Mildly Context-Sensitive Grammar Formalisms:

LCFRS: Relations to other Formalisms

Laura Kallmeyer

Sommersemester 2011

Overview

1. CFG and LCFRS
2. TAG and LCFRS
3. Set-local MCTAG and LCFRS
4. Minimalist Grammar and LCFRS
   (a) MG
   (b) MG and LCFRS
5. Other formalisms equivalent to LCFRS

CFG and LCFRS (1)

Every CFG is a simple 1-RCG and vice versa [Boullier, 2000], only with a slightly different syntax:

Construction of a 1-LCFRS for a given CFG: write every CFG production $A \to X_1 \dots X_k$ as a LCFRS rule $\overline{A}(X_1 \dots X_k) \to \gamma$ where γ is the concatenation of all $\overline{X_i}(X_i)$ where $1 \leq i \leq k$ and $X_i \in N$. The start predicate is $\overline{S}$.

	CFG:	1-LCFRS:
Example:	$S \to aSb$	$\overline{S}(aSb) \to \overline{S}(S)$
	$S \to \varepsilon$	$\overline{S}(\varepsilon) \to \varepsilon$

CFG and LCFRS (2)

Construction of a CFG for a given 1-LCFRS: write every rule $A(\alpha) \to A_1(X_1) \dots A_k(X_k)$ as a CFG rule $A \to f(\alpha)$ where f is a homomorphism with $f(a) = a$ for all $a \in T$ and $f(X_i) = A_i$ for all $1 \leq i \leq k$.

Example:

1-LCFRS:		CFG:	
$S(aXbY) \to S(X)A(Y)$	$S(\varepsilon) \to \varepsilon$	$S \to aSbA$	$S \to \varepsilon$
$A(cX) \to A(X)$	$A(\varepsilon) \to \varepsilon$	$A \to cA$	$A \to \varepsilon$

Proposition 1 *For a language L there is a CFG G with $L = L(G)$ iff there is a 1-LCFRS G' with $L = L(G')$.*

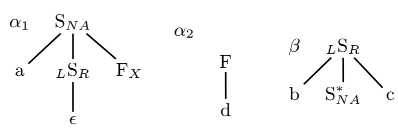
TAG and LCFRS (1)

General idea of the transformation of a TAG into an equivalent LCFRS [Boullier, 1998]:

- The LCFRS contains non-terminals $\langle\alpha\rangle(X)$ and $\langle\beta\rangle(L, R)$ for initial trees α and auxiliary trees β respectively.
- X covers the yield of α and all trees added to α, while L and R cover those parts of the yield of β (including all trees added to β) that are to the left and the right of the foot node of β.
- The rules reduce the components of these non-terminals by identifying those parts that come from the elementary tree α/β itself and those parts that come from one of the elementary trees added by substitution or adjunction.

TAG and LCFRS (2)

TAG:



Equivalent LCFRS:

$S(X) \rightarrow \langle\alpha_1\rangle(X) \mid \langle\alpha_2\rangle(X)$ $\quad\langle\alpha_1\rangle(aX) \rightarrow \langle\alpha_2\rangle(X)$

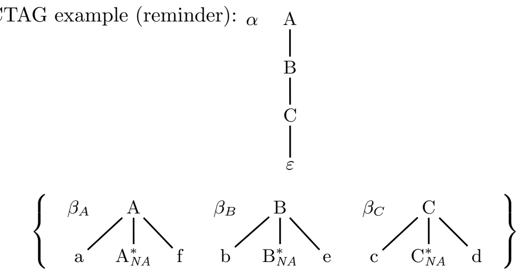
$\langle\alpha_1\rangle(aLRX) \rightarrow \langle\beta\rangle(L, R)\langle\alpha_2\rangle(X)$ $\quad\langle\beta\rangle(Lb, cR) \rightarrow \langle\beta\rangle(L, R)$

$\langle\alpha_2\rangle(d) \rightarrow \epsilon$ $\quad\langle\beta\rangle(b, c) \rightarrow \epsilon$

Proposition 2 *For every TAL L there is a 2-LCFRS G with $L = L(G)$.*

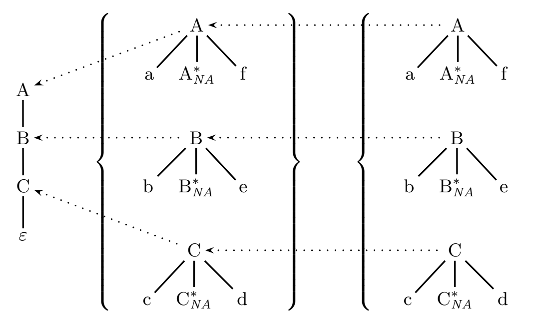
MCTAG and LCFRS (1)

MCTAG example (reminder):



MCTAG and LCFRS (2)

Derivation for *aabbccddeeff*:



MCTAG and LCFRS (3)

Proposition 3 *Set-local MCTAG and LCFRS are weakly equivalent [Weir, 1988].*

The constructions given below are not exactly the ones from [Weir, 1988].

MCTAG and LCFRS (4)

Construction of an equivalent LCFRS for a given MCTAG:

- We introduce non-terminals for all multicomponent sets. Their fan-out depends on the number of trees and whether they are initial or auxiliary: every initial tree contributes one component while every auxiliary tree contributes two components.
- For every set Γ and all sets $\Gamma_1, \dots, \Gamma_k$ that can attach to Γ such that all obligatory adjunctions and substitutions are performed, we introduce a rule that tells us how the yield of Γ can be obtained from the yields of $\Gamma_1, \dots, \Gamma_k$ and from the terminals occurring in Γ.
- For every set Γ without substitution nodes or OA constraints, we add a terminating rule that lists only the terminals occurring in Γ.

MCTAG and LCFRS (5)

Example: LCFRS that is equivalent to MCTAG from previous slides:

$N = \{\alpha, \beta_{A,B,C}, S\}$, start symbol S and rules

$S(X) \to \alpha(X)$

$\alpha(\varepsilon) \to \varepsilon$

$\alpha(X_1X_2X_3X_4X_5X_6) \to \beta_{A,B,C}(X_1, X_2, X_3, X_4, X_5, X_6)$

$\beta_{A,B,C}(a, b, c, d, e, f) \to \varepsilon$

$\beta_{A,B,C}(X_1a, X_2b, X_3c, dX_4, eX_5, fX_6) \to \beta_{A,B,C}(X_1, X_2, X_3, X_4, X_5, X_6)$

MCTAG and LCFRS (6)

Construction of an equivalent MCTAG for a given LCFRS: First, we make sure no non-terminal occurs twice in a rhs and our LCFRS is monotone. Then the construction is as follows:

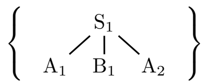
- For each rule we introduce a multicomponent set that contains an initial tree for each component of the lhs.
- The root of this initial tree is labelled A_k if A is the lhs symbol and the tree describes the kth component.
- The daughters describe the elements of this component from left to right, they are labelled (from left to right) with the terminals from the lhs and with B_i if the lhs element is the ith argument of the rhs element B.
- The MCTAG has a start symbol, namely S_1.

Note: This construction yields an MCTAG without adjunction.

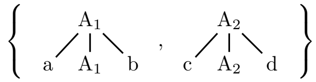
MCTAG and LCFRS (7)

Example:

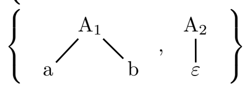
$S(XYZ) \rightarrow A(X,Z)B(Y)$



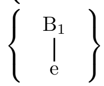
$A(aXb, cYd) \rightarrow A(X,Y)$



$A(ab, \varepsilon) \rightarrow \varepsilon$



$B(e) \rightarrow \varepsilon$



Minimalist Grammar (1)

- Minimalist Grammars (MGs) were proposed by [Stabler, 1997] as a formalization of Chomsky's Minimalist Program [Chomsky, 1995].
- Roughly, MGs consist of a set of trees together with two operations, *merge* and *move*, that allow us to transform these trees.
- Michaelis [Michaelis, 2001a, Michaelis, 2001b] has shown that MGs are equivalent to LCFRS.

Minimalist Grammar (2)

- An MG consists of a set *Lex* of finite ordered binary trees $\tau = \langle V, E, r\rangle$, so-called *expressions*.
- In such expressions τ, there is an additional relation of *projection* defined among sisters. For every $v_1 \neq v_2$ such that there exists a v with $\langle v, v_1\rangle$, $\langle v, v_2\rangle \in E$, either v_1 projects over v_2 or vice versa.
- Furthermore, all leaves in τ are labeled with a finite sequence of features.

Minimalist Grammar (3)

- A node $v \in V$ in an expression $\tau = \langle V, E, r\rangle$ is called a *maximal projection* if either $v = r$ or its sister projects over v.
- The *head* of a node $v \in V$ is the leaf $h(v)$ such that $\{v' \mid \langle v, v'\rangle \in E^+, \langle v', h(v)\langle \in E^*\}$ does not contain maximal projections, i.e., the path from v to its head contains only nodes that project over their sisters.

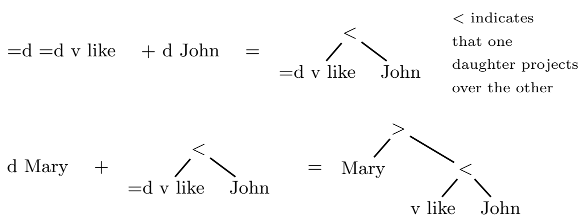
Minimalist Grammar (4)

Besides the set *Lex*, MG provides two operations, *merge* and *move* to create new expressions.

- *Merge* builds a new tree from two existing ones by considering them the two subtrees dominated by a new root node. Its application depends on the head features of the two trees and it modifies these features.
- *Move* transforms a single tree into a new one. Roughly, it consists of extracting a subtree, replacing it with a trace ε or deleting its phonetic material in the original place. The extracted subtree and the result of deleting it in the original tree become sisters with a new root node as mother.

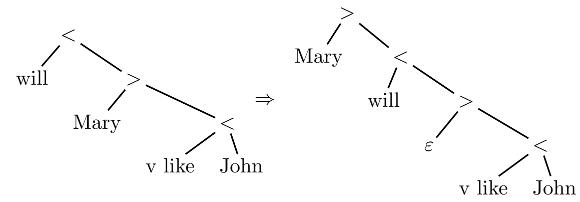
Minimalist Grammar (5)

Examples of *merge*:



Minimalist Grammar (5)

Example of *move*:



MG and LCFRS

From MG to an equivalent LCFRS (Intuition):

- Merge operations amount to concatenation, eventually with a gap in between the two concatenated parts.

  V'$(XY) \to$V(X)DP(Y), VP$(X, Y) \to$DP(X)V'(Y),
  T'$(X, Y, Z) \to$Aux(X)VP(Y, Z)

- Move amounts to a switching of components such that the kth component (for some $k > 1$) becomes the first/is concatenated to the first component.

  TP$(X, Y, Z) \to$T'(Y, X, Z)

This leads to non-monotone LCFRS (unordered simple RCG).

Other equivalent formalisms

- Finite-copying Lexical Functional Grammar [Seki et al., 1993]
- Hyperedge Replacement Grammars [Engelfriet and Heyker, 1991]
- Deterministic Tree-Walking Transducers [Weir, 1992]

References

[Boullier, 1998] Boullier, P. (1998). A generalization of mildly context-sensitive formalisms. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+4)*, pages 17–20, University of Pennsylvania, Philadelphia.

[Boullier, 2000] Boullier, P. (2000). A cubic time extension of context-free grammars. *Grammars*, 3(2/3):111–131.

[Chomsky, 1995] Chomsky, N. (1995). *The Minimalist Program*. MIT Press.

[Engelfriet and Heyker, 1991] Engelfriet, J. and Heyker, L. (1991). The string generating power of context-free hypergraph grammars. *Journal of Computer and System Sciences*, 43:328–360.

[Michaelis, 2001a] Michaelis, J. (2001a). Derivational minimalism is mildly context-sensitive. In Moortgat, M., editor, *Logical Aspects of Computational Linguistics*, volume 2014 of *LNCS/LNAI*, pages 179–198, Berlin, Heidelberg. Springer.

[Michaelis, 2001b] Michaelis, J. (2001b). Transforming linear context-free rewriting systems into minimalist grammars. In Philippe de Groote, Glyn Morrill, C. R., editor, *Logical Aspects of Computational Linguistics*, volume 2099 of *LNCS/LNAI*, pages 228–244, Berlin, Heidelberg. Springer.

[Seki et al., 1993] Seki, H., Nakanishi, R., Kaji, Y., Ando, S., and Kasami, T. (1993). Parallel multiple context-free grammars, finite-state translation systems, and polynomial-time recognizable subclasses of lexical-functional grammars. In *31st Meeting of the Association for Computational Linguistics (ACL'93)*, pages 121–129.

[Stabler, 1997] Stabler, E. P. (1997). Derivational Minimalism. In Retoré, C., editor, *Logical Aspects of Computational Linguistics*, number 1328 in Lecture Notes in Computer Science, pages 68–95, NY. Springer-Verlag.

[Weir, 1992] Weir, D. (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.

[Weir, 1988] Weir, D. J. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania.