

## Part II: Mildly context-sensitive grammars and extensions of TAG

### 8 Mildly context-sensitive grammars and linear context-free rewriting systems

Idea of mild context-sensitivity (Joshi 1985): characterize the amount of context-sensitivity necessary for natural languages, the corresponding class of languages are called *mildly context-sensitive*.

#### 8.1 Definition of mild context-sensitivity

Joshi (1985) proposes three properties for grammar formalisms that are adequate for natural languages:

1. Limited cross-serial dependencies: There is a  $n \geq 2$  such that the formalism can generate all string languages  $\{w^n \mid w \in T^*\}$ .
2. The formalism is polynomially parsable.
3. The string languages generated by the formalism have the *constant growth property*.

Constant growth property: the length of the words generated by the grammar grows in a linear way, e.g.,  $\{a^{2^n} \mid n \geq 0\}$  does not have that property.

**Definition 12 (Constant Growth Property)** *Let  $X$  be an alphabet and  $L \subseteq X^*$ .  $L$  has the constant growth property iff there is a constant  $c_0 > 0$  and a finite set of constants  $C \subset \mathbb{N} \setminus \{0\}$  such that for all  $w \in L$  with  $|w| > c_0$ , there is a  $w' \in L$  with  $|w| = |w'| + c$  for some  $c \in C$ .*

**Proposition 19** *TAGs are mildly context-sensitive.*

Proof:

1. Limited cross-serial dependencies: TAG satisfies this condition for  $n = 2$ .
2. Polynomial parsing: TAGs can be parsed in  $O(n^6)$  (Joshi and Schabes 1997; Schabes and Joshi 1988).
3. Constant growth property: Follows from the pumping lemma.

□

In general, one can show the constant growth property of a class of languages by showing the semi-linearity (Parikh 1966) of the languages. Constant growth follows from semilinearity.

We define for  $\langle a_1, \dots, a_n \rangle, \langle b_1, \dots, b_n \rangle \in \mathbb{N}^n$  and  $m \in \mathbb{N}$ :  $\langle a_1, \dots, a_n \rangle + \langle b_1, \dots, b_n \rangle := \langle a_1 + b_1, \dots, a_n + b_n \rangle$  and  $m \langle a_1, \dots, a_n \rangle := \langle ma_1, \dots, ma_n \rangle$ .

A Parikh mapping is a function counting for each letter of an alphabet the occurrences of this letter in a word  $w$ :

**Definition 13 (Parikh mapping)** *Let  $X = \{a_1, \dots, a_n\}$  be an alphabet with some (arbitrary) fixed order of the elements. The Parikh mapping  $p : X^* \rightarrow \mathbb{N}^n$  (wrt this order) is defined as follows:*

- For all  $w \in X^* : p(w) := \langle |w|_{a_1}, \dots, |w|_{a_n} \rangle$  where  $|w|_{a_i}$  is the number of occurrences of  $a_i$  in  $w$ .

- For all languages  $L \subseteq X^* : p(L) := \{p(w) \mid w \in L\}$  is the Parikh image of  $L$ .

Two words are *letter equivalent* if they contain equal number of occurrences of each terminal symbol, and two languages are letter equivalent if every string in one language is letter-equivalent to a string in the other language and vice-versa.

**Definition 14 (Letter equivalent)** Let  $X$  be an alphabet.

1. Two words  $w_1, w_2 \in X^*$  are letter equivalent if there is a Parikh mapping  $p$  such that  $p(w_1) = p(w_2)$ .
2. Two languages  $L_1, L_2 \subseteq X^*$  are letter equivalent if there is a Parikh mapping  $p$  such that  $p(L_1) = p(L_2)$ .

**Definition 15 (Semilinear)** 1. Let  $x_0, \dots, x_m$  with  $m \geq 0$  be in  $\mathbb{N}^n$  for some  $n \geq 0$ .

The set  $\{x_0 + n_1x_1 + \dots + n_mx_m \mid n_i \in \mathbb{N} \text{ for } 1 \leq i \leq m\}$  is a linear subset of  $\mathbb{N}^n$ .

2. The union of finitely many linear subsets of  $\mathbb{N}^n$  is a semilinear subset of  $\mathbb{N}^n$ .
3. A language  $L \subseteq X^*$  is semilinear iff there is a Parikh mapping  $p$  such that  $p(L)$  is a semilinear subset of  $\mathbb{N}^n$  for some  $n \geq 0$ .

**Lemma 3** The constant growth property holds for semilinear languages.

Proof: Assume  $L \subseteq X^*$  is semilinear and  $p(L)$  is a semilinear Parikh image of  $L$  where  $p(L)$  is the union of the linear sets  $M_1, \dots, M_l$ . Then the constant growth property holds for  $L$  with

$$c_0 := \max\{\sum_{i=1}^n y_i \mid \begin{array}{l} \text{there are } x_1, \dots, x_m \text{ such that} \\ \langle y_1, \dots, y_n \rangle + n_1x_1 + \dots + n_mx_m \mid n_i \in \mathbb{N} \\ \text{is one of the sets } M_1, \dots, M_l \end{array}\}$$

and

$$C := \{\sum_{i=1}^n y_i \mid \begin{array}{l} \text{there are } x_1, \dots, x_m \text{ such that} \\ \langle x_1 + n_1\langle y_1, \dots, y_n \rangle + \dots + n_mx_m \mid n_i \in \mathbb{N} \\ \text{is one of the sets } M_1, \dots, M_l \end{array}\}$$

□

Each language that is letter equivalent to a semilinear language is semilinear as well since the Parikh images of the two languages are equal.

**Proposition 20 (Parikh-Theorem)** Each context-free language is semilinear.

Therefore, in order to show the semilinearity (and constant growth) of a language, it is sufficient to show letter equivalence to a context-free language.

**Exercise 15**

1. Show that the copy language  $\{ww \mid w \in T^*\}$  for some alphabet  $T$  is semilinear using the Parikh-Theorem.
2. Show that  $\{a^{2^n} \mid n \geq 0\}$  is not semilinear.

*Hint: if the language was semilinear it would satisfy the constant growth property. Show that this is not the case.*

## 8.2 Linear context-free rewriting systems (LCFRS)

LCFRS (Weir 1988): class of grammar formalisms.

Idea: set of grammars that have an underlying (generalized) context-free grammar. The context-free trees are terms, their interpretations are certain objects (e.g., strings, trees etc.). A set of yield functions maps the context-free trees to sequences of strings.

**Definition 16 (Generalized Context-Free Grammar)** A Generalized Context-Free Grammar (GCFG)  $G$  is a tuple  $G = \langle N, S, F, P \rangle$  where

- $N$  and  $F$  are disjoint alphabets, the nonterminals and the function symbols,
- $S \in N$  is the start symbol, and
- $P$  is a finite set of productions of the form  $A \rightarrow f(A_1, \dots, A_n)$  where  $n \geq 0, f \in F$  and  $A, A_1, \dots, A_n \in N$ .

A GCFG derives a set of terms:

- $A \Rightarrow f()$  if  $A \rightarrow f()$  is a production.
- $A \xRightarrow{*} f(t_1, \dots, t_n)$  if there is a production  $A \rightarrow f(A_1, \dots, A_n)$  and  $A_i \xRightarrow{*} t_i$  for  $1 \leq i \leq n$ .

The trees derived by a GCFG can be seen as derivation trees, i.e., as trees telling how to combine the elements of the grammar. These combinations lead to a larger structure (e.g., the derived tree in TAG). For a term  $t$ ,  $\llbracket t \rrbracket$  denotes the corresponding structure.

Examples:

### 1. CFG:

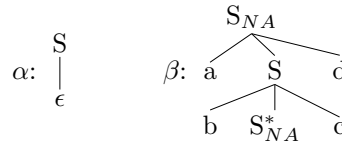
Take a CFG with  $S \rightarrow a, S \rightarrow SS$ . Corresponding GCFG:  $S \rightarrow f(X_a), S \rightarrow f(S, S), X_a \rightarrow f_a()$ . Corresponding structures (strings):  $\llbracket f_a() \rrbracket := a, \llbracket f(t_1) \rrbracket := \llbracket t_1 \rrbracket, \llbracket f(t_1, t_2) \rrbracket := \llbracket t_1 \rrbracket \circ \llbracket t_2 \rrbracket$ .

General construction:

- Replace each production  $A \rightarrow \alpha$  with  $A \rightarrow f(\alpha)$  while inserting commata between the symbols in  $\alpha$ .
- Replace each terminal  $a$  in these productions with a new nonterminal  $X_a$ , add productions  $X_a \rightarrow f_a()$ .
- Define  $\llbracket f_a() \rrbracket := a$ , and  $\llbracket f(t_1, \dots, t_n) \rrbracket := \llbracket t_1 \rrbracket \circ \dots \circ \llbracket t_n \rrbracket$ .

### 2. TAG (without substitution nodes)

Take the TAG for  $L_4$ :



Corresponding GCFG:

$\alpha \rightarrow f_\alpha(), \beta \rightarrow f_\beta()$  (no adjunctions), and  $\alpha \rightarrow f_{\alpha:\epsilon}(\beta), \beta \rightarrow f_{\beta:1}(\beta)$  (adjunctions of  $\beta$ ).

Start symbol is  $\alpha$ .

Corresponding structures:  $\llbracket f_\alpha() \rrbracket := \alpha, \llbracket f_\beta() \rrbracket := \beta, f_{\alpha:\epsilon}(t_1) := \alpha[\epsilon, \llbracket t_1 \rrbracket], f_{\beta:1}(t_1) := \beta[1, \llbracket t_1 \rrbracket]$ .<sup>13</sup>

General construction:

<sup>13</sup>As usual,  $\gamma[p, \gamma']$  is defined as follows: if  $\gamma'$  is (derived from) an initial tree and the node at position  $p$  in  $\gamma$  a substitution node, then  $\gamma[p, \gamma']$  is the tree one obtains by substitution of  $\gamma'$  into  $\gamma$  at node position  $p$ . If  $\gamma'$  is (derived from) an auxiliary tree and the node at position  $p$  in  $\gamma$  an internal node, then  $\gamma[p, \gamma']$  is the tree one obtains by adjunction of  $\gamma'$  to  $\gamma$  at node position  $p$ . Otherwise  $\gamma[p, \gamma']$  is undefined.

- For each elementary  $\gamma$  without OA constraints: there is a zero arity function  $f_\gamma$  and a production  $\gamma \rightarrow f_\gamma()$ , and  $\llbracket f_\gamma() \rrbracket := \gamma$ .
- For each  $\gamma$  and positions  $p_1, \dots, p_n$  in  $\gamma$  comprising all OA nodes and  $\gamma_1, \dots, \gamma_n$  that can be adjoined at positions  $p_1, \dots, p_n$  resp.: There is a  $n$ -ary function  $f_{\gamma:p_1, \dots, p_n}$ , a production  $\gamma \rightarrow f_{\gamma:p_1, \dots, p_n}(\gamma_1, \dots, \gamma_n)$ , and  $\llbracket f_{\gamma:p_1, \dots, p_n}(t_1, \dots, t_n) \rrbracket := \gamma[p_1, \llbracket t_1 \rrbracket] \dots [p_n, \llbracket t_n \rrbracket]$

In the case of the CFG the GCFG generates terms that are interpreted as strings while in the case of the TAG the interpretation of the terms generated by the GCFG are trees. Here, the terms correspond to the derivation trees while their denotations are the derived trees.

Apart from the structures (the interpretations of the terms), the yields of the terms also can be computed. The yield is a sequence of strings. A LCFRS is a GCFG together with equations defining the yield function.

**Definition 17 (Linear context-free rewriting system)** A linear context-free rewriting system (LCFRS) is a pair  $G = \langle C, E \rangle$  where

- $C$  is a GCFG, and
- $E$  is a set of equations defining yield functions such that the following holds:
  - the yield  $\phi(t)$  of each term  $t$  generated in  $C$  is a sequence of strings, and
  - a unique equation is associated with each production  $A \rightarrow f(A_1, \dots, A_n)$  in  $C$ . It describes how to compute the yield of a term  $f(t_1, \dots, t_n)$  from the yields of  $t_1, \dots, t_n$  and a bounded collection of new terminals.  $\phi$  does neither copy nor erase (i.e., apart from the bounded number of new terminals, each string in the  $\phi(t_1), \dots, \phi(t_n)$  occurs exactly once in  $\phi(f(t_1, \dots, t_n))$ ).

If  $k$  is the maximal number of components in the yield of any nonterminal of  $G$ ,  $G$  is called an LCFRS( $k$ ) grammar.

Examples:

1. CFG:

Take example from above:  $\phi(f_a()) := a, \phi(f(t_1)) := \phi(t_1), \phi(f(t_1, t_2)) := \phi(t_1) \circ \phi(t_2)$ .

In general: for a GCFG corresponding to a CFG, the yield of a left hand side is the concatenation of the yields of the right hand side. CFG is an LCFRS(1).

2. TAG (without substitution nodes):

Example of GCFG for  $L_4$  from above.

$\phi(f_\alpha()) := \epsilon, \phi(f_\beta()) := \langle ab, cd \rangle$ .

$\phi(f_{\alpha;\epsilon}(t)) := \langle w_1 w_2 \rangle$  where  $\langle w_1, w_2 \rangle = \phi(t)$ , and  $\phi(f_{\beta;1}(t)) := \langle aw_1 b, cw_2 d \rangle$  where  $\langle w_1, w_2 \rangle = \phi(t)$ .

In general:

The yields of initial trees have one component, the yields of auxiliary trees have two, i.e., TAG is an LCFRS(2).

- For each production  $\gamma \rightarrow f_\gamma()$ :
  - If  $\gamma$  is an initial tree, then  $\phi(f_\gamma()) := \langle w \rangle$  where  $w$  is the yield of the tree  $\gamma$ .
  - If  $\gamma$  is an auxiliary tree, then  $\phi(f_\gamma()) := \langle w_1, w_2 \rangle$  where  $w_1$  and  $w_2$  are the parts of the yield of  $\gamma$  that are on the left and on the right of the foot node.
- For each production  $\gamma \rightarrow f_{\gamma:p_1, \dots, p_n}(\gamma_1, \dots, \gamma_n)$ :
  - Assume that  $\langle w_i^l, w_i^r \rangle$  is the sequence of strings for  $\gamma_i, 1 \leq i \leq n$ .
  - Define a decoration string  $\sigma_{\gamma:p_1, \dots, p_n}$  as follows: for  $1 \leq i \leq n$  the node at position  $p_i$  in  $\gamma$  is equipped with a left and a right string, namely  $w_i^l$  and  $w_i^r$ ;  $\sigma_{\gamma:p_1, \dots, p_n}$  is then obtained collecting

in a top-down-left-to-right traversal of  $\gamma$  the left strings of the  $p_i$  during top-down traversal, the terminals while visiting the (non foot node) leaves and the right strings during bottom-up traversal.

If  $\gamma$  is an initial tree, then  $\phi(f_{\gamma:p_1,\dots,p_n}(\gamma_1, \dots, \gamma_n)) := \langle \sigma_{\gamma:p_1,\dots,p_n} \rangle$ .

If  $\gamma$  is an auxiliary tree and if  $\sigma_{\gamma:p_1,\dots,p_n}^l$  and  $\sigma_{\gamma:p_1,\dots,p_n}^r$  are the parts of the decoration string obtained before/after visiting the foot node respectively, then  $\phi(f_{\gamma:p_1,\dots,p_n}(\gamma_1, \dots, \gamma_n)) := \langle \sigma_{\gamma:p_1,\dots,p_n}^l, \sigma_{\gamma:p_1,\dots,p_n}^r \rangle$ .

The language of an LCFRS is defined as follows:  $L(C, E) := \{w_1 \dots w_n \mid S \xRightarrow{*} t \text{ in } C \text{ and } \phi(t) = \langle w_1, \dots, w_n \rangle\}$ .

**Proposition 21** *If  $L$  is a language generated by a LCFRS  $\langle C, E \rangle$ , then  $L$  is semilinear.*

Proof: We construct a letter-equivalent CFG  $G$ .

- for all productions  $A \rightarrow f()$  in  $C$ : if  $\langle w_1, \dots, w_m \rangle$  is the string sequence defined by  $E$  for this production, then the CFG has a production

$$A \rightarrow w_1 \dots w_m$$

- for all productions  $A \rightarrow f(A_1, \dots, A_n)$  in  $C$ : if  $w$  is the bounded number of new symbols introduced in the computation of the yield of  $A \rightarrow f(A_1, \dots, A_n)$ , then the CFG has a production

$$A \rightarrow A_1 \dots A_n w$$

By induction on the length of derivations the following can be shown: there are  $w_1, \dots, w_n$  with  $S \xRightarrow{*} \langle w_1, \dots, w_n \rangle$  in  $\langle C, E \rangle$  iff there is a  $w$  letter equivalent to  $w_1 \dots w_n$  with  $S \xRightarrow{*} w$  in  $G$ .

□

LCFRS are weakly equivalent to simple Range Concatenation Grammars (RCG, Boullier 1999, 2000) that are polynomially parsable since RCGs in general (more powerful than simple RCG) generate exactly the class of all polynomially parsable languages (Bertsch and Nederhof 2001). Consequently:

**Proposition 22** *If  $L$  is a language generated by a LCFRS, then  $L$  can be recognized in polynomial time.*

Therefore LCFRSs are mildly context-sensitive.

**Exercise 16** *Give the corresponding LCFRS for the TAG for the copy language:*

