Mildly Context-Sensitive Grammar Formalisms:

Linear Context-Free Rewriting Systems

Laura Kallmeyer
Heinrich-Heine-Universität Düsseldorf

Sommersemester 2011

Overview

1. Basic Ideas
2. LCFRS and CL
3. LCFRS and MCFG
4. LCFRS with Simple RCG syntax

Basic Ideas (1)

Linear Context-Free Rewriting Systems (LCFRS) can be conceived as a natural extension of CFG:

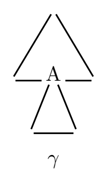
- In a CFG, non-terminal symbols A can span single strings, i.e., the language derivable from A is a subset of T^*.
- Extension to LCFRS: non-terminal symbols A can span tuples of (possibly non-adjacent) strings, i.e., the language derivable from A is a subset of $(T^*)^k$

$\Rightarrow$ LCFRS displays an extended domain of locality

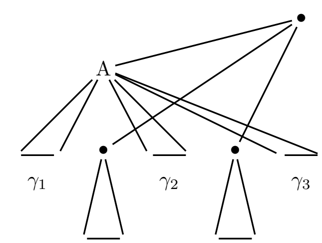
Basic Ideas (2)

Different spans in CFG and LCFRS:

CFG:



LCFRS:



Basic Ideas (3)

Example for a non-terminal with a yield consisting of 2 components:

$$yield(A) = \langle a^n b^n, c^n d^n \rangle, \text{ with } n \geq 1.$$

The rules in an LCFRS describe how to compute an element in the yield of the lefthand-side (lhs) non-terminal from elements in the yields of the right-hand side (rhs) non-terminals.

Ex.: $A(ab, cd) \rightarrow \varepsilon \qquad A(aXb, cYd) \rightarrow A(X, Y)$

The start symbol S is of dimension 1, i.e., has single strings as yield elements.

Ex.: $S(XY) \rightarrow A(X, Y)$

Language generated by this grammar (yield of S):
$\{a^n b^n c^n d^n \mid n \geq 1\}$.

Basic Ideas (4)

- In a CFG derivation tree (parse tree), dominance is determined by the relations between lhs symbol and rhs symbols of a rule.
- Furthermore, there is a linear order on the terminals and on all rhs of rules.

In an LCFRS, we can also obtain a derivation tree from the rules that have been applied:

- Dominance is also determined by the relations between lhs symbol and rhs symbols of a rule.
- There is a linear order on the terminals. BUT: there is no linear order on all rhs of rules.

  As a convention, we draw a non-terminal A left of a non-terminal B if the first terminal in the span of A precedes the first terminal in the span of B.

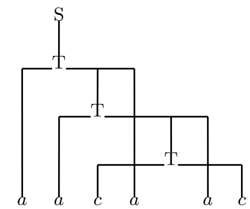
Basic Ideas (5)

Ex.: LCFRS for $\{wcwc \mid w \in \{a, b\}^*\}$:

$S(XY) \rightarrow T(X, Y) \qquad T(aY, aU) \rightarrow T(Y, U)$

$T(bY, bU) \rightarrow T(Y, U) \qquad T(c, c) \rightarrow \varepsilon$

Derivation tree for *aacaac*:



LCFRS and CL (1)

Interest of LCFRS for CL:

1. Data-driven parsing.
2. Mild context-sensitivity.
3. Equivalence with several important CL formalisms.

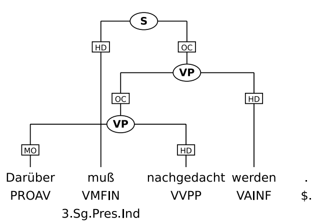
LCFRS and CL (2)

Data-driven parsing:

- Just like phrase structure trees (without crossing branches) can be described with CFG rules, trees with crossing branches can be described with LCFRS rules.
- Trees with crossing branches allow to describe discontinuous constituents, as for example in the Negra and Tiger treebanks.

LCFRS and CL (3)

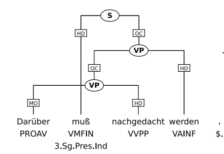
Example of a Negra tree with crossing branches:



LCFRS and CL (4)

Trees with crossing branches can be interpreted as LCFRS derivation trees.

⇒ an LCFRS can be straight-forwardly extracted from such treebanks. This makes LCFRS particularly interesting for data-driven parsing.



$$
\begin{aligned}
\text{PROAV(Darüber)} &\to \varepsilon\\
\text{VMFIN(muß)} &\to \varepsilon\\
\text{VVPP(nachgedacht)} &\to \varepsilon\\
\text{VAINF(werden)} &\to \varepsilon\\
\text{S}(X_1 X_2 X_3) &\to \text{VP}(X_1, X_3)\ \text{VMFIN}(X_2)\\
\text{VP}(X_1, X_2 X_3) &\to \text{VP}(X_1, X_2)\ \text{VAINF}(X_3)\\
\text{VP}(X_1, X_2) &\to \text{PROAV}(X_1)\ \text{VVPP}(X_2)
\end{aligned}
$$

LCFRS and CL (5)

Mild Context-Sensitivity:

- Natural languages are not context-free.
- Question: How complex are natural languages? In other words, what are the properties that a grammar formalism for natural languages should have?
- Goal: extend CFG only as far as necessary to deal with natural languages in order to capture the complexity of natural languages.

This effort has lead to the definition of *mild context-sensitivity* (Aravind Joshi).

LCFRS and CL (6)

A formalism is mildly context-sensitive if the following holds:

1. It generates at least all context-free languages.
2. It can describe a limited amount of crossing dependencies.
3. Its string languages are polynomial.
4. Its string languages are of constant growth.

LCFRS and CL (7)

- LCFRS are mildly context-sensitive.
- We do not have any other formalism that is also mildly context-sensitive and whose set of string languages properly contains the string languages of LCFRS.
- Therefore, LCFRS are often taken to provide a grammar-formalism-based characterization of mild context-sensitivity.

BUT: There are polynomial languages of constant growth that cannot be generated by LCFRS.

LCFRS and CL (8)

Equivalence with CL formalisms:

LCFRS are weakly equivalent to

- *set-local Multicomponent Tree Adjoining Grammar*, an extension of TAG that has been motivated by linguistic considerations;
- *Minimalist Grammar*, a formalism that was developed in order to provide a formalization of a GB-style grammar with transformational operations such as movement;
- *finite-copying Lexical Functional Grammar*, a version of LFG where the number of nodes in the c-structure that a single f-structure can be related with is limited by a grammar constant.

LCFRS and MCFG (1)

- *Multiple Context-Free Grammars (MCFG)* have been introduced by [Seki et al., 1991] while the equivalent *Linear Context-Free Rewriting Systems (LCFRS)* were independently proposed by [Vijay-Shanker et al., 1987].
- The central idea is to extend CFGs such that non-terminal symbols can span a tuple of strings that need not be adjacent in the input string.
- The grammar contains productions of the form $A_0 \to f[A_1, \ldots, A_q]$ where $A_0, \ldots, A_q$ are non-terminals and f is a function describing how to compute the yield of A_0 (a string tuple) from the yields of $A_1, \ldots, A_q$.
- The definition of LCFRS is slightly more restrictive than the one of MCFG. However, [Seki et al., 1991] have shown that the two formalisms are equivalent.

LCFRS and MCFG (2)

Example: MCFG/LCFRS for the double copy language.

Rewriting rules:

$S \to f_1[A] \quad A \to f_2[A] \quad A \to f_3[A] \quad A \to f_4[\,] \quad A \to f_5[\,]$

Operations:

$f_1[\langle X, Y, Z\rangle] = \langle XYZ\rangle \qquad f_4[\,] = \langle a, a, a\rangle$

$f_2[\langle X, Y, Z\rangle] = \langle aX, aY, aZ\rangle \qquad f_5[\,] = \langle b, b, b\rangle$

$f_3[\langle X, Y, Z\rangle] = \langle bX, bY, bZ\rangle$

LCFRS and MCFG (3)

Definition 1 (Multiple Context-Free Grammar) *A* multiple context-free grammar (MCFG) *is a 5-tuple* $\langle N, T, F, P, S\rangle$ *where*

- N *is a finite set of non-terminals, each* $A \in N$ *has a* fan-out $dim(A) \geq 1, dim(A) \in \mathbb{N}$*;*
- T *is a finite set of terminals;*
- F *is a finite set of mcf-functions;*
- P *is a finite set of rules of the form* $A_0 \to f[A_1, \dots, A_k]$ *with* $k \geq 0, f \in F$ *such that* $f : (T^*)^{dim(A_1)} \times \dots \times (T^*)^{dim(A_k)} \to (T^*)^{dim(A_0)}$*;*
- $S \in N$ *is the start symbol with* $dim(S) = 1$.

A MCFG with maximal non-terminal fan-out k *is called a* k*-MCFG.*

LCFRS and MCFG (4)

Mcf-functions are such that

- each component of the value of f is a concatenation of some constant strings and some components of its arguments.
- Furthermore, each component of the right-hand side arguments of a rule is not allowed to appear in the value of f more than once.

LCFRS and MCFG (5)

Definition 2 (mcf-function) f *is an mcf-function if there is a* $k \geq 0$ *and there are* $d_i > 0$ *for* $0 \leq i \leq k$ *such that* f *is a total function from* $(T^*)^{d_1} \times \dots \times (T^*)^{d_k}$ *to* $(T^*)^{d_0}$ *such that*

- *the components of* $f(\vec{x_1}, \dots, \vec{x_k})$ *are concatenations of a limited amount of terminal symbols and the components* x_{ij} *of the* $\vec{x_i}$ *(*$1 \leq i \leq k, 1 \leq j \leq d_i$*), and*
- *the components* x_{ij} *of the* $\vec{x_i}$ *are used at most once in the components of* $f(\vec{x_1}, \dots, \vec{x_k})$.

A LCFRS is a MCFG where the mcf-functions f are such that the the components x_{ij} of the $\vec{x_i}$ are used exactly once in the components of $f(\vec{x_1}, \dots, \vec{x_k})$.

LCFRS and MCFG (6)

- We can understand a MCFG as a generative device that specifies the yields of the non-terminals.
- The language of a MCFG is then the yield of the start symbol S.

Ex.: LCFRS for the double copy language.

$yield(A) = \{\langle w, w, w\rangle \mid w \in \{a, b\}^*\}$

$yield(S) = \{\langle www\rangle \mid w \in \{a, b\}^*\}$

LCFRS and MCFG (7)

Definition 3 (String Language of an MCFG/LCFRS)

Let $G = \langle N, T, F, P, S\rangle$ be a MCFG/LCFRS.

1. *For every $A \in N$:*
   - *For every $A \to f[\,] \in P$, $f(\,) \in yield(A)$.*
   - *For every $A \to f[A_1, \dots, A_k] \in P$ with $k \geq 1$ and all tuples $\tau_1 \in yield(A_1), \dots, \tau_k \in yield(A_k)$, $f(\tau_1, \dots, \tau_k) \in yield(A)$.*
   - *Nothing else is in $yield(A)$.*
2. *The string language of G is $L(G) = \{w \mid \langle w\rangle \in yield(S)\}$.*

LCFRS with Simple RCG syntax (1)

- *Range Concatentation Grammars (RCG)* and the restricted *simple RCG* have been introduced in [Boullier, 2000].
- Simple RCG are not only equivalent to MCFG and LCFRS but also represent a useful syntactic variant.

Example: Simple RCG for the double copy language.

$S(XYZ) \to A(X, Y, Z)$

$A(aX, aY, aZ) \to A(X, Y, Z)$

$A(bX, bY, bZ) \to A(X, Y, Z)$

$A(a, a, a) \to \varepsilon$

$A(b, b, b) \to \varepsilon$

LCFRS with Simple RCG syntax (2)

We redefine LCFRS with the simple RCG syntax:

Definition 4 (LCFRS) *A* LCFRS *is a tuple $G = (N, T, V, P, S)$ where*

1. *N, T and V are disjoint alphabets of non-terminals, terminals and variables resp. with a fan-out function dim: $N \to \mathbb{N}$.* *$S \in N$ is the start predicate; $dim(S) = 1$.*
2. *P is a finite set of rewriting rules of the form*

$$A_0(\vec{\alpha_0}) \to A_1(\vec{x_1}) \cdots A_m(\vec{x_m})$$

with $m \geq 0$, $\vec{\alpha_0} \in [(T \cup V)^*]^{dim(A_0)}$, $\vec{x_i} \in V^{dim(A_i)}$ for $1 \leq i \leq m$ and it holds that every variable $X \in V$ occurring in the rule occurs exactly once in the left-hand side and exactly once in the right-hand side.

LCFRS with Simple RCG syntax (3)

In order to apply a rule, we have to map variables to strings of terminals:

Definition 5 (LCFRS rule instantiation) *Let $G = \langle N, T, V, S, P \rangle$ be a LCFRS.*

For a rule $c = A(\vec{\alpha}) \rightarrow A_1(\vec{\alpha_1}) \dots A_m(\vec{\alpha_m}) \in P$, every function $f : \{x \,|\, x \in V, x \text{ occurs in } c\} \rightarrow T^*$ is an instantiation *of c.*

We call $A(f(\vec{\alpha})) \rightarrow A_1(f(\vec{\alpha_1})) \dots A_m(f(\vec{\alpha_m}))$ 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. $f(\langle \alpha_1, \dots, \alpha_m \rangle) = (\langle f(\alpha_1), \dots, f(\alpha_m) \rangle)$ *for all* $(\langle \alpha_1, \dots, \alpha_m \rangle) \in [(T \cup V)^*]^m$, $m \geq 1$.

LCFRS with Simple RCG syntax (4)

Definition 6 (LCFRS string language) *Let $G = \langle N, T, V, S, P \rangle$ be a LCFRS.*

1. *The set $L_{pred}(G)$ 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) $$\frac{}{A(\vec{\tau})} \quad A(\vec{\tau}) \rightarrow \varepsilon \text{ is an instantiated clause}$$

   b) $$\frac{A_1(\vec{\tau_1}) \dots A_m(\vec{\tau_m})}{A(\vec{\tau})} \quad \begin{array}{l} A(\vec{\tau}) \rightarrow A_1(\vec{\tau_1}) \dots A_m(\vec{\tau_m}) \\ \text{is an instantiated clause} \end{array}$$

2. *The string language of G is*

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

References

[Boullier, 2000] Boullier, P. (2000). Range Concatenation Grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT2000)*, pages 53–64, Trento, Italy.

[Seki et al., 1991] Seki, H., Matsumura, T., Fujii, M., and Kasami, T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

[Vijay-Shanker et al., 1987] Vijay-Shanker, K., Weir, D. J., and Joshi, A. K. (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of ACL*, Stanford.