Parsing Beyond Context-Free Grammars: LCFRS Normal Forms

Laura Kallmeyer & Tatiana Bladier
Heinrich-Heine-Universität Düsseldorf

Sommersemester 2018

Overview

[Kal10]

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.

Eliminating useless rules (1)

[Bou98] 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 [HU79].

Eliminating useless rules (2)

The removal of the useless rules can be done in the same way as in the CFG case [HU79]:

1. All rules need to be eliminated that cannot lead to a terminal sequence.
   This can be done recursively: Starting from the terminating rules and following the rules from right to left, the set of all non-terminals leading to terminals can be computed recursively.

Eliminating useless rules (3)

1. (continued)
   We can characterize this set N_T with the following deduction rules:

$$\frac{}{[A]} \quad A(\vec{\alpha}) \to \varepsilon \in P$$

$$\frac{[A_1], \dots, [A_m]}{[A]} \quad A(\vec{\alpha}) \to A_1(\vec{\alpha_1}) \dots A_m(\vec{\alpha_m}) \in P$$

   All rules that contain non-terminals in their right-hand side that are not in this set are eliminated.

Eliminating useless rules (4)

2. Then the unreachable rules need to be eliminated.
   This is done starting from all S-rules and moving from left-hand sides to right-hand sides. If the right-hand side contains a predicate A, then all A-rules are reachable and so on. Each time, the rules for the predicates in a right-hand side are added.

   We can characterize the set N_S of non-terminals reachable from S with the following deduction rules:

$$\frac{}{[S]} \qquad\qquad \frac{[A]}{[A_1], \dots, [A_m]} \; A(\vec{\alpha}) \to A_1(\vec{\alpha_1}) \dots A_m(\vec{\alpha_m}) \in P$$

   Rules whose left-hand side predicate is not in this set are eliminated.

Eliminating ε-rules (1)

[Bou98, SMFK91] 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 $k' \leq k$.

Eliminating ε-rules (2)

- First, we have to compute for all predicates A, all possibilities to have empty ranges among the components of the yields.
- For this, we introduce vectors $\vec{\iota} \in \{0, 1\}^{dim(A)}$ and we generate a set N_ε of pairs $(A, \vec{\iota})$ where $\vec{\iota}$ signifies that it is possible for A to have a tuple τ in its yield with $\tau(i) = \varepsilon$ if $\vec{\iota}(i) = 0$ and $\tau(i) \neq \varepsilon$ if $\vec{\iota}(i) \neq 0$.

Example:
$S(XY) \to A(X, Y)$, $A(a, \varepsilon) \to \varepsilon$, $A(\varepsilon, a) \to \varepsilon$, $A(a, b) \to \varepsilon$

Set of pairs characterizing possibilities for ε-components:
$N_\varepsilon = \{(S, 1), (A, 10), (A, 01), (A, 11)\}$

Eliminating ε-rules (3)

The set N_ε is constructed recursively:

1. $N_\varepsilon = \emptyset$.
2. For every rule $A(x_1, \dots, x_{dim(A)}) \to \varepsilon$, add $(A, \vec{\iota})$ to N_ε with for all $1 \leq i \leq dim(A)$: $\vec{\iota}(i) = 0$ if $x_i = \varepsilon$, else $\vec{\iota}(i) = 1$.
3. Repeat until N_ε does not change any more:
   For every rule $A(x_1, \dots, x_{dim(A)}) \to A_1(\alpha_1) \dots A_k(\alpha_k)$ and all $(A_1, \vec{\iota}_1)$, ..., $(A_k, \vec{\iota}_k) \in N_\varepsilon$:
   Calculate a vector $(x'_1, \dots, x'_{dim(A)})$ from $(x_1, \dots, x_{dim(A)})$ by replacing every variable that is the jth variable of A_m in the right-hand side such that $\vec{\iota}_m(j) = 0$ with ε.
   Then add $(A, \vec{\iota})$ to N_ε with for all $1 \leq i \leq dim(A)$: $\vec{\iota}(i) = 0$ if $x'_i = \varepsilon$, else $\vec{\iota}(i) = 1$.

Eliminating ε-rules (4)

Now that we have the set N_ε we can obtain reduced rules from the ones in the grammar where ε-arguments are left out.

Example:
$S(XY) \to A(X,Y)$, $A(a,\varepsilon) \to \varepsilon$, $A(\varepsilon,a) \to \varepsilon$, $A(a,b) \to \varepsilon$
$N_\varepsilon = \{(S,1),(A,10),(A,01),(A,11)\}$

Rules after ε-elimination ($(A,\vec{\iota})$ is written $A^{\vec{\iota}}$):
$S'(X) \to S^1(X)$, (S' takes care of the case of $\varepsilon \in L(G)$)
$S^1(X) \to A^{10}(X)$, $A^{10}(a) \to \varepsilon$,
$S^1(X) \to A^{01}(X)$, $A^{01}(b) \to \varepsilon$,
$S^1(XY) \to A^{11}(X,Y)$, $A^{11}(a,b) \to \varepsilon$

Eliminating ε-rules (5)

To obtain the new rules P_ε, we proceed as follows:

1. $P_\varepsilon = \emptyset$
2. We pick a new start symbol $S' \notin N_\varepsilon$.
   If $\varepsilon \in L(G)$, we add $S'(\varepsilon) \to \varepsilon$ to P_ε.
   If $S^1 \in N_\varepsilon$, we add $S'(X) \to S^1(X)$ to P_ε.
3. For every rule $A(\alpha) \to A_1(\vec{x}_1) \dots A_k(\vec{x}_k) \in P$: add all ε-reductions of this rule to P_ε.

Eliminating ε-rules (6)

The ε-reductions of $A(\alpha) \rightarrow A_1(\vec{x}_1) \dots A_k(\vec{x}_k)$ are obtained as follows: For all combinations of $\vec{\iota}_1, \dots, \vec{\iota}_k$ such that $A_i^{\vec{\iota}_i} \in N_\varepsilon$ for $1 \leq i \leq k$:

(i) For all i, $1 \leq i \leq k$: replace A_i in the rhs with $A_i^{\vec{\iota}_i}$ and for all j, $1 \leq j \leq dim(A_i)$: if $\vec{\iota}_i(j) = 0$, then remove the jth component of $A_i^{\vec{\iota}_i}$ from the rhs and delete the variable $\vec{x}_i(j)$ in the lhs.

(ii) Let $\vec{\iota} \in \{0, 1\}^{dim(A)}$ be the vector with $\vec{\iota}(i) = 0$ iff the ith component of A is empty in the rule obtained from (i). Remove all ε-components in the lhs and replace A with $A^{\vec{\iota}}$.

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

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 $A(\vec{\alpha}) \to A_1(\vec{\alpha_1}) \dots A_k(\vec{\alpha_k})$ 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′.

[Mic01, Kra03, Kal10]

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

Ordered Simple RCG (3)

Idea of the transformation:

- We check for every rule whether the component order in one of the right-hand side predicates A does not correspond to the one in the left-hand side.
- If so, we add a new predicate that differs from A only with respect to the order of the components. We replace A in the rule with the new predicate with reordered components.
- Furthermore, we add a copy of every A-rule with A replaced in the left-hand side by the new predicate and reordering of the components.

We notate the permutations of components as vectors where the ith element is the image of i. For a predicate A, *id* is the vector $\langle 1, 2, \dots, \dim(A) \rangle$.

Ordered Simple RCG (4)

Transformation into an ordered simple RCG:

$P' := P$ `with all predicates` A `replaced with` A^{id}`;`

$N' := \{A^{id} \mid A \in N\}$`;`

`repeat until` P' `does not change any more:`

```
for all r = A^p(α⃗) → A_1^{p_1}(α⃗_1) ... A_k^{p_k}(α⃗_k) in P':
   for all i, 1 ≤ i ≤ k:
      if A_i^{p_i}(α⃗_i) = A_i^{p_i}(Y_1, ..., Y_{dim(A_i)}) and the order of the
         Y_1, ..., Y_{dim(A_i)} in α⃗ is p(Y_1, ..., Y_{dim(A_i)})
         where p is not the identity

      then replace A_i^{p_i}(α⃗_i) in r with A_i^{p_i∘p}(p(α⃗_i))
         if A_i^{p_i∘p} ∉ N' then add A_i^{p_i∘p} to N' and
            for every A_i^{p_i}-rule A_i^{p_i}(γ⃗) → Γ ∈ P':
               add A_i^{p_i∘p}(p(γ⃗)) → Γ to P'
```

Ordered Simple RCG (5)

Consider again our example
$P' = \{S(XY) \to A(X, Y), A(aXb, cYd) \to A(Y, X), A(e, f) \to \varepsilon\}$.

- Problematic rule: $A^{\langle 1,2\rangle}(aXb, cYd) \to A^{\langle 1,2\rangle}(Y, X)$
- Introduce new non-terminal $A^{\langle 2,1\rangle}$ where $\langle 2, 1\rangle$ is the permutation that switches the two arguments.
  Replace $A^{\langle 1,2\rangle}(aXb, cYd) \to A^{\langle 1,2\rangle}(Y, X)$ with $A^{\langle 1,2\rangle}(aXb, cYd) \to A^{\langle 2,1\rangle}(X, Y)$.

  $P' = \{S(XY) \to A(X, Y), A(aXb, cYd) \to A^{\langle 2,1\rangle}(X, Y), A(e, f) \to \varepsilon\}$
- Add $A^{\langle 2,1\rangle}(f, e) \to \varepsilon$ and $A^{\langle 2,1\rangle}(cYd, aXb) \to A^{\langle 2,1\rangle}(X, Y)$.
- Now, $A^{\langle 2,1\rangle}(cYd, aXb) \to A^{\langle 2,1\rangle}(X, Y)$ is problematic.
  $\langle 2, 1\rangle \circ \langle 2, 1\rangle = \langle 1, 2\rangle$, therefore we replace this rule with $A^{\langle 2,1\rangle}(cYd, aXb) \to A^{\langle 1,2\rangle}(Y, X)$. $A^{\langle 1,2\rangle}$ is no new non-terminal, so no further rules are added.

Ordered Simple RCG (6)

Result:

$$S^{\langle 1\rangle}(XY) \rightarrow A^{\langle 1,2\rangle}(X, Y) \qquad A^{\langle 1,2\rangle}(e, f) \rightarrow \varepsilon$$
$$A^{\langle 1,2\rangle}(aXb, cYd) \rightarrow A^{\langle 2,1\rangle}(X, Y) \qquad A^{\langle 2,1\rangle}(f, e) \rightarrow \varepsilon$$
$$A^{\langle 2,1\rangle}(cYd, aXb) \rightarrow A^{\langle 1,2\rangle}(Y, X)$$

Note that in general, this transformation algorithm is exponential in the size of the original grammar.

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 [HU79, GJ08].

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. $\langle aX_1 X_2 bX_3 \rangle$ 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 $r = A(\vec{\alpha}) \rightarrow A_0(\vec{\alpha_0}) \dots A_m(\vec{\alpha_m})$ 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 $A(\vec{\alpha}) \rightarrow A_0(\vec{\alpha_0})C_1(\vec{\gamma_1})$ to R where $\vec{\gamma_1}$
      is obtained by reducing $\vec{\alpha}$ with $\vec{\alpha_0}$
   for all i, $1 \leq i \leq m-2$:
      add the rule $C_i(\vec{\gamma_i}) \rightarrow A_i(\vec{\alpha_i})C_{i+1}(\vec{\gamma_{i+1}})$ to R where $\vec{\gamma_{i+1}}$
         is obtained by reducing $\vec{\gamma_i}$ with $\vec{\alpha_i}$
   add the rule $C_{m-1}(\vec{\gamma_{m-2}}) \rightarrow A_{m-1}(\vec{\alpha_{m-1}})A_m(\vec{\alpha_m})$ 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
$S(XYZUVW) \rightarrow A(X, U)B(Y, V)C(Z, W)$, we obtain

$$R = \{ \quad \begin{array}{l} S(XYZUVW) \rightarrow A(X, U)C_1(YZ, VW), \\ C_1(YZ, VW) \rightarrow B(Y, V)C(Z, W) \end{array} \qquad \}$$

Collapsing sequences of adjacent variables in the rhs leads to the two rules
$S(XPUQ) \rightarrow A(X, U)C_1(P, Q)$, $C_1(YZ, VW) \rightarrow B(Y, V)C(Z, W)$

References I

[Bou98] Pierre Boullier.
A Proposal for a Natural Language Processing Syntactic Backbone.
Technical Report 3342, INRIA, 1998.

[GJ08] Dick Grune and Ceriel Jacobs.
Parsing Techniques. A Practical Guide.
Monographs in Computer Science. Springer, 2008.
Second Edition.

[HU79] John E. Hopcroft and Jeffrey D. Ullman.
Introduction to Automata Theory, Languages and Computation.
Addison Wesley, 1979.

[Kal10] Laura Kallmeyer.
Parsing Beyond Context-Free Grammars.
Cognitive Technologies. Springer, Heidelberg, 2010.

[Kra03] Marcus Kracht.
The Mathematics of Language.
Number 63 in Studies in Generative Grammar. Mouton de Gruyter, Berlin, 2003.

[Mic01] Jens Michaelis.
On Formal Properties of Minimalist Grammars.
PhD thesis, Potsdam University, 2001.

[SMFK91] Hiroyuki Seki, Takahashi Matsumura, Mamoru Fujii, and Tadao Kasami.
On multiple context-free grammars.
Theoretical Computer Science, 88(2):191–229, 1991.