Parsing Beyond Context-Free Grammars: LCFRS Normal Forms

Laura Kallmeyer, Patrick Hommers
Heinrich-Heine-Universität Düsseldorf

Sommersemester 2013

Overview

1. Introduction
2. Useless rules and ε-rules
3. Ordered Simple RCG
4. Binarization

Introduction (1)

- A *normal form* for a grammar formalism puts additional constraints on the form of the grammar while keeping the generative capacity.
- In other words, for every grammar G of a certain formalism, one can construct a weakly equivalent grammar G' of the same formalism that satisfies additional normal form constraints.
- Example: For CFGs we know that we can construct equivalent ε-free CFGs, equivalent CFGs in Chomsky Normal Form and equivalent CFGs in Greibach Normal Form.
- Normal Forms are useful since they facilitate proofs of properties of the grammar formalism.

Useless rules and ε-rules (1)

[Boullier, 1998] shows a range of useful properties of simple RCG/LCFRS/MCFG that can help to make formal proofs and parsing easier.

Boullier defines rules that cannot be used in any derivations for some $w \in T^*$ as *useless*.

Proposition 1 *For each k-LCFRS (k-simple RCG) G, there exists an equivalent simple k'-LCFRS (k'-simple RCG) G' with $k' \leq k$ that does not contain useless rules.*

The removal of the useless rules can be done in the same way as in the CFG case [Hopcroft and Ullman, 1979].

Useless rules and ε-rules (2)

[Boullier, 1998, Seki et al., 1991] show that the elimination of ε-rules is possible in a way similar to CFG. We define that a rule is an ε-rule if one of the arguments of the left-hand side is the empty string ε.

Definition 1 *A simple RCG/LCFRS is* ε-free *if it either contains no* ε*-rules or there is exactly one rule* $S(\varepsilon) \rightarrow \varepsilon$ *and* S *does not appear in any of the right-hand sides of the rules in the grammar.*

Proposition 2 *For every simple* k*-RCG (*k*-LCFRS)* G *there exists an equivalent* ε*-free simple* k'*-RCG (*k'*-LCFRS)* G' *with* $k' \leq k$*.*

Ordered Simple RCG (1)

In general, in MCFG/LCFRS/simple RCG, when using a rule in a derivation, the order of the components of its lhs in the input is not necessarily the order of the components in the rule.

Example:
$S(XY) \rightarrow A(X,Y), A(aXb, cYd) \rightarrow A(Y,X), A(e,f) \rightarrow \varepsilon.$

String language:

$\{(ac)^n e (db)^n (ca)^n f (bd)^n \mid n \geq 0\}$
$\cup \{(ac)^n afb (db)^n (ca)^n ced (bd)^n \mid n \geq 0\}$

Ordered Simple RCG (2)

Definition 2 (Ordered simple RCG) *A simple RCG is* ordered *if for every rule* $A(\vec{\alpha}) \rightarrow A_1(\vec{\alpha_1}) \dots A_k(\vec{\alpha_k})$ *and every* $A_i(\vec{\alpha_i}) = A_i(Y_1, \dots, Y_{dim(A_i)})$ $(1 \leq i \leq k)$*, the order of the components of* $\vec{\alpha_i}$ *in* $\vec{\alpha}$ *is* $Y_1, \dots, Y_{dim(A_i)}$*.*

Proposition 3 *For every simple* k*-RCG* G *there exists an equivalent ordered simple* k*-RCG* G'*.*

[Michaelis, 2001, Kracht, 2003, Kallmeyer, 2010]

In LCFRS terminology, this property is called *monotone* while in MCFG terminology, it is called *non-permuting*.

Binarization (1)

In LCFRS terminology, the length of the right-hand side of a production is called its *rank*. The *rank* of an LCFRS is given by the maximal rank of its productions.

Proposition 4 *For every simple RCG/LCFRS* G *there exists an equivalent simple RCG/LCFRS* G' *that is of rank 2.*

Unfortunately, the fan-out of G' might be higher than the fan-out of G.

The transformation can be performed similarly to the CNF transformation for CFG
[Hopcroft and Ullman, 1979, Grune and Jacobs, 2008].

Binarization (2)

Example:

$S(XYZUVW) \to A(X,U)B(Y,V)C(Z,W)$

$A(aX,aY) \to A(X,Y) \qquad A(a,a) \to \varepsilon$

$B(bX,bY) \to B(X,Y) \qquad B(b,b) \to \varepsilon$

$C(cX,cY) \to C(X,Y) \qquad C(c,c) \to \varepsilon$

Equivalent binarized grammar:

$S(XPUQ) \to A(X,U)C_1(P,Q) \qquad C_1(YZ,VW) \to B(Y,V)C(Z,W)$

$A(aX,aY) \to A(X,Y) \qquad A(a,a) \to \varepsilon$

$B(bX,bY) \to B(X,Y) \qquad B(b,b) \to \varepsilon$

$C(cX,cY) \to C(X,Y) \qquad C(c,c) \to \varepsilon$

Binarization (3)

We define the *reduction of a vector* $\vec{\alpha_1} \in [(T \cup V)^*]^{k_1}$ by a vector $\vec{x} \in (V^*)^{k_2}$ where all variables in $\vec{x}$ occur in $\vec{\alpha_1}$ as follows:

Take all variables from $\vec{\alpha_1}$ (in their order) that are not in $\vec{x}$ while starting a new component in the resulting vector whenever an element is, in $\vec{\alpha_1}$, the first element of a component or preceded by a variable from $\vec{x}$ or a terminal.

Examples:

1. $\langle aX_1, X_2, bX_3\rangle$ reduced with $\langle X_2\rangle$ yields $\langle X_1, X_3\rangle$.
2. $\langle aX_1X_2bX_3\rangle$ reduced with $\langle X_2\rangle$ yields $\langle X_1, X_3\rangle$ as well.

Binarization (4)

Transformation into a simple RCG of rank 2:

`for all` $r = A(\vec{\alpha}) \to A_0(\vec{\alpha_0}) \dots A_m(\vec{\alpha_m})$ `in` P `with` $m > 1$:

`remove` r `from` P `and pick new non-terminals` $C_1, \dots, C_{m-1}$

$R := \emptyset$

`add the rule` $A(\vec{\alpha}) \to A_0(\vec{\alpha_0})C_1(\vec{\gamma_1})$ `to` R `where` $\vec{\gamma_1}$

`is obtained by reducing` $\vec{\alpha}$ `with` $\vec{\alpha_0}$

`for all` i`,` $1 \le i \le m-2$:

`add the rule` $C_i(\vec{\gamma_i}) \to A_i(\vec{\alpha_i})C_{i+1}(\vec{\gamma_{i+1}})$ `to` R `where` $\vec{\gamma_{i+1}}$

`is obtained by reducing` $\vec{\gamma_i}$ `with` $\vec{\alpha_i}$

`add the rule` $C_{m-1}(\vec{\gamma_{m-2}}) \to A_{m-1}(\vec{\alpha_{m-1}})A_m(\vec{\alpha_m})$ `to` R

`for every rule` $r' \in R$

`replace rhs arguments of length` > 1 `with new variables`

`(in both sides) and add the result to` P

Binarization (5)

In our example, for the rule $S(XYZUVW) \to A(X,U)B(Y,V)C(Z,W)$, we obtain

$$R = \{ \quad S(XYZUVW) \to A(X,U)C_1(YZ,VW), \quad C_1(YZ,VW) \to B(Y,V)C(Z,W) \quad \}$$

Collapsing sequences of adjacent variables in the rhs leads to the two rules

$S(XPUQ) \to A(X,U)C_1(P,Q)$, $C_1(YZ,VW) \to B(Y,V)C(Z,W)$

References

[Boullier, 1998] Boullier, P. (1998). A Proposal for a Natural Language Processing Syntactic Backbone. Technical Report 3342, INRIA.

[Grune and Jacobs, 2008] Grune, D. and Jacobs, C. (2008). *Parsing Techniques. A Practical Guide*. Monographs in Computer Science. Springer. Second Edition.

[Hopcroft and Ullman, 1979] Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison Wesley.

[Kallmeyer, 2010] Kallmeyer, L. (2010). *Parsing Beyond Context-Free Grammars*. Cognitive Technologies. Springer, Heidelberg.

[Kracht, 2003] Kracht, M. (2003). *The Mathematics of Language*.

Number 63 in Studies in Generative Grammar. Mouton de Gruyter, Berlin.

[Michaelis, 2001] Michaelis, J. (2001). *On Formal Properties of Minimalist Grammars*. PhD thesis, Potsdam University.

[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.