Mildly Context-Sensitive Grammar
Formalisms:
LMG and RCG

Laura Kallmeyer
Heinrich-Heine-Universität Düsseldorf
Sommersemester 2011

Overview
1. Introduction
2. Literal Movement Grammar
3. Range Concatenation Grammar

Introduction (1)
- So far, when dealing with LCFRS, we required the grammar to be linear:
  Every variable in the left-hand side of a rule appears exactly once in its right-hand side and vice versa.
- If we drop this constraint, we obtain more general grammars.
- So far, it did not matter whether we instantiated the variables with strings or ranges.
- If we drop linearity, this difference matters.
  In Literal Movement Grammars, variables are instantiated with strings, while in Range Concatenation Grammars, variables are instantiated with ranges. In the latter, all range concatenations must be possible with respect to some word in the language.

Introduction (2)
Example:
- Grammar 1: rules
  \( S(aXb) \rightarrow B(XX), B(bX) \rightarrow B(X), B(\varepsilon) \rightarrow \varepsilon \)
  RCG string language: \( \{ab\} \)
  LMG string language: \( \{ab^k | k \geq 1\} \)
- Grammar 2: rules
  \( S(XY) \rightarrow A(X)bC(Y), A(ab) \rightarrow \varepsilon, C(b) \rightarrow \varepsilon, C(c) \rightarrow \varepsilon \)
  RCG string language: \( \{ab\} \)
  LMG string language: \( \{ab, ac\} \)
Introduction (3)

- Range Concatenation Grammars were inspired by Groenink’s work on LMG and were first defined in [Boullier, 1998a].

Grammar Formalisms 5 LMG and RCG

Literal Movement Grammars (1)

LMG variables are instantiated wrt. strings:

Definition 2 (LMG clause instantiation)
Let $G = \langle N, T, V, S, P \rangle$ be a LMG. For a rule $c = A(\alpha_0) \rightarrow A_1(\alpha_{i1}, \ldots, \alpha_{in}) \ldots A_m(\alpha_m)$ $\in P$, every function $f : \{x | x \in V, x \text{ occurs in } c \} \rightarrow T^*$ is an instantiation of $c$. We call $A(f(\alpha)) \rightarrow A_1(f(\alpha_{i1}), \ldots, A_m(f(\alpha_m)))$ then an instantiated clause where $f$ is extended as follows:

1. $f(\varepsilon) = \varepsilon$;
2. $f(t) = t$ for all $t \in T$;
3. $f(xy) = f(x)f(y)$ for all $x, y \in T^*$;
4. $f((\alpha_1, \ldots, \alpha_m)) = (f(\alpha_1), \ldots, f(\alpha_m))$ for all $(\alpha_1, \ldots, \alpha_m) \in \left((T \cup V)\right)^m, m \geq 1$.

The string language is defined as for LCFRSs.

Grammar Formalisms 7 LMG and RCG

Literal Movement Grammars (2)

Ex.: LMG for $L_{MIX} = \{w | w \in \{a, b, c\}^*, |w|_a = |w|_b = |w|_c\}$

$S(\varepsilon) \rightarrow \varepsilon$ 

$S(XaY) \rightarrow A(XY)$

$A(XbY) \rightarrow B(XY)$ 

$B(XcY) \rightarrow S(Y)$

Possible rule instantiations for $A(XbY) \rightarrow B(XY)$:

- $f(X) = ab, f(Y) = cc, \sim A(abcc) \rightarrow B(abcc)$
- $f(X) = cab, f(Y) = c, \sim A(cabc) \rightarrow B(cabc)$
- $f(X) = aaa, f(Y) = a, \sim A(aaaaba) \rightarrow B(aaaa)$ (this one gets never used in an actual derivation)

The string language is defined as for LCFRSs.
Literal Movement Grammars (3)

Definition 3 (LMG string language) Let $G = \langle N, T, V, S, P \rangle$ be a LMG.

1. The set $L_{\text{pred}}(G)$ of instantiated predicates $A(\vec{\tau})$ where $A \in N$ and $\vec{\tau} \in (T^*)^k$ for some $k \geq 1$ is defined by the following deduction rules:

- $A(\vec{\tau}) \rightarrow \varepsilon$ is an instantiated clause
- $A_1(\vec{\tau}_1) \ldots A_m(\vec{\tau}_m) \rightarrow A(\vec{\tau})$ is an instantiated clause

2. The string language of $G$ is

$$\{w \in T^* | S(w) \in L_{\text{pred}}(G)\}.$$

Grammar Formalisms 9  LMG and RCG

Literal Movement Grammars (4)

Ex: Derivation of $w = aabcbc$ (deduction of $S(aabcbc)$)

$$
\begin{align*}
S(\varepsilon) & \rightarrow \varepsilon & \quad & \quad S(\varepsilon) & \rightarrow B(\varepsilon) & \quad & B(XeY) \rightarrow S(XY) \\
B(\varepsilon) & \rightarrow A(X\varepsilon Y) & \quad & \quad A(bc) & \rightarrow S(abc) & \quad & S(XabcY) \rightarrow A(XY) \\
A(abc) & \rightarrow B(abc) & \quad & \quad B(abc) & \rightarrow S(abc) & \quad & S(XabcY) \rightarrow A(XY) \\
B(abc) & \rightarrow A(abc) & \quad & \quad A(abc) & \rightarrow S(abc) & \quad & S(XabcY) \rightarrow A(XY) \\
S(abc) & \rightarrow B(abc) & \quad & \quad B(abc) & \rightarrow S(abc) & \quad & S(XabcY) \rightarrow A(XY)
\end{align*}
$$

Grammar Formalisms 10  LMG and RCG

Literal Movement Grammars (5)

• The general definition of LMGs allows to have any combination of variables and terminals in the components of the left-hand side and the right-hand side of a clause.
• In particular, in the instantiated clauses, strings can be copied or deleted and we can combine strings into new strings.
• If all this is disallowed, we obtain LCFRSs.

Grammar Formalisms 11  LMG and RCG

Literal Movement Grammars (6)

Definition 4 (Linear Context-Free Rewriting Systems) A LMG is

• non-combinatorial if for every clause $c \in P$, all the arguments in the right-hand side of $c$ are single variables.
• bottom-up (top-down) linear if for every $c \in P$, no variable appears more than once in the left-hand (right-hand) side of $c$.
• linear if it is top-down and bottom-up linear.
• bottom-up (top-down) non-erasing if for every $c \in P$, each variable occurring in the right-hand (left-hand) side of $c$ occurs also in its left-hand (right-hand) side.
• non-erasing if it is top-down and bottom-up non-erasing.
• a Linear Context-Free Rewriting System (LCFRS) if it is non-combinatorial, linear and non-erasing.

Grammar Formalisms 12  LMG and RCG
Literal Movement Grammars (7)

Other interesting LMGs are parallel multiple context-free grammars (PMCFG) and simple LMGs:

Definition 5 (PMCFG) An LMG is a parallel multiple context-free grammar (PMCFG) if it is non-combinatorial, top-down non-erasing and top-down linear.

Ex.:

\[
S(XXX) \rightarrow A(X) \\
A(aX) \rightarrow A(X) \\
A(bX) \rightarrow A(X) \\
A(\varepsilon) \rightarrow A(\varepsilon)
\]

Literal Movement Grammars (8)

Definition 6 (Simple LMG) An LMG is simple if it is non-combinatorial, bottom-up non-erasing and bottom-up linear.

In other words, every variable in a clause occurs exactly once in its left-hand side. Furthermore, the right-hand side components are single variables.

Simple LMG for \(\{a^{2^n} \mid n \geq 0\}\):

\[
S(XX) \rightarrow S(X)eq(X,Y) \\
S(A) \rightarrow \varepsilon \\
eq(aX,aY) \rightarrow \eq(X,Y) \\
eq(a,a) \rightarrow \varepsilon
\]
**Literal Movement Grammars (11)**

Simple LMGs generate the entire set of all polynomial languages:

**Proposition 2** The set of string languages generated by simple LMGs is exactly the class PTIME, i.e., the class of all polynomial languages [Groenink, 1996].

PMCFGs are less powerful than simple LMGs. [Ljunglöf, 2005] extends PMCFG with intersection, which leads to a formalism equivalent to simple LMG.

**Proposition 3**

- For every PMCFG G, there is a simple LMG G' such that L(G) = L(G').
- There exists a simple LMG G such that there is no PMCFG G with L(G) = L(G').

---

**Range Concatenation Grammars (1)**

Now we keep the syntax of the clauses but instantiate variables with ranges with respect to a given string w. This leads to Range Concatenation Grammars (RCGs) [Boullier, 1998a, Boullier, 1998b, Boullier, 1999, Boullier, 2000].

Example:

\[
\begin{align*}
S(X) & \rightarrow M(X, X, X) \\
M(bX, Y, Z) & \rightarrow M(X, Y, Z) \\
M(X, aY, Z) & \rightarrow M(X, Y, Z) \\
M(X, Y, aZ) & \rightarrow M(X, X, Z) \\
M(aX, bY; cZ) & \rightarrow M(X, Y, Z) \\
M(\varepsilon, \varepsilon, \varepsilon) & \rightarrow \varepsilon
\end{align*}
\]

\[L(G) = MIX = \{ \text{w} | \text{w} \in \{a, b, c\}^*, |\text{w}|_a = |\text{w}|_b = |\text{w}|_c \}\]

---

**Range Concatenation Grammars (2)**

**Definition 7** (Clause instantiation) Let G = (N, T, V, P, S) be a RCG. For a given clause \( c = A_0(\vec{a}_0) \rightarrow A_1(\vec{a}_1) \cdots A_m(\vec{a}_m) \) (0 ≤ m) and a string \( w = t_1 \cdots t_n \), the set of string languages generated by simple LMGs is exactly the class PTIME, i.e., the class of all polynomial languages.

1. an instantiation of \( c \) with respect to \( w \) consists of a function \( f : \{ t' | t' \text{ is an occurrence of some } t \in T \text{ in the clause} \} \cup V \cup \{ \text{Eps}_{i,j} \ | 1 \leq i \leq m, 1 \leq j \leq \text{dim}(A_i), \vec{a}_i(j) = \varepsilon \} \rightarrow \{ (i, j) | i \leq j, i, j \in \mathbb{N} \} \) such that:
   - a) for all occurrences \( t' \) of a \( t \in T \) in the clause, \( f(t') := (i, i + 1) \) for some \( i, 0 \leq i < n \) such that \( t_i \neq t \),
   - b) for all \( X \in V \), \( f(X) = (j, k) \) for some \( 0 \leq j \leq k \leq n \),
   - c) for all \( x, y \) adjacent in one of the elements of \( \vec{a}_i \) (0 ≤ i ≤ m), there are \( i, j, r \) with \( f(x) = (i, j), f(y) = (j, r) \); we then define \( f(xy) = (i, r) \),
   - d) for all

\[\text{Eps} \in \{ \text{Eps}_{i,j} \ | 1 \leq i \leq m, 1 \leq j \leq \text{dim}(A_i), \vec{a}_i(j) = \varepsilon \}, \text{ there is a } k, 0 \leq k \leq n \text{ with } f(\text{Eps}) = (k, k) \text{; we then define for every } \varepsilon \text{-argument } \vec{a}_i(j) \text{ that } f(\vec{a}_i(j)) = f(\text{Eps}_{i,j}). \]

2. if \( f \) is an instantiation of \( c \) with respect to \( w \), then \( A_0(f(\vec{a}_0)) \rightarrow A_1(f(\vec{a}_1)) \cdots A_m(f(\vec{a}_m)) \) is an instantiated clause where \( f(x_1, \ldots, x_k) = \langle f(x_1), \ldots, f(x_k) \rangle \).
Range Concatenation Grammars (3)

In each RCG derivation step, the left-hand side of an instantiated clause is replaced by its right-hand side. In other words, the set of instantiated rules with respect to some given $w$ is used as a CFG with start symbol $S(\langle\langle 0, |w|\rangle\rangle)$.

The string language of an RCG $G$ is

$$L(G) = \{ w \in T^* \mid S(\langle\langle 0, |w|\rangle\rangle) \Rightarrow \varepsilon \ \text{with respect to} \ w \}.$$ 

Ex.:

$w = abc$, RCG for the MIX language.

Derivation:

$$S(\langle\langle 0, 3\rangle\rangle) \rightarrow M(\langle\langle 0, 3\rangle, \langle 0, 3\rangle\rangle)$$
$$\rightarrow M(\langle\langle 0, 3\rangle, \langle 1, 3\rangle, \langle 0, 3\rangle\rangle)$$
$$\rightarrow M(\langle\langle 0, 3\rangle, \langle 1, 3\rangle, \langle 2, 3\rangle\rangle)$$
$$\rightarrow M(\langle\langle 1, 3\rangle, \langle 2, 3\rangle, \langle 3, 3\rangle\rangle)$$
$$\rightarrow M(\langle\langle 2, 3\rangle, \langle 2, 3\rangle, \langle 3, 3\rangle\rangle)$$
$$\rightarrow M(\langle\langle 3, 3\rangle, \langle 2, 3\rangle, \langle 3, 3\rangle\rangle)$$
$$\rightarrow M(\langle\langle 3, 3\rangle, \langle 3, 3\rangle, \langle 3, 3\rangle\rangle)$$
$$\rightarrow \varepsilon$$

Definition 8 (Simple Range Concatenation Grammar)

An RCG is simple if it is non-combinatorial, linear and non-erasing.

In the LCFRS/simple RCG case, it does not matter which string language definition we adopt, the result is the same.

**Proposition 4** LCFRS and simple RCG are equivalent.

Proposition 5

1. For any RCG, there is an equivalent non-combinatorial RCG.
2. For any non-combinatorial bottom-up erasing RCG, there is an equivalent non-combinatorial bottom-up non-erasing RCG.
3. For any non-combinatorial bottom-up non-erasing top-down erasing RCG, there is an equivalent non-combinatorial non-erasing RCG.

In other words, the possibilities of combinatorial clauses and erasing clauses do not increase the generative capacity of the grammar. The crucial property for RCG’s being more powerful than simple RCG is the possible non-linearity of the clauses.
Range Concatenation Grammars (7)

Proposition 6 The set of string languages generated by RCGs is exactly the class PTIME of all polynomial languages ([Bertsch and Nederhof, 2001]).

- The fact that every language generated by an RCG is polynomial is confirmed by the existence of polynomial parsing algorithms [Kallmeyer et al., 2009].
- The inclusion of all polynomial languages in the set of RCG string languages is shown in Appendix A of [Bertsch and Nederhof, 2001] by constructing an equivalent RCG for a given two-way alternating finite automaton with \( k \) heads. It is known that two-way alternating finite automata recognize exactly the class PTIME.

References


