dots, B_n(\boldsymbol{x}_{n,1}, \dots, \boldsymbol{x}_{n,r_n}),$$

   where

   (a) $n \geq 0$,
   (b) $B \in N^{(r)}, B_1 \in N^{(r_1)}, \dots, B_n \in N^{(r_n)}$,
   (c) $\boldsymbol{x}_{i,j}$ are pairwise distinct variables,
   (d) $\alpha_1, \dots, \alpha_r$ are strings over $\Sigma \cup \{ \boldsymbol{x}_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq r_i \}$,
   (e) each $\boldsymbol{x}_{i,j}$ occurs at most once in the string $\alpha_1 \dots \alpha_r$, and
4. $S \in N^{(1)}$.

We say that G is an m-MCFG if the arities of nonterminals do not exceed m.

The language of G is $L(G) = \{ w \in \Sigma^* \mid \vdash_G S(w) \}$, where $\vdash_G$ is inductively defined as follows:

[1] A rule of an MCFG was originally written in the form $B \to f[B_1, \dots, B_n]$, where f is the name of a function that takes tuples of strings and returns a tuple of strings, whose definition was given separately. I find the Horn clause representation (first used by Groenink (1997a,b) in connection with MCFGs, as far as I am aware) much easier to understand.

- if $B(\alpha_1, \ldots, \alpha_r) \leftarrow B_1(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{1,r_1}), \ldots, B_n(\boldsymbol{x}_{n,1}, \ldots, \boldsymbol{x}_{n,r_n})$ is a rule in P and $\vdash_G B_i(w_{i,1}, \ldots, w_{i,r_i})$ for $i = 1, \ldots, n$, then $\vdash_G B(\alpha_1, \ldots, \alpha_r)\sigma$, where σ is the substitution that sends $\boldsymbol{x}_{i,j}$ to $w_{i,j}$.[2]

If G is an m-MCFG, $L(G)$ is called an *m-MCFL*.

An (m-)MCFG is *non-deleting* if each rule satisfies the strengthened form of condition (e):

(e′) each $\boldsymbol{x}_{i,j}$ occurs exactly once in $\alpha_1 \ldots \alpha_r$.

Non-deleting (m-)MCFGs are also known as *(string-based) (m-)linear context-free rewriting systems* (LCFRSs) (Vijay-Shanker et al. 1987, Weir 1988). It is known that for every m-MCFG G, there is an m-LCFRS G' such that $L(G) = L(G')$ (Seki et al. 1991).[3]

In a rule

$$B(\alpha_1, \ldots, \alpha_r) \leftarrow B_1(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{1,r_1}), \ldots, B_n(\boldsymbol{x}_{n,1}, \ldots, \boldsymbol{x}_{n,r_n}),$$

$B(\alpha_1, \ldots, \alpha_r)$ is called the *head* of the rule, and $B_1(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{1,r_1}), \ldots, B_n(\boldsymbol{x}_{n,1}, \ldots, \boldsymbol{x}_{n,r_n})$ is called the *body*. Each $B_i(\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,r_i})$ is called a *subgoal*. A rule whose body is empty is called a *terminating rule*.

The class of MCFLs properly includes the class of TALs (LILs). The following 3-MCFG $G = (N, \Sigma, P, S)$ generates COUNT-5 = $\{ \mathsf{a}^n\mathsf{b}^n\mathsf{c}^n\mathsf{d}^n\mathsf{e}^n \mid n \geq 0 \}$:

$$\begin{aligned} &G = (N, \Sigma, P, S), \\ &N = \{S, A\}, \quad \text{where } N^{(1)} = \{S\},\ N^{(3)} = \{A\}, \\ &\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}\}, \end{aligned}$$

[2] We use the postfix notation for the result of applying a substitution to an expression.

[3] It is often said that LCFRSs are "equivalent to" or even "weakly equivalent to" MCFGs. I find such a statement misleading since LCFRSs are simply MCFGs with a certain restriction. A better formulation would be "constitute a normal form of". (This point was correctly recognized by Joshi et al. (1991), who wrote "Kasami, Seki, and Fujii (1988) have studied a system called multiple context-free grammars, which is the same as LCFRSs.") They certainly do not warrant completely unrelated names. Original work on LCFRSs and MCFGs was carried out independently and was published roughly at the same time, and neither side seems willing to give up their respective nomenclature. I prefer "multiple context-free grammar" because (i) the modifier "multiple" is useful to designate other formalisms, e.g., "multiple regular tree grammar" (Engelfriet 1997), (ii) it was Seki et al. (1991) who first published a journal article on the subject proving some important properties of the formalism, (iii) "string-based linear context-free rewriting system" is actually a special instance of a more general notion of linear context-free rewriting system, but the latter was never precisely defined, and (iv) it is best to interpret MCFG/LCFRS in a bottom-up fashion, so "rewriting system" is not a good description of what they are.

$$
\begin{aligned}
P = \{&S(\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_3) \leftarrow A(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3),\\
&A(\varepsilon, \varepsilon, \varepsilon) \leftarrow,\\
&A(\mathsf{a}\boldsymbol{x}_1\mathsf{b}, \mathsf{c}\boldsymbol{x}_2\mathsf{d}, \mathsf{e}\boldsymbol{x}_3) \leftarrow A(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)\}
\end{aligned}
$$

This grammar is non-deleting, so it is a 3-LCFRS.

The following normal form for MCFGs is convenient. We call a rule $B(\alpha_1, \ldots, \alpha_r) \leftarrow B_1(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{1,r_1}), \ldots, B_n(\boldsymbol{x}_{n,1}, \ldots, \boldsymbol{x}_{n,r_n})$ of an MCFG G *non-permuting* if it satisfies the following condition:

- there are no i, j, k and β, γ, δ such that $1 \leq i \leq n$, $1 \leq j < k \leq r_i$, and $\alpha_1 \ldots \alpha_r = \beta\boldsymbol{x}_{i,k}\gamma\boldsymbol{x}_{i,j}\delta$.

A *non-permuting* MCFG is one whose rules are all non-permuting. Non-deleting and non-permuting MCFGs correspond to what Villemonte de la Clergerie (2002a,b) called *ordered simple RCG* and Kracht (2003) called *monotone* LCFRSs.

Theorem 5.1 (Michaelis 2001b, Kracht 2003)**.** *For every m-MCFG G, there is a non-deleting and non-permuting m-MCFG G' such that $L(G) = L(G')$.*

In what follows, we will only consider MCFGs that are non-deleting and non-permuting.

The following (non-deleting non-permuting) 2-MCFG generates RESP $= \{\, \mathsf{a}_1^m\mathsf{a}_2^m\mathsf{b}_1^n\mathsf{b}_2^n\mathsf{a}_3^m\mathsf{a}_4^m\mathsf{b}_3^n\mathsf{b}_4^n \mid m, n \geq 0 \,\}$:

$$
\begin{aligned}
&S(\boldsymbol{x}_1\boldsymbol{y}_1\boldsymbol{x}_2\boldsymbol{y}_2) \leftarrow P(\boldsymbol{x}_1, \boldsymbol{x}_2), Q(\boldsymbol{y}_1, \boldsymbol{y}_2)\\
&P(\varepsilon, \varepsilon) \leftarrow\\
&P(\mathsf{a}_1\boldsymbol{x}_1\mathsf{a}_2, \mathsf{a}_3\boldsymbol{x}_2\mathsf{a}_4) \leftarrow P(\boldsymbol{x}_1, \boldsymbol{x}_2)\\
&Q(\varepsilon, \varepsilon) \leftarrow\\
&Q(\mathsf{b}_1\boldsymbol{y}_1\mathsf{b}_2, \mathsf{b}_3\boldsymbol{y}_2\mathsf{b}_4) \leftarrow Q(\boldsymbol{y}_1, \boldsymbol{y}_2)
\end{aligned}
$$

We call a rule $B(\alpha_1, \ldots, \alpha_r) \leftarrow B_1(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{1,r_1}), \ldots, B_n(\boldsymbol{x}_{n,1}, \ldots, \boldsymbol{x}_{n,r_n})$ *well-nested* if it is non-deleting and non-permuting and satisfies the following condition:[4]

- there are no i, j, k, i', j', k' and $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ such that $1 \leq i, i' \leq n$, $i \neq i'$, $1 \leq j < k \leq r_i$, $1 \leq j' < k' \leq r_{i'}$, and $\alpha_1 \ldots \alpha_r = \beta_1\boldsymbol{x}_{i,j}\beta_2\boldsymbol{x}_{i',j'}\beta_3\boldsymbol{x}_{i,k}\beta_4\boldsymbol{x}_{i',k'}\beta_5$.

[4]It is important to require well-nested rules to be non-permuting. For instance, the first rule for the above grammar for RESP may be replaced by the following rules:

$$
\begin{aligned}
S(\boldsymbol{x}_1\boldsymbol{x}_3\boldsymbol{x}_2\boldsymbol{x}_4) &\leftarrow S'(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4),\\
S'(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{y}_1, \boldsymbol{y}_2) &\leftarrow P(\boldsymbol{x}_1, \boldsymbol{x}_2), Q(\boldsymbol{y}_1, \boldsymbol{y}_2).
\end{aligned}
$$

The second rule is well-nested, but the first rule is not, since it is permuting.

We say that an MCFG G is *well-nested* if every rule of G is well-nested. Note that the above grammar generating RESP is not well-nested.

Exercise 5.1. Let $(_1,)_1, \ldots, (_n,)_n$ be n pairs of parentheses. Let σ be the substitution:

$$\sigma(\boldsymbol{x}_{i,j}) = \begin{cases} (_i)_i & \text{if } j = 1 = r_i, \\ (_i & \text{if } j = 1 < r_i, \\ )_i(_i & \text{if } 1 < j < r_i, \\ )_i & \text{if } 1 < j = r_i. \end{cases}$$

Show that a rule $B(\alpha_1, \ldots, \alpha_r) \leftarrow B_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, B_n(x_{n,1}, \ldots, x_{n,r_n})$ is well-nested if and only if $(\alpha_1 \ldots \alpha_r)\sigma$ is well-parenthesized.

Well-nested (m-)MCFGs are equivalent to *coupled-context-free grammars* (of rank m) (Hotz and Pitsch 1996).[5]

Head grammars

Head grammars were introduced by Pollard (1984) and their formal properties were studied by Roach (1987). We present (a slightly modified version of) an equivalent formulation suggested by Roach (1987) at the end of his paper, often called *modified head grammars*.[6] In this formulation, a head grammar is a special type of 2-MCFG.

A head grammar is a 2-MCFG $G = (N, \Sigma, P, S)$, where

1. every nonterminal in N except S has arity 2, and
2. every rule in P has one of the following forms:

$$\begin{aligned} &S(\boldsymbol{x}_1\boldsymbol{x}_2) \leftarrow A(\boldsymbol{x}_1, \boldsymbol{x}_2), \\ &A(\boldsymbol{x}_{1,1}\boldsymbol{x}_{1,2} \ldots \boldsymbol{x}_{i-1,1}\boldsymbol{x}_{i-1,2}\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}\boldsymbol{x}_{i+1,1}\boldsymbol{x}_{i+1,2} \ldots \boldsymbol{x}_{l,1}\boldsymbol{x}_{l,2}) \\ &\qquad\qquad \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, B_l(\boldsymbol{x}_{l,1}, \boldsymbol{x}_{l,2}), \\ &A(\boldsymbol{x}_1\boldsymbol{y}_1, \boldsymbol{y}_2\boldsymbol{x}_2) \leftarrow B_1(\boldsymbol{x}_1, \boldsymbol{x}_2), B_2(\boldsymbol{y}_1, \boldsymbol{y}_2), \\ &A(w_1, w_2) \leftarrow, \end{aligned}$$

where $A, B_j \in N - \{S\}$.

[5] Hotz and Pitsch (1996) take a top-down view of rules, which is more natural in the special case of well-nested rules than in the case of general MCFG rules.

[6] The term "modified head grammar" is due to Vijayashanker (1987). Roach (1987) himself referred to this formulation as "center grammar".

If $L = L(G)$ for some HG G, then L is called a *head language* (HL).

A head grammar $G = (N, \Sigma, P, S)$ generating COPY $= \{\, ww \mid w \in \{\mathsf{a}, \mathsf{b}\}^* \,\}$ may be defined as follows:

$$
\begin{aligned}
&\Sigma = \{\mathsf{a}, \mathsf{b}\},\\
&N = \{S, S', C, D, A_1, A_2, B_1, B_2\},\\
&P = \left\{ \begin{array}{l}
S(\boldsymbol{x}_1\boldsymbol{x}_2) \leftarrow S'(\boldsymbol{x}_1, \boldsymbol{x}_2),\\
C(\boldsymbol{x}_1\boldsymbol{y}_1, \boldsymbol{y}_2\boldsymbol{x}_2) \leftarrow S'(\boldsymbol{x}_1, \boldsymbol{x}_2), A_1(\boldsymbol{y}_1, \boldsymbol{y}_2),\\
S'(\boldsymbol{x}_1\boldsymbol{y}_1, \boldsymbol{y}_2\boldsymbol{x}_2) \leftarrow A_2(\boldsymbol{x}_1, \boldsymbol{x}_2), C(\boldsymbol{y}_1, \boldsymbol{y}_2),\\
D(\boldsymbol{x}_1\boldsymbol{y}_1, \boldsymbol{y}_2\boldsymbol{x}_2) \leftarrow S'(\boldsymbol{x}_1, \boldsymbol{x}_2), B_1(\boldsymbol{y}_1, \boldsymbol{y}_2),\\
S'(\boldsymbol{x}_1\boldsymbol{y}_1, \boldsymbol{y}_2\boldsymbol{x}_2) \leftarrow B_2(\boldsymbol{x}_1, \boldsymbol{x}_2), D(\boldsymbol{y}_1, \boldsymbol{y}_2),\\
S'(\varepsilon, \varepsilon) \leftarrow,\\
A_1(\mathsf{a}, \varepsilon) \leftarrow,\\
A_2(\varepsilon, \mathsf{a}) \leftarrow,\\
B_1(\mathsf{b}, \varepsilon) \leftarrow,\\
B_2(\varepsilon, \mathsf{b}) \leftarrow
\end{array} \right\}.
\end{aligned}
$$

Roach (1987) gives a complicated pumping lemma for head grammars. Weir (1988) conjectured that RESP is not an HL. This conjecture was later proved by Seki et al. (1991).

Theorem 5.2. *For every well-nested 2-MCFG G, there is a head grammar $G' = (N', \Sigma', P', S')$ such that $L(G) = L(G')$ and every rule in P' has one of the following forms:*

$$
\begin{aligned}
&S'(\boldsymbol{x}_1\boldsymbol{x}_2) \leftarrow A(\boldsymbol{x}_1, \boldsymbol{x}_2),\\
&A(\boldsymbol{x}_1, \boldsymbol{x}_2\boldsymbol{y}_1\boldsymbol{y}_2) \leftarrow B(\boldsymbol{x}_1, \boldsymbol{x}_2), C(\boldsymbol{y}_1, \boldsymbol{y}_2),\\
&A(\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{y}_1, \boldsymbol{y}_2) \leftarrow B(\boldsymbol{x}_1, \boldsymbol{x}_2), C(\boldsymbol{y}_1, \boldsymbol{y}_2),\\
&A(\boldsymbol{x}_1\boldsymbol{y}_1, \boldsymbol{y}_2\boldsymbol{x}_2) \leftarrow B(\boldsymbol{x}_1, \boldsymbol{x}_2), C(\boldsymbol{y}_1, \boldsymbol{y}_2),\\
&A(w_1, w_2) \leftarrow,
\end{aligned}
$$

where $A, B, C \in N' - \{S'\}$.

Proof. Let $G = (N, \Sigma, P, S)$ be a well-nested 2-MCFG. First, we remove occurrences of terminals from all non-terminating rules, as follows. For every $c \in \Sigma$, we introduce a new nonterminal A_c and create a new rule $A_c(c) \leftarrow$. For each non-terminating rule that has $k \geq 1$ terminal occurrences $c_1, \ldots, c_k$ in the head, we replace those occurrences by distinct fresh variables $z_1, \ldots, z_k$, and add $A_{c_1}(z_1), \ldots, A_{c_k}(z_k)$ to the body.

Second, we change all nonterminals of arity 1 to nonterminals of arity 2 by the following procedure. If $A(\boldsymbol{x})$ appears as a subgoal of a non-terminating rule, then

we replace that occurrence of $A(\boldsymbol{x})$ by $A(z_1, z_2)$ and the occurrence of $\boldsymbol{x}$ in the head of the rule with $z_1 z_2$, where z_1, z_2 are fresh variables. After this rewriting is performed for all rules and all nonterminals of arity 1, we replace all rules of the form $A(\alpha) \leftarrow \ldots$ with $A(\alpha, \varepsilon) \leftarrow \ldots$ (including terminating rules).

After the preceding transformations, every non-terminating rule of the grammar is of the following form:

$$B(\alpha_1, \alpha_2) \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2})$$

where $n \geq 1$ and $\alpha_1, \alpha_2 \in \{\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}, \ldots, \boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}\}^*$. We introduce a new nonterminal E, create a new rule $E(\varepsilon, \varepsilon) \leftarrow$, and replace each non-terminating rule

$$B(\alpha_1, \alpha_2) \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2})$$

by

$$B(\alpha_1 z_1, z_2 \alpha_2) \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}), E(z_1, z_2),$$

where z_1, z_2 are fresh variables.

Now the length of the right-hand side of every non-terminating rule is ≥ 2. We can rearrange subgoals of non-terminating rules, so that each non-terminating rule has the form

$$B(\beta \boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2} \gamma) \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}), \tag{5.1}$$

where for every j, k such that $1 \leq j < k \leq n$, $\boldsymbol{x}_{j,1}$ appears to the left of $\boldsymbol{x}_{k,1}$ in $\beta \boldsymbol{x}_{i,1} \boldsymbol{x}_{i,2} \gamma$.

Finally, we make all non-terminating rules have at most two subgoals by repeatedly applying the following procedure as long as there remains any non-terminating rule with more than two subgoals. The resulting grammar will be of the desired form.

Let

$$B(\alpha_1, \alpha_2) \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}), \tag{5.2}$$

be a non-terminating rule with $n \geq 3$ ($\alpha_1 = \beta \boldsymbol{x}_{i,1}, \alpha_2 = \boldsymbol{x}_{i,2} \gamma$). Let δ, ζ be such that

$$\boldsymbol{x}_{1,1} \delta \boldsymbol{x}_{1,2} \zeta = \alpha_1 \alpha_2.$$

Note $|\delta \zeta| \geq 4$.

Case 1. $\delta = \varepsilon$.

Case 1.1. $\alpha_1 = \boldsymbol{x}_{1,1}$. Then $\alpha_2 = \boldsymbol{x}_{1,2} \gamma' \boldsymbol{x}_{n,1} \boldsymbol{x}_{n,2} \gamma''$. Let C be a fresh nonterminal and $\boldsymbol{y}_1, \boldsymbol{y}_2$ be fresh variables. We replace (5.2) by two rules:

$$B(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2} \boldsymbol{y}_1 \boldsymbol{y}_2) \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), C(\boldsymbol{y}_1, \boldsymbol{y}_2).$$

$$C(\gamma' \boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}\gamma'') \leftarrow B_2(\boldsymbol{x}_{2,1}, \boldsymbol{x}_{2,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}).$$

Case 1.2. $\alpha_1 = \boldsymbol{x}_{1,1}\boldsymbol{x}_{1,2}\beta'\boldsymbol{x}_{i,1}$, $\alpha_2 = \boldsymbol{x}_{i,2}\gamma$. Let C be a fresh nonterminal and $\boldsymbol{y}_1, \boldsymbol{y}_2$ be fresh variables. We replace (5.2) by two rules:

$$B(\boldsymbol{x}_{1,1}\boldsymbol{x}_{1,2}\boldsymbol{y}_1, \boldsymbol{y}_2) \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), C(\boldsymbol{y}_1, \boldsymbol{y}_2).$$
$$C(\beta' \boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}\gamma) \leftarrow B_2(\boldsymbol{x}_{2,1}, \boldsymbol{x}_{2,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}).$$

Case 2. $\delta \neq \varepsilon$.

Case 2.1. $\alpha_1 = \boldsymbol{x}_{1,1}\delta'\boldsymbol{x}_{i,1}$, $\alpha_2 = \boldsymbol{x}_{i,2}\delta''\boldsymbol{x}_{1,2}\zeta$.

Case 2.1.1. $\zeta = \varepsilon$. Let C be a fresh nonterminal and $\boldsymbol{y}_1, \boldsymbol{y}_2$ be fresh variables. We replace (5.2) by two rules:

$$B(\boldsymbol{x}_{1,1}\boldsymbol{y}_1, \boldsymbol{y}_2\boldsymbol{x}_{1,2}) \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), C(\boldsymbol{y}_1, \boldsymbol{y}_2).$$
$$C(\delta' \boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}\delta'') \leftarrow B_2(\boldsymbol{x}_{2,1}, \boldsymbol{x}_{2,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}).$$

Case 2.1.2. $\zeta \neq \varepsilon$. Then $\zeta = \boldsymbol{x}_{j,1}\zeta'$ for some $j > i$.

Case 2.1.2.1. $\delta'\delta'' = \varepsilon$. Then $i = 2$ and $j = 3$. Let C be a fresh nonterminal and $\boldsymbol{y}_1, \boldsymbol{y}_2$ be fresh variables. We replace (5.2) by two rules:

$$B(\boldsymbol{y}_1\boldsymbol{x}_{2,1}, \boldsymbol{x}_{2,2}\boldsymbol{y}_2) \leftarrow C(\boldsymbol{y}_1, \boldsymbol{y}_2), B_2(\boldsymbol{x}_{2,1}, \boldsymbol{x}_{2,2}).$$
$$C(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}\boldsymbol{x}_{3,1}\zeta') \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), B_3(\boldsymbol{x}_{3,1}, \boldsymbol{x}_{3,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}).$$

Case 2.1.2.2. $\delta'\delta'' \neq \varepsilon$. Let C, D be fresh nonterminals and $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{z}_1, \boldsymbol{z}_2$ be fresh variables. We replace (5.2) by three rules:

$$B(\boldsymbol{y}_1\boldsymbol{z}_1, \boldsymbol{z}_2\boldsymbol{y}_2) \leftarrow C(\boldsymbol{y}_1, \boldsymbol{y}_2), D(\boldsymbol{z}_1, \boldsymbol{z}_2).$$
$$C(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}\boldsymbol{x}_{j,1}\zeta') \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), B_j(\boldsymbol{x}_{j,1}, \boldsymbol{x}_{j,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}).$$
$$D(\delta' \boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}\delta'') \leftarrow B_2(\boldsymbol{x}_{2,1}, \boldsymbol{x}_{2,2}), \ldots, B_{j-1}(\boldsymbol{x}_{j-1,1}, \boldsymbol{x}_{j-1,2}).$$

Case 2.2. $\alpha_1 = \boldsymbol{x}_{1,1}\delta\boldsymbol{x}_{1,2}\zeta'\boldsymbol{x}_{i,1}$, $\alpha_2 = \boldsymbol{x}_{i,2}\gamma$. There is some j with $1 < j < i$ such that $\delta = \delta'\boldsymbol{x}_{j,1}\boldsymbol{x}_{j,2}\delta''$.

Case 2.2.1. $\gamma = \varepsilon$. Then $i = n$. Let C be a fresh nonterminal and $\boldsymbol{y}_1, \boldsymbol{y}_2$ be fresh variables. We replace (5.2) by two rules:

$$B(\boldsymbol{y}_1\boldsymbol{y}_2\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}) \leftarrow C(\boldsymbol{y}_1, \boldsymbol{y}_2), B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}).$$
$$C(\boldsymbol{x}_{1,1}\delta'\boldsymbol{x}_{j,1}, \boldsymbol{x}_{j,2}\delta''\boldsymbol{x}_{1,2}\zeta') \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, B_{n-1}(\boldsymbol{x}_{n-1,1}, \boldsymbol{x}_{n-1,2}).$$

Case 2.2.2. $\gamma \neq \varepsilon$. Then $\zeta'\boldsymbol{x}_{i,1} = \boldsymbol{x}_{k,1}\zeta''$ for some $k \leq i$. Let C, D be fresh nonterminals and $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{z}_1, \boldsymbol{z}_2$ be fresh variables. We replace (5.2) by three rules:

$$B(\boldsymbol{y}_1\boldsymbol{y}_2\boldsymbol{z}_1, \boldsymbol{z}_2) \leftarrow C(\boldsymbol{y}_1, \boldsymbol{y}_2), D(\boldsymbol{z}_1, \boldsymbol{z}_2).$$
$$C(\boldsymbol{x}_{1,1}\delta'\boldsymbol{x}_{j,1}, \boldsymbol{x}_{j,2}\delta''\boldsymbol{x}_{1,2}) \leftarrow B_1(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, B_{k-1}(\boldsymbol{x}_{k-1,1}, \boldsymbol{x}_{k-1,2}).$$
$$D(\zeta' \boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}\gamma) \leftarrow B_k(\boldsymbol{x}_{k,1}, \boldsymbol{x}_{k,2}), \ldots, B_n(\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}).$$

In all cases, new non-terminating rules will be well-nested and of the form (5.1) with $n \geq 2$. □

From tree-adjoining grammars to head grammars

The fact that each tree-adjoining grammar has an equivalent well-nested 2-MCFG is easy to see once you recognize a certain simple fact about the adjunction operation.

Consider a tree β with a designated leaf q labeled by a special symbol $\Box$. We assume that $\Box$ does not label any other nodes of β, so that the designated leaf is uniquely determined by β. Note that $\mathbf{y}(\beta) = v\Box w$ for some strings v, w. Let us write $\mathbf{y}'(\beta)$ for the pair of strings (v, w). Now let α be a tree, and let p be a node of α. Since the designated leaf q of β is uniquely determined by β, we may as well use the notation $\alpha[p \leftarrow \beta]$ for the result of adjunction $\mathrm{adjoin}(\alpha, p, \beta, q)$. Note that if α has a unique designated leaf r, $\alpha[p \leftarrow \beta]$ again has a unique designated leaf labeled by $\Box$, and the function $\mathbf{y}'$ is defined on it.

It is easy to see that

$$\mathbf{y}(\alpha[p \leftarrow \beta]) = xvywz$$

for some strings x, y, z, where $(v, w) = \mathbf{y}'(\beta)$. When α has a unique designated leaf r labeled by $\Box$, we have

$$\begin{aligned} x &= x_1 \Box x_2 \quad \text{if } p \text{ lies to the right of } r, \\ y &= y_1 \Box y_2 \quad \text{if } p \leqslant r, \\ z &= z_1 \Box z_2 \quad \text{if } p \text{ lies to the left of } r, \end{aligned}$$

and hence

$$\mathbf{y}'(\alpha[p \leftarrow \beta]) \begin{cases} (x_1, x_2 v y w z) & \text{if } p \text{ lies to the right of } r, \\ (xvy_1, y_2 w z) & \text{if } p \leqslant r, \\ (xvywz_1, z_2) & \text{if } p \text{ lies to the left of } r. \end{cases}$$

So we always have

$$\mathbf{y}(\alpha[p \leftarrow \beta]) = \mathbf{y}(\alpha[p \leftarrow \beta']) \quad \text{whenever} \quad \mathbf{y}'(\beta) = \mathbf{y}'(\beta'),$$

and when α has a designated leaf, we also have

$$\mathbf{y}'(\alpha[p \leftarrow \beta]) = \mathbf{y}'(\alpha[p \leftarrow \beta']) \quad \text{whenever} \quad \mathbf{y}'(\beta) = \mathbf{y}'(\beta').$$

In other words, α and p determine a certain function $g_{\alpha,p}$ that maps a pair of strings to a string such that

$$\mathbf{y}(\alpha[p \leftarrow \beta]) = g_{\alpha,p}(\mathbf{y}'(\beta)),$$

where $g_{\alpha,p}((v, w)) = xvywz$ for some strings x, y, z. Also, when α has a designated leaf,

$$\mathbf{y}'(\alpha[p \leftarrow \beta]) = h_{\alpha,p}(\mathbf{y}'(\beta)),$$

where $h_{\alpha,p}((v,w))$ is one of (x_1, x_2vywz), (xvy_1, y_2wz), $(xvywz_1, z_2)$, depending on where p lies relative to α's designated leaf.

We may express these properties with "commutative diagrams", as in Figure 5.1.

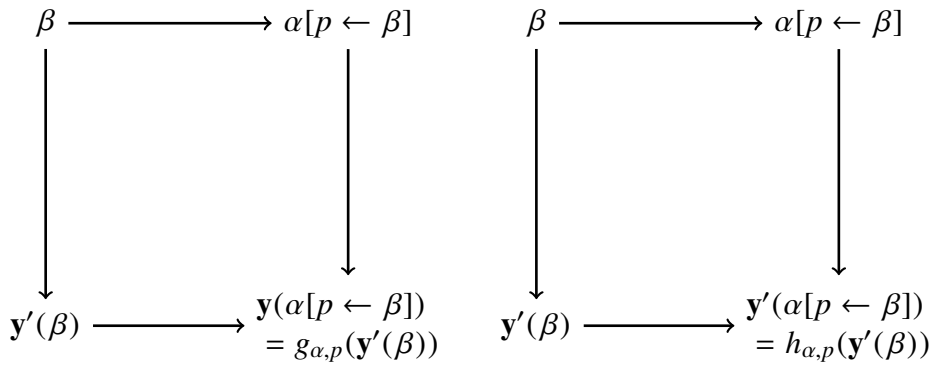


Figure 5.1: Adjunction commutes with "yield".

A similar property holds with iterated adjunction $\alpha[p_1 \leftarrow \beta_1, \ldots, p_n \leftarrow \beta_n]$. If $p_1, \ldots, p_n$ are distinct nodes of α (listed in preorder), we always have

$$\mathbf{y}(\alpha[p_1 \leftarrow \beta_1, \ldots, p_n \leftarrow \beta_n]) = g_{\alpha,p_1,\ldots,p_n}(\mathbf{y}'(\beta_1), \ldots, \mathbf{y}'(\beta_n))$$

for some function $g_{\alpha,p_1,\ldots,p_n}$, where $g_{\alpha,p_1,\ldots,p_n}((v_1,w_1),\ldots,(v_n,w_n))$ is formed by concatenating $v_1, w_1, \ldots, v_n, w_n$ and $2n+1$ fixed strings in some specified way. Moreover, it is easy to see that the function $g_{\alpha,p_1,\ldots,p_n}$ is "well-nested". Similarly, when α has a designated leaf, we have

$$\mathbf{y}'(\alpha[p_1 \leftarrow \beta_1, \ldots, p_n \leftarrow \beta_n]) = h_{\alpha,p_1,\ldots,p_n}(\mathbf{y}'(\beta_1), \ldots, \mathbf{y}'(\beta_n))$$

for some "well-nested" function $h_{\alpha,p_1,\ldots,p_n}$, where $h_{\alpha,p_1,\ldots,p_n}((v_1,w_1),\ldots,(v_n,w_n))$ is a pair of strings formed by concatenating $v_1, w_1, \ldots, v_n, w_n$ and $2n+2$ fixed strings.

This allows us to construct a well-nested 2-MCFG G' from the multi-start CFG $\mathrm{cf}(G)$ corresponding to a tree-adjoining grammar G that we used to define the derivation trees of G. Corresponding to a production

$$[\alpha p_1 \ldots p_n] \to [\beta_1 q_{1,1} \ldots q_{1,m_1}] \ldots [\beta_n q_{n,1} \ldots q_{n,m_n}]$$

of $\mathrm{cf}(G)$, the 2-MCFG has the rule

$$[\alpha p_1 \ldots p_n](\delta) \leftarrow [\beta_1 q_{1,1} \ldots q_{1,m_1}](\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, [\beta_1 q_{n,1} \ldots q_{n,m_n}](\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}),$$

where $\delta = g_{\alpha,p_1,\ldots,p_n}((\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, (\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}))$, if α is an initial tree, or the rule

$$[\alpha p_1 \ldots p_n](\delta_1, \delta_2) \leftarrow [\beta_1 q_{1,1} \ldots q_{1,m_1}](\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, [\beta_1 q_{n,1} \ldots q_{n,m_n}](\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}),$$

where $(\delta_1, \delta_2) = h_{\alpha,p_1,\ldots,p_n}((\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), \ldots, (\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}))$, if α is an auxiliary tree. Each terminal

$$[\alpha]$$

of $\mathrm{cf}(G)$ is translated into a terminating rule of G'. If α is an initial tree, then G' has the terminating rule

$$[\alpha](w) \leftarrow ,$$

where $w = \mathbf{y}(\alpha)$. If α is an auxiliary tree, then G' has the terminating rule

$$[\alpha](v, w) \leftarrow ,$$

where $(v, w) = \mathbf{y}'(\alpha)$. In addition, we need to add a start nonterminal S and rules

$$S(\boldsymbol{x}) \leftarrow [\alpha p_1 \ldots p_n](\boldsymbol{x})$$

for initial trees α.

Theorem 5.3. *For every TAG G, there is a well-nested 2-MCFG G' such that $L(G) = L(G')$.*

Corollary 5.4. *For every TAG G, there is an HG G' such that $L(G) = L(G')$.*

Exercise 5.2. Construct a head grammar that is equivalent to the tree-adjoining grammar for COUNT-4 given in Lecture 4.

Exercise 5.3. Convert the TAG you wrote in Exercise 4.2 to a well-nested 2-MCFG.

From head grammars to tree-adjoining grammars

Given an HG $G = (N, \Sigma, P, S)$, we construct a TAG $G' = (N, \Sigma, \mathscr{I}, \mathscr{A})$ such that $L(G) = L(G')$. By Theorem 5.2, we may assume without loss of generality that the length of the right-hand side of every rule in P is at most 2, so that each rule is of one of the following forms:

$$S(\boldsymbol{x}\boldsymbol{y}) \leftarrow A(\boldsymbol{x}, \boldsymbol{y}), \tag{5.3}$$

$$A(\boldsymbol{x}_1, \boldsymbol{y}_1\boldsymbol{x}_2\boldsymbol{y}_2) \leftarrow B(\boldsymbol{x}_1, \boldsymbol{y}_1), C(\boldsymbol{x}_2, \boldsymbol{y}_2), \tag{5.4}$$

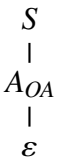
$$A(\boldsymbol{x}_1\boldsymbol{y}_1\boldsymbol{x}_2, \boldsymbol{y}_2) \leftarrow B(\boldsymbol{x}_1, \boldsymbol{y}_1), C(\boldsymbol{x}_2, \boldsymbol{y}_2), \tag{5.5}$$

$$A(\boldsymbol{x}_1\boldsymbol{x}_2, \boldsymbol{y}_2\boldsymbol{y}_1) \leftarrow B(\boldsymbol{x}_1, \boldsymbol{y}_1), C(\boldsymbol{x}_2, \boldsymbol{y}_2), \tag{5.6}$$

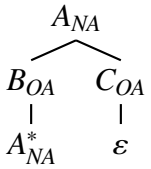
$$A(w_1, w_2) \leftarrow . \tag{5.7}$$

$\mathscr{I}$ and $\mathscr{A}$ are constructed as follows:

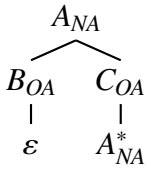
- For each rule of the form (5.3), $\mathscr{I}$ contains the initial tree



- For each rule of the form (5.4), $\mathscr{A}$ contains the auxiliary tree

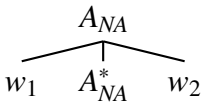


- For each rule of the form (5.5), $\mathscr{A}$ contains the auxiliary tree



- For each rule of the form (5.6), $\mathscr{A}$ contains the auxiliary tree

- For each rule of the form (5.7), $\mathscr{A}$ contains the auxiliary tree



Theorem 5.5. *For every HG G, there is a TAG G' such that $L(G) = L(G')$.*

Exercise 5.4. Describe a procedure for directly converting arbitrary HGs (without the constraint on the length of the righ-hand side) to TAGs.

The equivalence of TAG, LIG, and HG is due to Vijay-Shanker and Weir (1994), who proved it using a certain special version of *combinatory categorial grammar* (CCG) (see Steedman 2000) as an intermediate step from TAG to LIG. As we have seen, the conversion between TAGs and HGs is easy, while that between TAGs and LIGs requires some work. This is because TAGs and HGs are similar formalisms, with sets of derivation trees that can be described by context-free grammars. On the other hand, the set of derivation trees of a LIG is in general unlike the set of derivation trees of any context-free grammar; ignoring the stack of indices, it is the tree language of some TAG.

The convergence of mildly context-sensitive grammar formalisms

Joshi (1985) introduced the notion of *mildly context-sensitive grammar*. The associated languages are *mildly context-sensitive languages*. Mildly context-sensitive grammars are supposed to

- generate all context-free languages,
- allow polynomial-time parsing,
- capture nested dependencies and certain limited kinds of crossing dependencies, and
- generate only languages with the *constant growth property*.

By the second condition, the class of mildly context-sensitive languages is a subclass of the complexity class P. The third condition is vague, but is meant to allow nested dependencies and certain types of crossed dependencies, but not others. It was suggested by Joshi et al. (1991) that it may be reasonable to construe this condition as excluding $\mathrm{MIX} = \{ w \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}^* \mid \#_{\mathsf{a}}(w) = \#_{\mathsf{b}}(w) = \#_{\mathsf{c}}(w) \}$ from the class of mildly context-sensitive languages.[7]

A language L is said to have the *constant growth property* if either L is finite or there exists a constant c such that for every $w \in L$, there is a $w' \in L$ with $|w| < |w'| \leq |w|+c$. The constant growth property is a consequence of the stronger notion of *semilinearity*. A language $L \subseteq \{\mathsf{a}_1, \ldots, \mathsf{a}_k\}^*$ is said to be *semilinear* iff $\{ \#(w) \mid w \in L \}$ is a semilinear subset of $\mathbb{N}^k$, where $\#(w) = (\#_{\mathsf{a}_1}(w), \ldots, \#_{\mathsf{a}_k}(w))$. A subset of $\mathbb{N}^k$ is *semilinear* if it is a finite union of linear sets. A set $A \subseteq \mathbb{N}^k$ is *linear* if there are $\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_m \in \mathbb{N}^k$ such that $A = \{ \vec{u}_0 + \sum_{i=1}^{m} c_i \vec{u}_i \mid c_1, \ldots, c_m \in \mathbb{N} \}$.

[7] If d is a symbol and w is a string, $\#_d(w)$ denotes the number of occurrences of d in w.

It is clear that all grammar formalisms we have looked at so far satisfy all of Joshi's (1985) conditions for mild context-sensitivity, except possibly the third condition about "limited crossing dependencies". The generated languages are all semilinear and belong to the complexity class *LOGCFL*, which is a subclass of P. Since the third condition is only vaguely defined, it is impossible to say with certainty whether or not a given formalism satisfies this condition. Joshi et al. (1991) clearly indicated, however, that *set-local multi-component tree-adjoining grammars* (MCTAGs) are mildly context-sensitive. If this is so, then MCFGs (LCFRSs) are mildly context-sensitive as well, since set-local MCTAGs are equivalent to them in string-generating power.

Joshi et al. (1991) suggested that the fact that the superficially quite distinct formalisms of TAGs, LIGs, HGs, and (the relevant variant of) CCGs are all equivalent indicate that "we are getting a handle on some fundamental properties of the objects that these formalisms were designed to describe". This is slightly offset by the fact that the relevant variant of CCG used by Vijay-Shanker and Weir (1994) seems rather artificial—it does not include Raising or Substitution among the modes of combination even though these are important in Steedman's work (see, e.g., Steedman 2000), and it allows the empty string in lexical entries, which goes against the spirit of categorial grammars. In retrospect, the larger class of MCFLs has turned out to be much more robust, with a wide range of different formalisms generating it, including, besides MCFGs (LCFRSs), *set-local multi-component tree-adjoining grammars* (MCTAGs) (Weir 1988), *deterministic tree walking transducers* (Engelfriet and Heyker 1991, Weir 1992), *deterministic finite-copying top-down tree transducers* (Engelfriet et al. 1980), *hyperedge replacement graph grammars* (Engelfriet and Heyker 1991), *minimalist gramamrs* (Michaelis 2001a,c, Harkema 2001), and *second-order abstract categorial grammars* (de Groote and Pogodalla 2004, Salvati 2007, Kanazawa and Salvati 2007).

Even though the class of MCFLs has long been regarded by many researchers as an adequate formalization of Joshi's notion of mildly context-sensitivity, this issue is far from settled. This was highlighted by a couple of recent results about MIX. Salvati (2011, 2015) surprisingly proved that MIX is a 2-MCFL,[8] while Kanazawa and Salvati (2012) proved Joshi's (1985) conjecture that MIX is not a TAL. If MIX is not a mildly context-sensitive language since it violates the condition of limited crossing dependencies, as Joshi et al. (1991) suggested, then MCFGs and equivalent formalisms should not be considered mildly context-sensitive.

Joshi's (1985) goal in proposing the notion of mild context-sensitivity was to capture a minimal extension of context-free grammars that is sufficient for adequate description of natural language syntax. His set of criteria, especially the third,

[8]Salvati's (2011, 2015) proof uses tools from algebraic topology. See Nederhof (to appear) for a recent attempt at a shorter proof of Salvati's theorem.

need not be considered the final word and may be improved upon. We will revisit the questions of what criteria might serve Joshi's goal best and of what is the best formalization of mild context-sensitivity.

Parikh's theorem

Any grammar whose set of derivation trees can be described by a context-free grammar generates a semilinear language, provided that the function from derivation trees to derived strings is "linear" in some suitable sense. This follows from the proof of *Parikh's theorem* (Parikh 1966), which says that every context-free language is semilinear.

Parikh's theorem was famously omitted from Hopcroft and Ullman's (1979) textbook, but many other texts include its proof. In the following proof, we borrow some notation and terminology from Kozen (1997).[9]

Let $G = (N, \Sigma, P, S)$ be a context-free grammar. As was defined in Lecture 1, a *parse tree* is any tree satisfying the following properties:

1. each non-leaf node is labeled by a nonterminal,
2. each leaf node is labeled by a terminal or ε,
3. the root node is labeled by the start symbol,
4. for each non-leaf node labeled by a nonterminal A, G has a production $A \to \alpha$ such that α is the string obtained by reading the labels of its children, from left to right, and
5. any leaf node labeled by ε has no sibling.

A *pump* is any tree with at least two nodes satisfying conditions 1, 4, 5, and

2′. each leaf node is labeled by a terminal or ε, except for one, which is labeled by the same nonterminal as the root.

An *initial* parse tree is one in which no nonterminal labels two distinct nodes on the same path. The path from the root to the only nonterminal-labeled leaf in a pump is called its *spine*. A *basic* pump is one in which no nonterminal labels two distinct nodes on the spine or on any path disjoint from the spine, except that the root and the leaf on the spine have the same label. It is easy to see that there are only finitely many initial parse trees and basic pumps.

If τ is a pump, we let $\mathrm{root}(\tau)$ denote the label of its root. If τ is a parse tree or a pump, $\mathrm{nont}(\tau)$ denotes the set of nonterminals that occur in τ.

[9] My thanks to Sylvain Pogodalla for pointing out a mistake in an earlier version of this proof.

If τ is a parse tree and υ is a basic pump, we write $\tau \overset{\upsilon}{\lhd_1} \tau'$ to mean that τ' is the result of adjoining υ at some node p of τ that is labeled by root(υ). (Formally, if q is the only nonterminal-labeled leaf of υ, $\tau' = \text{adjoin}(\tau, p, \upsilon, q)$, in the notation of Lecture 4.) Clearly, this implies that τ' is also a parse tree. The notation $\tau \lhd_1 \tau'$ means $\tau \overset{\upsilon}{\lhd_1} \tau'$ holds for some basic pump υ. We denote the reflexive transitive closure of $\lhd_1$ by $\lhd$, and we write $\tau \overset{\upsilon_1 \ldots \upsilon_n}{\lhd} \tau'$ to mean there are $\tau_0, \tau_1, \ldots, \tau_n$ such that $\tau_0 = \tau$, $\tau_n = \tau'$ and for each $i = 1, \ldots, n$, it holds that $\tau_{i-1} \overset{\upsilon_i}{\lhd_1} \tau_i$. If τ is an initial parse tree and $\tau \overset{\upsilon_1 \ldots \upsilon_n}{\lhd} \tau'$, we call the sequence $(\tau, \upsilon_1, \ldots, \upsilon_n)$ a *valid pump sequence* (from τ).

Note that $\tau \overset{\upsilon}{\lhd_1} \tau'$ for some τ' if and only if root(υ) $\in$ nont(τ). Note also that if $\tau \overset{\upsilon_1 \ldots \upsilon_n}{\lhd} \tau'$, then $\text{nont}(\tau') = \text{nont}(\tau) \cup \text{nont}(\upsilon_1) \cup \cdots \cup \text{nont}(\upsilon_n)$.

Lemma 5.6. *If τ is a parse tree whose root node is labeled by S, then there is an initial parse tree τ_0 such that $\tau_0 \lhd \tau$.*

Proof. Induction on the size of τ. If τ is an initial parse tree, (ii) holds trivially. Otherwise, τ has a node p labeled by nonterminal A that is an ancestor of another node labeled by A. Take a lowest such node p, and let q be a descendant of p that has the same label. It is easy to see that the set of nodes that are descendants of p (including p itself) but are not descendants of q constitutes a basic pump υ, and the remaining nodes plus p form a smaller parse tree τ' such that $\tau' \overset{\upsilon}{\lhd_1} \tau$. □

Lemma 5.7. *Let $(\tau, \upsilon_1, \ldots, \upsilon_n)$ be a valid pump sequence.*

(i) *$(\tau, \upsilon_1, \ldots, \upsilon_n, \upsilon_i)$ is a valid pump sequence for every $i = 1, \ldots, n$.*

(ii) *If $i < j$ and $\upsilon_i = \upsilon_j$, then $(\tau, \upsilon_1, \ldots, \upsilon_{j-1}, \upsilon_{j+1}, \ldots, \upsilon_n)$ is a valid pump sequence.*

Proof. Let $\tau \overset{\upsilon_1 \ldots \upsilon_i}{\lhd} \tau_i$ for every $i \leq n$.

(i). Since υ_i is adjoinable into τ_{i-1}, we must have root(υ_i) $\in$ nont(τ_{i-1}) = nont(τ) $\cup$ nont(υ_1) $\cup \cdots \cup$ nont(υ_{i-1}). Then $\upsilon_i \in \text{nont}(\tau_n) = \text{nont}(\tau) \cup \text{nont}(\upsilon_1) \cup \cdots \cup \text{nont}(\upsilon_n) \supseteq \text{nont}(\tau_{i-1})$.

(ii). For every $k > j$, we have root(υ_k) $\in$ nont(τ) $\cup$ nont(υ_1) $\cup \cdots \cup$ nont(υ_{k-1}) = nont(τ) $\cup$ nont(υ_1) $\cup \cdots \cup$ nont(υ_{j-1}) $\cup$ nont(υ_{j+1}) $\cup \cdots \cup$ nont(υ_{k-1}), so we can show by induction that $(\tau, \upsilon_1, \ldots, \upsilon_{j-1}, \upsilon_{j+1}, \ldots, \upsilon_k)$ is a valid pump sequence. □

Let $\Sigma = \{\mathsf{a}_1, \ldots, \mathsf{a}_k\}$. If τ is a parse tree or a pump, define

$$\#(\tau) = (\#_{\mathsf{a}_1}(\tau), \ldots, \#_{\mathsf{a}_k}(\tau)),$$

where $\#_c(\tau)$ is the number of occurrences of c in τ. For a valid pump sequence $(\tau, \upsilon_1, \ldots, \upsilon_n)$, define

$$\#(\tau, \upsilon_1, \ldots, \upsilon_n) = \#(\tau) + \#(\upsilon_1) + \cdots + \#(\upsilon_n).$$

It is easy to see that if $\tau \overset{\upsilon_1 \ldots \upsilon_n}{\lhd} \tau'$, we have

$$\#(\tau') = \#(\tau, \upsilon_1, \ldots, \upsilon_n).$$

Lemma 5.8. *Let τ be an initial parse tree. Then $P_\tau = \{ \#(\tau') \mid \tau \lhd \tau' \}$ is semilinear.*

It is important to understand that P_τ need not be linear.[10] Let

$U_\tau = \{ \upsilon \mid \upsilon$ is a basic pump that appears in some valid pump sequence from $\tau \}$.

In other words,

$$U_\tau = \{ \upsilon \mid \upsilon \text{ is a basic pump such that } \mathrm{root}(\upsilon) \in N_\tau \},$$

where N_τ is the smallest set satisfying the condition:

$$\mathrm{nont}(\tau) \subseteq N_\tau \wedge \forall \upsilon((\upsilon \text{ is a basic pump} \wedge \mathrm{root}(\upsilon) \in N_\tau) \rightarrow \mathrm{nont}(\upsilon) \subseteq N_\tau).$$

Let $\beta_1, \ldots, \beta_m$ list the elements of U_τ. Let $\vec{u} = \#(\tau)$ and $\vec{v}_i = \#(\beta_i)$ for $i = 1, \ldots, m$. Then

$$P_\tau \subseteq \{ \vec{u} + \sum_{i=1}^{m} c_i \vec{v}_i \mid c_i \in \mathbb{N} \text{ for each } i = 1, \ldots, m \},$$

but the converse inclusion need not hold. The reason is that

$$\vec{u} + c_1\vec{v}_1 + \cdots + c_{i-1}\vec{v}_{i-1} + \vec{v}_i + c_{i+1}\vec{v}_{i+1} + \cdots + c_m\vec{v}_m \in P_\tau$$

does not imply

$$\vec{u} + c_1\vec{v_1} + \cdots + c_{i-1}\vec{v}_{i-1} + c_{i+1}\vec{v}_{i+1} + \ldots c_m\vec{v}_m \in P_\tau,$$

because there may be some j such that β_j can adjoin only after β_i does, which will be the case when $\mathrm{root}(\beta_j) \in \mathrm{nont}(\beta_i) - (\mathrm{nont}(\tau) \cup \bigcup_{h \in \{1,\ldots,m\} - \{i,j\}} \mathrm{nont}(\beta_h))$.

For $M \subseteq \{1, \ldots, m\}$, define

$$Q_M = \{ \vec{u} + \sum_{i \in M} c_i \vec{v}_i \mid c_i \in \mathbb{N}_+ \text{ for } i \in M \}.$$

[10] Rich (2007) incorrectly claims that this set is linear.

($\mathbb{N}_+$ is the set of positive integers.) Clearly, Q_M is a linear set. To prove Lemma 5.8, it suffices to prove that P_τ is a union of sets of the form Q_M. Let

$$\mathscr{M} = \{\, \{i_1, \ldots, i_n\} \mid (\tau, \beta_{i_1}, \ldots, \beta_{i_n}) \text{ is a valid pump sequence} \,\}.$$

It is not difficult to show

$$P_\tau = \bigcup_{M \in \mathscr{M}} Q_M$$

using Lemma 5.7

Exercise 5.5. Give an example of a CFG G and an initial parse tree τ of G such that P_τ is not linear.

Parikh's Theorem immediately follows from Lemma 5.8.

Theorem 5.9 (Parikh 1966)**.** *If L is a context-free language, then $\{\, \#(w) \mid w \in L \,\}$ is semilinear.*

Derivation trees of MCFGs

To apply the proof of Parikh's Theorem to MCFGs, we need to define the notion of a derivation tree for MCFGs. We could adopt a definition analogous to one for Datalog where each node is labeled by a derivable fact of the form $B(w_1, \ldots, w_r)$, with each w_i being a string of terminals. That would make the set of labels infinite, and it would not be a convenient representation for the present purpose because we want to be able to build all derivation trees from a finite number of "initial" derivation trees and a finite number of "basic pumps". For this reason, we label nodes of derivation trees by MCFG rules. (Such trees may be called *rule trees* in contradistinction to trees analogous to derivation trees of Datalog.)

We assume that a fixed infinite set

$$X = \{\, \boldsymbol{x}_{i,j} \mid i, j \geq 1 \,\}$$

of variables is given, and each rule of an MCFG is of the form

$$B(\alpha_1, \ldots, \alpha_r) \leftarrow B_1(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{1,r_1}), \ldots, B_n(\boldsymbol{x}_{n,1}, \ldots, \boldsymbol{x}_{n,r_n}),$$

where r is the rank of B, r_i is the rank of B_i, and $\alpha_i \in (\Sigma \cup \{\, \boldsymbol{x}_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq r_i \,\})^*$.

Let $G = (N, \Sigma, P, S)$ be an MCFG. We say that a node in a tree is of type B if it is labeled by a rule whose left-hand side nonterminal is B. A *derivation tree* of G is any tree satisfying the following conditions:

1. each node is labeled by some rule in P,
2. for each node p, if p is labeled by $B(\alpha_1, \ldots, \alpha_r) \leftarrow B_1(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{1,r_1}), \ldots, B_n(\boldsymbol{x}_{n,1}, \ldots, \boldsymbol{x}_{n,r_n})$, then p has n children, and for $i = 1, \ldots, n$, the ith child of p is of type B_i.

The *derived string* $\mathrm{dstr}(\tau)$ of a derivation tree τ is defined inductively as follows. If

$$\tau = \begin{array}{c} B(\alpha_1, \ldots, \alpha_r) \leftarrow B_1(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{1,r_1}), \ldots, B_n(\boldsymbol{x}_{n,1}, \ldots, \boldsymbol{x}_{n,r_n}) \\ \tau_1 \quad \cdots \quad \tau_n \end{array}$$

and for each $i = 1, \ldots, n$,

$$\mathrm{dstr}(\tau_i) = (w_{i,1}, \ldots, w_{i,r_i}),$$

then

$$\mathrm{dstr}(\tau) = (\alpha_1, \ldots, \alpha_r)\sigma,$$

where σ is the substitution that sends $\boldsymbol{x}_{i,j}$ to $w_{i,j}$.

For any MCFG G, the set of derivation trees of G coincides with the set of parse trees of a context-free grammar $\mathrm{cf}(G)$ determined by G. With appropriate changes to the definition of $\#(\tau)$, we can satisfy the equation

$$\#(\tau) = \#(\mathrm{dstr}(\tau)),$$

and the proof of Parikh's Theorem turns into a proof of the following theorem:

Theorem 5.10 (Vijay-Shanker et al. 1987)**.** *If L is an MCFL, then $\{ \#(w) \mid w \in L \}$ is semilinear.*

Exercise 5.6. Fill in details of the proof of Theorem 5.10.

Some non-MCS phenomena in natural language

Unlike Swiss German, transitive verbs in Dutch all mark their object NPs with accusative case, so the existence of cross-serial dependencies in Ducth does not in itself suffice to show that the set of grammatical word strings of Dutch is not a context-free language (Pullum and Gazdar 1982). Manaster-Ramer (1987) uses the interaction between coordination and cross-serial dependencies to show that the set of grammatical word strings in Dutch is not a context-free or tree-adjoining language:

(5.8) dat Jan Piet Marie liet opbellen, hoorde uitnodigen, hielp ontmoeten
that made call heard invite helped meet
en zag omhelzen
and saw embrace
'that Jan made Piet call Marie, heard [him] invite [her], helped [him] meet [her] and saw [him] embrace [her]'

(5.9) dat Jan Piet Marie^{n+1} liet laten^{n} opbellen, hoorde horen^{n} uitnodigen, hielp helpen^{n} ontmoeten en zag zien^{n} omhelzen

Exercise 5.7. Show that the set of strings of the form (5.9) is not a TAL.

This construction was taken up again by Groenink (1997a), who tried to demonstrate in detail Manaster-Ramer's (1987) observation that it can be used to show that Ducth is not even an MCFL:[11]

(5.10) dat Jan Piet Marie Fred^{n} (hoorde leren^{n} uitnodigen,)$^{k-1}$ en zag leren^{n} omhelzen

Radzinski (1991) discusses number names in Mandarin Chinese:

(5.11) a. wu zhao zhao wu zhao
five trillion trillion five trillion

b. wu zhao zhao zhao zhao zhao wu zhao zhao zhao zhao wu zhao zhao zhao wu zhao zhao wu zhao

c. * wu zhao zhao wu zhao zhao zhao

d. wu zhao zhao zhao zhao wu zhao zhao

e. * wu zhao zhao wu zhao zhao wu zhao zhao zhao zhao

According to Radzinski (1991), the word *zhao* is the single word number name in Chinese that expresses the higest numerical value, namely 10^{12} (i.e., 'trillion' in American English). Numbers that are $\geq 10^{24}$ are expressed by stringing together instances of *zhao*, as in (5.11a), which expresses $5 \times 10^{24} + 5 \times 10^{12}$, and (5.11b), which expresses $5 \times 10^{60} + 5 \times 10^{48} \times 5 \times 10^{36} + 5 \times 10^{24} + 5 \times 10^{12}$. There is a constraint on word order in this system of number representation, namely that longer strings of *zhao* must precede shorter strings of *zhao*. Thus, the set of grammatical number names that can be formed with *zhao* and *wu* is

(5.12) $J = \{ wu\ zhao^{k_1}\ wu\ zhao^{k_2} \ldots wu\ zhao^{k_n} \mid k_1 > k_2 > \cdots > k_n \geq 0 \}$.[12]

[11] In fact, some of the NPs can be omitted, so the number of NPs need not match the number of verbs in each conjunct. The proof that Ducth is not an MCFL still goes through, however (with the required modification explained below). See Manaster-Ramer 1987 for a detailed proof that Dutch is not a TAL.

[12] Radzinski (1991) has "$k_n > 0$" instead of "$k_n \geq 0$", but strings such as *wu zhao wu* are perfectly grammatical. (My thanks to Sheng Shaohua for confirming this.)

Radzinski (1991) claimed that this fact shows that the set of grammatical word strings of Chinese is not an MCFL.

Groenink (1997a) based his proof that the set of grammatical word strings of Dutch is not an MCFL on the claim that any MCFL is *k-pumpable* for some $k \geq 1$ in the following sense:

(5.13) **(Groenink)** A language L is (universally) *k-pumpable* if there is a constant p such that for any $z \in L$ with $|z| \geq p$, there are strings $u_0, \ldots, u_k$ and $v_1, \ldots, v_k$ such that

- $z = u_0 v_1 u_1 v_2 u_2 \ldots u_{k-1} v_k u_k$,
- $1 \leq |v_i| \leq p$ for each i, and
- $u_0 v_1^n u_1 v_2^n u_2 \ldots u_{k-1} v_k^n u_k \in L$ for every $n \geq 0$.

Radzinski based his proof that J is not an MCTAL (i.e., MCFL) on the following claim:

(5.14) **(Radzinski)** If L is an MCTAL, then there are constants n and m such that if $z \in L$ and $|z| \geq n$, then z may be written as $z = u_1 v_1 w_1 s_1 u_2 v_2 w_2 s_2 u_3 \ldots u_m v_m w_m s_m u_{m+1}$ with $\sum_{j=1}^{m} |v_j s_j| \geq 1$ such that for all $i \geq 0$, $u_1 v_1^i w_1 s_1^i u_2 v_2^i w_2 s_2^i u_3 \ldots u_m v_m^i w_m s_m^i u_{m+1} \in L$.

In the terminology of Greibach (1978), this says that every MCTAL is $2m$*-iterative* for some m. The difference from Groenink's notion is that there is no bound on the length of each of the string $v_1, s_1, \ldots, v_m, s_m$.

Both Radzinski (1991) and Groenink (1997b) claimed that the proof given by Seki et al. (1991) for Theorem 5.11 below actually establishes the stronger forms given above.[13] As a matter of fact, Seki et al.'s (1991) proof establishes no such thing.[14] For a long time, it was an open question whether or not the above strong form of the pumping lemma for MCFLs (MTALs) actually holds, as noted by Kanazawa and Salvati (2007). A recent paper by Kanazawa et al. (2014) finally settled the question in the negative.

Theorem 5.11 (Seki et al. 1991)**.** *For any m-MCFL L, if L is an infinite set, then there exist some* $u_j \in \Sigma^*$ $(1 \leq j \leq m+1)$, $v_j, w_j, s_j \in \Sigma^*$ $(1 \leq j \leq m)$ *which satisfy the following conditions:*

1. $\sum_{j=1}^{m} |v_j s_j| > 0$, *and*

[13] Radzinski (1991) refers to an earlier technical report (Kasami et al. 1987), which was later incorporated into Seki et al. 1991.

[14] See also Kracht 2003 for a "proof sketch" of Groenink's (1997b) claim.

2. *for any non-negative integer i,*

$$z_i = u_1 v_1^i w_1 s_1^i u_2 v_2^i w_2 s_2^i u_3 \ldots u_m v_m^i w_m s_m^i u_{m+1} \in L.$$

In Greibach's (1978) terminology, this theorem states that every m-MCFL is *weakly* $2m$*-iterative*.

Radzinski (1991) emphasizes that the stronger form (5.14) is essential to prove that the set J is not an MCFL. Seki et al's (1991) proof of their pumping lemma establishes the following slightly stronger statement, but it does not suffice for his purpose. (Below, if $\vec{u}, \vec{v} \in \mathbb{N}^l$, $\vec{u} \geq \vec{v}$ means that $(\vec{u})_i \geq (\vec{v})_i$ holds for all $i \leq l$, where $(\vec{v})_i$ denotes the ith component of $\vec{v}$.)

Theorem 5.12 (Seki et al. 1991)**.** *For any m-MCFL L, there exists a constant p satisfying the following property: for every $y \in L$ with $|y| \geq p$, there exist some $u_j \in \Sigma^*$ $(1 \leq j \leq m+1)$, $v_j, w_j, s_j \in \Sigma^*$ $(1 \leq j \leq m)$ which satisfy the following conditions:*

1. $\#(u_1 v_1 w_1 s_1 u_2 v_2 w_2 s_2 u_3 \ldots u_m v_m w_m s_m u_{m+1}) \geq \#(y)$,
2. $\sum_{j=1}^{m} |v_j s_j| > 0$,
3. $\sum_{j=1}^{m} |v_j w_j s_j| \leq p$,
4. *for any non-negative integer i,*

$$z_i = u_1 v_1^i w_1 s_1^i u_2 v_2^i w_2 s_2^i u_3 \ldots u_m v_m^i w_m s_m^i u_{m+1} \in L.$$

As for Groenink's (1997a) claim, it is possible to fix the flaw using Theorem 5.12.[15]

Exercise 5.8. Use Theorem 5.12 to show that the set of strings of the form (5.10) is not an MCFL.

Exercise 5.9. Show that the set J in (5.12) does not satisfy the condition in (5.14), but does satisfy the condition in Theorem 5.12.

[15]By the same method, the set of Grammatical word strings of German can also be shown to lie outside of the class of MCFLs (Groenink 1997b), even though the German word order exhibits nested rather than cross-serial dependencies.

Michaelis and Kracht (1997) took up a phenomenon called *Suffixaufnahme* (suffix-absorption) found in Old Georgian.[16] The following examples are originally from Boeder 1995:

(5.15) Davit-is galob-isa muql-ta ama-t ṭartkuma-j
David-Gen singing-Gen verse-Pl(Gen) Art-Pl(Gen) recitation-Nom
'the recitation of the verses of the song of David'

(5.16) tkuenda micemul ars cnob-ad saidumlo-j igi sasupevel-isa
to=you given is knowing-Adv mystery-Nom Art=Nom kingdom-Gen
m-is ɣmrt-isa-jsa-j
Art-Gen god-Gen-Gen-Nom
'Unto you it is given to know the mystery of the kingdom of God'

(5.17) q̇ovel-i igi sisxl-i saxl-isa-j m-is Saul-is-isa-j
all-Nom Art=Nom blood-Nom house-Gen-Nom Art-Gen Saul-Gen-Gen-Nom
'all the blood of the house of Saul'

In addition to the basic word order found in (5.15), nouns in the genitive construction can also appear in reverse order, in which case lower nouns can inherit case suffixes from higher nouns, as seen in (5.16) and (5.17). Thus, complex nominative NPs with k stacked NPs may have the following form:

(5.18) N_1-Nom N_2-Gen-Nom N_3-Gen^2-Nom . . . N_k-Gen^{k-1}-Nom

Moreover, Michaelis and Kracht (1997) states that the following condition holds:

(5.19) The number of all genitive suffixes of all nouns within a complex NP consisting of k stacked NPs, where $k \in \mathbb{N}_+$, is bounded by $\sum_{i=1}^{k-1} i = k(k-1)/2$.

Michaelis and Kracht (1997) used this construction to prove that the set L_G of grammatical strings of morphemes in Old Georgian is not semilinear.[17]

We can state their argument as follows. Pick a fixed enumeration $a_1, a_2, a_3, \ldots, a_l$ of Old Georgian morphemes so that

- a_1 is a fixed noun stem,
- a_2 is the genitive suffix,

[16]This phenomenon is also known to occur in some other Caucasian and ancient Middle Eastern languages as well as many Australian languages (Wikipedia entry "Suffixaufnahme"), but according to Michaelis and Kracht (1997), Old Georgian is the only language that allows for an unobunded iteration of Suffixaufnahme.

[17]Bhatt and Joshi (2004) dispute Michaelis and Kracht's (1997) interpretation of the data and argue that Suffixaufnahme in Old Georgian only gives rise to a semilinear set.

- a_3 is a nominative suffix,
- a_4 is a genitive article,
- a_5 is a nominative article, and
- a_6 is a fixed intransitive verb.

Let $L_G \subseteq \{a_1, \ldots, a_l\}^*$ be the set of grammatical morpheme strings of Old Georgian, and consider the Parikh image $A = \#(L_G) \subseteq \mathbb{N}^l$. Suppose that A is semilinear. Since semilinear sets are closed under selection,

$$A_1 = \{\, \vec{v} \in A \mid (\vec{v})_5 = (\vec{v})_6 = 1 \text{ and } (\vec{v})_i = 0 \text{ for all } i \geq 7 \,\}$$

is a semilinear set. Since semilinear sets are closed under projection,

$$A_2 = \{\, ((\vec{v})_1, (\vec{v})_2) \mid \vec{v} \in A_1 \,\}$$

is also semilinear. By the above property of Suffixaufnahme in Old Georgian, it holds that for each $k \geq 1$,

$$\max\{\, y \mid (k, y) \in A_2 \,\} = \frac{k(k-1)}{2}.$$

This contradicts the following lemma. Therefore, the set of grammatical morpheme strings of Old Georgian is not semilinear.

Lemma 5.13. *Let $A \subseteq \mathbb{N} \times \mathbb{N}$ be a semilinear set. Suppose that for each $x \in \mathbb{N}$, the set $\{\, y \mid (x, y) \in A \,\}$ is finite. Then there exist constants c, d such that $(x, y) \in A$ implies $y \leq cx + d$.*

Proof. Clearly, it suffices to consider the case where A is a linear set, so suppose

$$A = \{\, (u_0, v_0) + \sum_{i=1}^{m} c_i \cdot (u_i, v_i) \mid c_1, \ldots, c_m \in \mathbb{N} \,\},$$

where $(u_0, v_0) \in \mathbb{N} \times \mathbb{N}$ and $(u_i, v_i) \in \mathbb{N} \times \mathbb{N} - \{(0, 0)\}$ for $i = 1, \ldots, m$. Since $\{\, y \mid (x, y) \in A \,\}$ is finite, $u_i > 0$ for each $i = 1, \ldots, m$. Let $d = v_0$ and $c = \max\{\, v_i \mid 1 \leq i \leq m \,\}$. We prove by induction on x that $(x, y) \in A$ implies $y \leq cx + d$. If $x = u_0$, then $y = v_0 = d \leq cx + d$. If $x > u_0$, then there must be an $i \geq 1$ such that $(x, y) - (u_i, v_i) \in A$. By induction hypothesis, we have $y - v_i \leq c(x - u_i) + d$. Then $y \leq c(x - u_i) + d + v_i \leq c(x - 1) + d + c = cx + d$. □

Exercise 5.10. Let $A \subseteq \mathbb{N}^l$ be a semilinear set. Show the following:

1. For every $d \in \mathbb{N}$ and every i such that $1 \leq i \leq l$, the set $\{\, \vec{v} \in A \mid (\vec{v})_i = d \,\}$ is semilinear.

2. $\{ ((\vec{v})_1, \ldots, (\vec{v})_{l-1}) \mid \vec{v} \in A \}$ is semilinear.

By a similar method, Michaelis and Kracht (1997) were able to show that the Dutch and Chinese fragments discussed by Groenink (1997a) and Radzinski (1991) are not semilinear. It follows that Chinese is not an MCFL; this reasoning remedies the flaw in Radzinski's (1991) proof.

Motivated by the Dutch and Chinese data, which show that k-pumpability cannot be maintained as a linguistic universal, Groenink (1997b,a) proposed to replace the constant growth property (or semilinearity) in the definition of mild context-sensitivity by the following condition, which he calls *finite pumpability*:

(5.20) **(Groenink)** A language L is *finitely pumpable* if there is a constant p such that for any $w \in L$ with $|w| \geq p$, there are a finite number k and strings $u_0, u_1, \ldots, u_k$ and $v_1, \ldots, v_k$ such that

- $w = u_0 v_1 u_1 v_2 u_2 \ldots u_{k-1} v_k u_k$,
- $1 \leq |v_i| \leq p$ for each i, and
- $u_0 v_1^n u_1 v_2^n u_2 \ldots u_{k-1} v_k^n u_k \in L$ for any $n \geq 0$.

Michaelis and Kracht (1997) note that the relevant fragment of Old Georgian may well be finitely pumpable.

Groenink (1997b) defined the class of *optimistically mildly context-sensitive* languages as the largest class of languages that (i) is a substitution-closed abstract family of languages, (ii) contains only finitely pumpable languages, and (iii) is included in P, and noted that "this class exists and subsumes ... MCFL". The second half of this claim was refuted 17 years later by Kanazawa et al. (2014), who showed that the language defined by the following 3-MCFG is not finitely pumpable:[18]

$$\begin{aligned} S(\boldsymbol{x}_2) &\leftarrow J(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3), \\ J(\mathtt{a}\boldsymbol{x}_1, \boldsymbol{y}_1 \mathtt{c} \boldsymbol{x}_2 \bar{\mathtt{c}} \mathtt{d} \boldsymbol{y}_2 \bar{\mathtt{d}} \boldsymbol{x}_3, \boldsymbol{y}_3 \mathtt{b}) &\leftarrow J(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3), J(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3), \\ J(\mathtt{a}, \varepsilon, \mathtt{b}) &\leftarrow . \end{aligned}$$

Exercise 5.11. Is $\{ a^n b^m \mid n \geq 0, n \leq m \leq n(n-1)/2 \}$ finitely pumpable? Does it have the constant growth property?

Parallel multiple context-free grammars

A grammar formalism that is capable of generating non-semilinear sets of strings is the *parallel multiple context-free grammar* (PMCFG), also introduced by Seki

[18] The first rule of this MCFG is deleting. I leave it as an exercise for the reader to convert this MCFG into a non-deleting one.

et al. (1991). A PMCFG is just like an MCFG except that the condition (e) in the definition of rules is dropped; each variable $\boldsymbol{x}_{i,j}$ is allowed to occur any number of times in the string $\alpha_1 \ldots \alpha_r$. The language generated by a PMCFG is a *parallel multiple context-free language* (PMCFL).

For example, the following PMCFG generates $\{\, \mathsf{a}^{2^n} \mid n \geq 0 \,\}$:[19]

$$\begin{aligned} S(\mathsf{a}) &\leftarrow , \\ S(\boldsymbol{x}\boldsymbol{x}) &\leftarrow S(\boldsymbol{x}). \end{aligned}$$

The next example generates $\{\, \mathsf{a}^{n^2} \mid n \geq 0 \,\}$:

$$\begin{aligned} S(\varepsilon) &\leftarrow , \\ S(\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_2\mathsf{a}) &\leftarrow A(\boldsymbol{x}_1, \boldsymbol{x}_2), \\ A(\varepsilon, \varepsilon) &\leftarrow , \\ A(\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_2\mathsf{a}, \boldsymbol{x}_2\mathsf{a}) &\leftarrow A(\boldsymbol{x}_1, \boldsymbol{x}_2). \end{aligned}$$

Groenink (1997a) gave the following toy PMCFG to describe the Dutch clauses of the form (5.10):

$$\begin{aligned} \mathrm{S}(\textsf{dat Jan Piet Marie } \boldsymbol{x}_2\ \boldsymbol{x}_1 \textsf{ en zag } \boldsymbol{x}_3 \textsf{ omhelzen}) &\leftarrow \mathrm{Cj}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3), \\ \mathrm{Cj}(\textsf{hoorde } \boldsymbol{x}_3 \textsf{ uitnodigen } \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) &\leftarrow \mathrm{Cj}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3), \\ \mathrm{Cj}(\textsf{hoorde } \boldsymbol{x}_2 \textsf{ uitnodigen}, \boldsymbol{x}_1, \boldsymbol{x}_2) &\leftarrow \mathrm{Fl}(\boldsymbol{x}_1, \boldsymbol{x}_2), \\ \mathrm{Fl}(\textsf{Fred } \boldsymbol{x}_1, \textsf{leren } \boldsymbol{x}_2) &\leftarrow \mathrm{Fl}(\boldsymbol{x}_1, \boldsymbol{x}_2), \\ \mathrm{Fl}(\varepsilon, \varepsilon) &\leftarrow . \end{aligned}$$

The next PMCFG was given by Clark and Yoshinaka (2014) to describe the set J in (5.12):

$$\begin{aligned} S(\textsf{wu } \boldsymbol{x}_1\ \boldsymbol{x}_2) &\leftarrow A(\boldsymbol{x}_1, \boldsymbol{x}_2), \\ A(\textsf{zhao } \boldsymbol{x}_1, \boldsymbol{x}_2) &\leftarrow A(\boldsymbol{x}_1, \boldsymbol{x}_2), \\ A(\textsf{zhao } \boldsymbol{x}_1, \textsf{wu } \boldsymbol{x}_1\ \boldsymbol{x}_2) &\leftarrow A(\boldsymbol{x}_1, \boldsymbol{x}_2), \\ A(\varepsilon, \varepsilon) &\leftarrow . \end{aligned}$$

[19] According to Kobele (2006), a non-semilinear set like this can arise with the verbal relative clause construction in the Nigerian language Yoruba, which may involve recursive copying. The situation is analogous to the following (putative) case of recursive copying in the English "X or no X" construction:

- w = war or no war, I'm joining the army
- claim that w or no claim that w, he's not joining the army.

Exercise 5.12. Give a PMCFG for Old Georgian NPs of the form (5.18), using $\mathtt{a}_1, \ldots, \mathtt{a}_n$ for N, g for Gen, and n for Nom.

Ljunglöf (2004) showed that the *Grammatical Framework* (Ranta 2011), which is a programming language for writing multilingual grammars, is equivalent to PMCFGs in its power of describing string sets. Like MCFLs, all PMCFLs belong to the complexity class LOGCFL, the closure of CFL under logspace reducibility.[20]

Problems

5.1. Prove Theorem 5.1.

5.2. Prove that for every well-nested m-MCFG G, there exists a well-nested m-MCFG G' each of whose rules has at most 2 subgoals.

5.3. Give a well-nested MCFG generating RESP. (Note that there is no well-nested 2-MCFG generating RESP.)

5.4. Show that the class of languages generated by m-MCFGs is closed under intersection with regular sets. Show the same for well-nested m-MCFGs.

5.5. Prove Theorem 5.3 by rigorously defining the functions $g_{\alpha,p_1,\ldots,p_n}$ and $h_{\alpha,p_1,\ldots,p_n}$ used to construct the well-nested 2-MCFG equivalent to a given TAG.

5.6. Show that $\{ (\mathtt{ab}^n)^m \mid m, n \geq 1 \}$ is not semilinear; i.e., its Parikh image $\{ (m, mn) \mid m, n \geq 1 \}$ is not a semilinear subset of $\mathbb{N}^2$. (*Hint:* Show that for any linear set $A \subseteq \mathbb{N}^2$, if $\{ y \mid (x, y) \in A \}$ is infinite for some x, then there is some k such that $(x, y) \in A$ implies $(x, y + k) \in A$.)

5.7. Show that 1-pumpability does not imply the constant growth property.

5.8. Show that semilinearity does not imply finite pumpability.

References

Bhatt, Rajesh and Aravind Joshi. 2004. Semilinearity is a syntactic invariant: A reply to Michaelis and Kracht. *Linguistic Inquiry* 35:683–692.

Boeder, Winfried. 1995. Suffixaufnahme in Kartvelian. In F. Plank, ed., *Double Case: Agreement by Suffixaufnahme*, pages 151–215. New York: Oxford University Press.

[20]The inclusion in P was shown by Kaji et al. (1992).

Clark, Alexander and Ryo Yoshinaka. 2014. Distributional learning of parallel multiple context-free grammars. *Machine Learning* 96(1–2):5–31.

de Groote, Philippe and Sylvain Pogodalla. 2004. On the expressive power of abstract categorial grammars: Representing context-free formalisms. *Journal of Logic, Language and Information* 13(4):421–438.

Engelfriet, Joost. 1997. Context-free graph grammars. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages, Vol. 3: Beyond Words*, pages 125–213. Berlin: Springer-Verlag.

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

Engelfriet, Joost, Grzegorz Rozenberg, and Giora Slutzki. 1980. Tree transducers, L systems, and two-way machines. *Journal of Computer and System Sciences* 20:150–202.

Greibach, S. A. 1978. One-way finite visit automata. *Theoretical Computer Science* 6:175–221.

Groenink, Annius V. 1997a. Mild context-sensitivity and tuple-based generalizations of context-free grammar. *Linguistics and Philosophy* 20:607–636.

Groenink, Annius V. 1997b. *Surface without Structure*. Ph.D. thesis, University of Utrecht.

Harkema, Henk. 2001. A characterization of minimalist languages. In P. de Groote, G. Morrill, and C. Retoré, eds., *Logical Aspects of Computational Linguistics: 4th International Conference, LACL 2001*, pages 193–211. Berlin: Springer.

Hopcroft, John E. and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Reading, Massachusetts: Addison-Wesley.

Hotz, Günter and Gisela Pitsch. 1996. On parsing coupled-context-free languages. *Thoretical Computer Science* 161:205–253.

Joshi, Aravind K. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D. R. Dowty, L. Karttunen, and A. M. Zwicky, eds., *Natural Language Parsing: Psychological, Computational and Theoretical Perspectives*, pages 206–250. Cambridge: Cambridge University Press.

Joshi, Aravind K., K. Vijay-Shanker, and David Weir. 1991. The convergence of mildly context-sensitive grammar formalisms. In P. Sells, S. Shieber, and T. Wasow, eds., *Processing of Linguistic Structure*, pages 31–81. Cambridge, Massachusetts: MIT Press.

Kaji, Yuichi, Ryuichi Nakanishi, Hiroyuki Seki, and Tadao Kasami. 1992. Universal recognition problems for parallel multiple context-free grammars and for their subclasses. *IEICE Transactions on Information and Systems* E75-D(4):499–508.

Kanazawa, Makoto, Gregory M. Kobele, Jens Michaelis, Sylvain Salvati, and Ryo Yoshinaka. 2014. The failure of the strong pumping lemma for multiple context-free languages. *Theory of Computing Systems* 55(1):250–278.

Kanazawa, Makoto and Sylvain Salvati. 2007. Generating control languages with abstract categorial grammars. In *FG-2007: The 12th Conference on Formal Grammar*.

Kanazawa, Makoto and Sylvain Salvati. 2012. MIX is not a tree-adjoining language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 664–674.

Kasami, Tadao, Hiroyuki Seki, and Mamoru Fujii. 1987. Generalized context-free grammars, multiple context-free grammars and head grammars. Tech. rep., Osaka University.

Kobele, Gregory Michael. 2006. *Generating Copies: An investigation into structural identity in language and grammar*. Ph.D. thesis, University of California Los Angeles.

Kozen, Dexter C. 1997. *Automata and Computability*. New York: Springer.

Kracht, Marcus. 2003. *The Mathematics of Language*. Berlin: Mouton de Gruyter.

Ljunglöf, Peter. 2004. *Expressivity and Complexity of the Grammatical Framework*. Ph.D. thesis, Chalmers University of Technology and Göteborg University.

Manaster-Ramer, Alexis. 1987. Dutch as a formal language. *Linguistics and Philosophy* 10:221–246.

Michaelis, Jens. 2001a. Derivational minimalism is mildly context-sensitive. In M. Moortgat, ed., *Logical Aspects of Computational Linguistics: Third International Conference (LACL'98)*, pages 179–198. Berlin: Springer.

Michaelis, Jens. 2001b. *On Formal Properties of Minimalist Grammars*, vol. 13. Universitätsbibliothek, Publikationsstelle. ISBN 3-935024-28-2.

Michaelis, Jens. 2001c. Transforming linear context-free rewriting systems into minimalist grammars. In P. de Groote, G. Morrill, and C. Retoré, eds., *Logical Aspects of Computational Linguistics: 4th International Conference, LACL 2001*, pages 228–244. Berlin: Springer.

Michaelis, Jens and Marcus Kracht. 1997. Semilinearity as a syntactic invariant. In C. Retoré, ed., *Logical Aspects of Computational Linguistics*, pages 329–345. Berlin: Springer.

Nederhof, Mark-Jan. to appear. A short proof that O_2 is an MCFL. In *Proceedings of the 54th Meeting of the Association for Computational Linguistics*.

Parikh, Rohit J. 1966. On context-free languages. *Journal of the Association for Computing Machinery* 13(4):570–581.

Pollard, Carl J. 1984. *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language*. Ph.D. thesis, Department of Linguistics, Stanford University.

Pullum, Geffrey K. and Gerald Gazdar. 1982. Natural languages and context-free languages. *Linguistics and Philosophy* 4(4):471–504.

Radzinski, Daniel. 1991. Chinese number-names, tree adjoining languages, and mild context-sensitivity. *Computational Linguistics* 17(3):277–299.

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

Ranta, Aarne. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. Stanford, Calif.: CSLI Publications.

Rich, Elaine. 2007. *Automata, Computability, and Complexity: Theory and Applications*. Upper Saddle River, NJ: Pearson Prentice Hall.

Roach, Kelly. 1987. Formal properties of head grammars. In A. Manaster-Ramer, ed., *Mathematics of Language*, pages 293–347. Amsterdam: John Benjamins.

Salvati, Sylvain. 2007. Encoding second order string ACG with deterministic tree walking transducers. In S. Wintner, ed., *Proceedings of FG 2006: The 11th conference on Formal Grammar*, FG Online Proceedings, pages 143–156. CSLI Publications.

Salvati, Sylvain. 2011. MIX is a 2-MCFL and the word problem in $\mathbb{Z}^2$ is captured by the IO and the OI hierarchies. Tech. rep., INRIA.

Salvati, Sylvain. 2015. MIX is a 2-MCFL and the word problem in $\mathbb{Z}^2$ is captured by the IO and the OI hierarchies. *Journal of Computer and System Sciences* 81(7):1252–1277.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88(2):191–229.

Steedman, Mark. 2000. *The Syntactic Process*. Cambridge, Massachusetts: MIT Press.

Vijay-Shanker, K. and D. J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory* 27:511–546.

Vijay-Shanker, K., David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Vijayashanker, K. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, University of Pennsylvania.

Villemonte de la Clergerie, Éric. 2002a. Parsing MCS languages with thread automata. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pages 101–108.

Villemonte de la Clergerie, Éric. 2002b. Parsing mildly context-sensitive languages with thread automata. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7.

Weir, David J. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.

Weir, David J. 1992. Linear context-free rewriting systems and deterministic tree-walking transducers. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 136–143.