Parsing beyond context-free grammar:

Simple RCG: Simplifying the Grammar

Laura Kallmeyer, Wolfgang Maier

Sommersemester 2009

Overview

1. Elimination of useless rules
2. Elimination of ε-Rules
3. Ordering
4. Binarization
5. Conclusion

Elimination of useless rules (1)

Boullier (1998)

A *useless rule* is a rule which cannot be used in a derivation $S(\langle 0, n\rangle) \stackrel{*}{\Rightarrow} \varepsilon$ for any $w \in T^*$.

Elimination (similar as in the CFG case) in two steps:

1. Compute the set N_T of all symbols $A \in N$ that can lead to a tuple of terminal strings:

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

$$\frac{[A_1], \ldots, [A_m]}{[A]}\ A(\vec{\alpha}) \rightarrow A_1(\vec{\alpha_1}) \ldots A_m(\vec{\alpha_m}) \in P$$

Rules that contain non-terminals not in N_T are eliminated.

Elimination of useless rules (2)

2. In the resulting simple RCG, eliminate unreachable rules:

Compute the set N_S of non-terminals reachable from S:

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

Rules that contain non-terminals not in N_S are eliminated.

Elimination of ε-Rules (1)

An ε-rule is a rule with one of the lhs arguments being ε.

A simple RCG is ϵ-*free* if

- it either contains no ϵ-rules
- or there is exactly one rule $S(\epsilon) \rightarrow \epsilon$ and S does not appear in any rhs of a rule in G

Elimination of ε-Rules (2)

First, compute for all $A \in N$, all possibilities to have ε-components in their yields:

- We introduce vectors $\vec{\iota} \in \{0,1\}^{dim(A)}$, and
- we compute the 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 $\vec{\iota}(i) \neq 0$.

Elimination of ε-Rules (3)

The set N_ϵ is constructed recursively with initial value $\emptyset$:

1. For every $A(x_1, \ldots, x_k) \rightarrow \epsilon \in P$:
   Add $(A, \vec{\iota})$ to N_ϵ with for all $1 \leq i \leq k$: $\vec{\iota}(i) = 0$ if $x_i = \varepsilon$, else $\vec{\iota}(i) = 1$ for all $1 \leq i \leq k$.
2. Repeat until N_ϵ does not change any more:
   For every $A(x_1, \ldots, x_k) \rightarrow A_1(\alpha_1) \ldots A_m(\alpha_m)$ and all $(A_1, \vec{\iota}_1)$, $\ldots$, $(A_m, \vec{\iota}_m) \in N_\epsilon$:
   - Calculate a vector $(x'_1, \ldots, x'_k)$ from $(x_1, \ldots, x_k)$ by removing every variable that is the jth variable of A_i in the righthand side such that $\vec{\iota}_i(j) = 0$ with ε.
   - Then add $(A, \vec{\iota})$ to N_ϵ with for all $1 \leq i \leq k$: $\vec{\iota}(i) = 0$ if $x'_i = \epsilon$, else $\vec{\iota}(i) = 1$.

Elimination of ε-Rules (4)

Now we can compute the new set of rules:

For every clause $A(x_1, \ldots, x_k) \rightarrow A_1(x_1^{(1)}, \ldots, x_{k_1}^{(1)}) \ldots A_m(x_1^{(m)}, \ldots, x_{k_m}^{(m)})$ and all $(A_1, \vec{\iota}_1)$, $\ldots$, $(A_m, \vec{\iota}_m) \in N_\epsilon$, we compute a clause for the new grammar:

- In the rhs, we replace A_i with $(A_i, \vec{\iota}_i)$ for all $1 \leq i \leq m$;
- in both, lhs and rhs, we delete all variables $x_j^{(i)}$ for $1 \leq i \leq m$ and $1 \leq j \leq k_i$ with $\vec{\iota}_i(j) = 0$;
- in the lhs, we replace A with $(A, \vec{\iota})$ where $\vec{\iota}(l) = 0$ iff the ith argument is ε;
- finally delete all ε-arguments in lhs and rhs.

Furthermore, we add a new start symbol (S') with $S'(X) \rightarrow S^1(X)$ and, if ε in the language, $S'(\varepsilon) \rightarrow \varepsilon$.

Elimination of ε-Rules (5)

Example: Original simple RCG rules:

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

Set of pairs characterizing possibilities for ε-components:

$N_\varepsilon = \{(S,1),(A,10),(A,01),(A,11)\}$

Rules after ε-elimination:

$S'(X) \to S^1(X)$,

$S^1(X) \to A^{10}(X)$, $A^{10}(a) \to \varepsilon$,

$S^1(X) \to A^{01}(X)$, $A^{01}(b) \to \varepsilon$,

$S^1(XY) \to A^{11}(X,Y)$, $A^{11}(a,b) \to \varepsilon$

Ordered simple RCG (1)

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 $A_i(\vec{\alpha_i}) = A_i(Y_1,\dots,Y_{dim(A_i)})$ $(1 \le i \le k)$, the order of the components of $\vec{\alpha_i}$ in $\vec{\alpha}$ is $Y_1,\dots,Y_{dim(A_i)}$.

Crucially, in an ordered simple RCG, the order of the components of the lhs predicate of a rule corresponds always to their order in the input.

For every simple RCG, there exists a weakly equivalent ordered simple RCG.

For the transformation, in addition to the $A \in N$, we introduce new predicates A^p where p a permutation of $\langle 1,\dots,dim(A)\rangle$.

Ordered simple RCG (2)

Transformation into an ordered simple RCG:

$P' := P$;

```
repeat until P' does not change any more:
    for all rules r = A(α⃗) → A_1(α⃗_1)...A_k(α⃗_k) in P':
        for all i, 1 ≤ i ≤ k:
            if A_i(α⃗_i) = A_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(α⃗_i) in r with A_i^p(p(α⃗_i));
                for every A_i-rule A_i(γ⃗) → Γ ∈ P':
                    add a new rule A_i^p(p(γ⃗)) → Γ to P'
                        (if not yet in P')
```


Ordered simple RCG (3)

Example:

Original clauses:

$S(XY) \to A(X,Y)$, $A(X,Y) \to A(Y,X)$, $A(aX,bY) \to A(X,Y)$, $A(a,b) \to \varepsilon$

Clauses after transformation into ordered RCG:

$S(XY) \to A(X,Y)$, $A(X,Y) \to A^p(X,Y)$, $A(aX,bY) \to A(X,Y)$, $A(a,b) \to \varepsilon$, $A^p(Y,X) \to A(Y,X)$, $A^p(bY,aX) \to A^p(Y,X)$, $A^p(b,a) \to \varepsilon$

(p being the permutation that switches the two arguments)

Binarization (1)

We call the length of the rhs of a rule its *rank*. The rank of a simple RCG is given by the maximal rank of its rules.

For every simple RCG, there is an equivalent simple RCG of rank 2.

CNF binarization for CFG: $A \rightarrow BCD$ replaced with $A \rightarrow BE$, $E \rightarrow CD$

A similar binarization algorithm can be applied to LCFRS/simple RCG:

For every rule with rhs length > 2, we introduce a new predicate that comprises all rhs predicates except the first. This is done repeatedly.

Binarization (2)

We define the *reduction* of a $\vec{\alpha_1} \in [(T \cup V)^*]^{k_1}$ by $\langle X_1, \ldots, X_{k_2} \rangle \in V^{k_2}$ where all X_i for $1 \leq i \leq k_2$ occur in $\vec{\alpha_1}$ as the following vector of variables:

We take all variables from $\vec{\alpha_1}$ (in their order) that are not in $\{X_1, \ldots, X_{k_2}\}$ while starting a new component whenever a variable is, in $\vec{\alpha_1}$, in a different component than the preceding variable in the result or in the same component but not adjacent to it.

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 $\langle X_1, X_3 \rangle$.

Binarization (3)

Binarization algorithm:

```
for all rules r = A(α⃗) → A_0(α⃗_0)...A_m(α⃗_m) in P with m > 1:
  remove r from P;
  R := ∅;
  pick new predicate names C_1,...,C_{m-1};
  add A(α⃗) → A_0(α⃗_0)C_1(γ⃗_1) to R where γ⃗_1 is α⃗ reduced with α⃗_0;
  for all i, 1 ≤ i < m-1:
      add C_i(γ⃗_i) → A_i(α⃗_i)C_{i+1}(γ⃗_{i+1}) to R where
          γ⃗_{i+1} is γ⃗_i reduced with α⃗_i;
  add C_{m-1}(γ⃗_{m-2}) → A_{m-1}(α⃗_{m-1})A_m(α⃗_m) to R;
  for every r' ∈ R:
      replace rhs arguments of length > 1 with new variables
      (in both sides) and add the result to P
```


Binarization (4)

Example: Original simple RCG:

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

$A(aX,aY) \rightarrow A(X,Y)$	$B(bX,bY) \rightarrow B(X,Y)$	$C(cX,cY) \rightarrow C(X,Y)$
$A(a,a) \rightarrow \varepsilon$	$B(b,b) \rightarrow \varepsilon$	$C(c,c) \rightarrow \varepsilon$

For the rule with righthand side of length > 2 we obtain

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

Equivalent simple RCG of rank 2:

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

$A(aX,aY) \rightarrow A(X,Y)$	$B(bX,bY) \rightarrow B(X,Y)$	$C(cX,cY) \rightarrow C(X,Y)$
$A(a,a) \rightarrow \varepsilon$	$B(b,b) \rightarrow \varepsilon$	$C(c,c) \rightarrow \varepsilon$

Binarization (5)

There are different ways to binarize a given simple RCG rule: We could choose any partition of the rhs predicates into two sets and then introduce new predicates for each element of the partition.

The arities (dimensions) of the new predicates depend on the partitions we choose.

Gomez et al. (2009) have shown how to obtain an optimal binarization for a given LCFRS in the sense of obtaining a minimal arity (*fan-out* in LCFRS terminology).

In addition, one could also try to minimize the number of variables per rule.

Binarization (6)

Example: $A(aXY, cZ, dU) \rightarrow B(X)C(Y, Z)D(U)$.

1. Partition into $B(X)$ and $C(Y, Z)D(U)$. Leads to
   $A(aXY, cZ, dU) \rightarrow B(X)E1(Y, Z, U)$,
   $E1(Y, Z, U) \rightarrow C(Y, Z)D(U)$
   (arity 3, 4 variables)
2. Partition into $C(Y, Z)$ and $B(X)D(U)$. Leads to
   $A(aXY, cZ, dU) \rightarrow E2(X, U)C(Y, Z)$, $E2(X, U) \rightarrow B(X)D(U)$
   (arity 2 and 4 variables)
3. Partition into $D(U)$ and $B(X)C(Y, Z)$. Leads to
   $A(aV, cZ, dU) \rightarrow E3(V, Z)D(U)$, $E3(XY, Z) \rightarrow B(X)C(Y, Z)$
   (arity 2 and 3 variables)

The third possibility is the best one since it gives us a minimal arity and a minimal variable number per clause.

Conclusion

- Since LCFRS/sRCG/MCFG is a natural extension of CFG, the transformation proposed for CFG in order to facilitate parsing can be applied to sRCG as well:
- We can eliminate useless rules and ε-rules, we can order the yield components and we can binarize the rules.