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 \( A \rightarrow X_1 \ldots X_k \) as a LCFRS rule
\( A(X_1 \ldots X_k) \rightarrow \gamma \)
where \( \gamma \) is the concatenation of all \( \gamma_i(X_i) \) where \( 1 \leq i \leq k \) and
\( X_i \in N \). The start predicate is \( \mathcal{S} \).

Example:

CFG:
\[
S \rightarrow aSb \\
S \rightarrow \epsilon
\]

1-LCFRS:
\[
\mathcal{S}(aSb) \rightarrow \mathcal{S}(S) \\
S \rightarrow \epsilon
\]

Proposition 1

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

Grammar Formalisms 1 LCFRS and other formalisms

Kallmeyer Sommersemester 2011

Grammar Formalisms 2 LCFRS and other formalisms

CFG and LCFRS (2)

Construction of a CFG for a given 1-LCFRS: write every rule
\( A(a) \rightarrow A_1(X_1) \ldots A_k(X_k) \) as a CFG rule
\( A \rightarrow f(a) \) 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:
\[
S(aXbY) \rightarrow S(X)A(Y) \\
S(\epsilon) \rightarrow \epsilon \\
A(cX) \rightarrow A(X) \\
A(\epsilon) \rightarrow \epsilon
\]

CFG:
\[
S \rightarrow aSbA \\
S \rightarrow \epsilon \\
A \rightarrow cA \\
A \rightarrow \epsilon
\]

Grammar Formalisms 3 LCFRS and other formalisms

Kallmeyer Sommersemester 2011

Grammar Formalisms 4 LCFRS and other formalisms
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 $\alpha$ and auxiliary trees $\beta$ respectively.
- $X$ covers the yield of $\alpha$ and all trees added to $\alpha$, while $L$ and $R$ cover those parts of the yield of $\beta$ (including all trees added to $\beta$) that are to the left and the right of the foot node of $\beta$.
- The rules reduce the components of these non-terminals by identifying those parts that come from the elementary tree $\alpha/\beta$ itself and those parts that come from one of the elementary trees added by substitution or adjunction.

MCTAG and LCFRS (1)

MCTAG example (reminder): $\alpha$

\[ \begin{array}{c}
\downarrow \\
A \\
\downarrow \\
B \\
\downarrow \\
C \\
\end{array} \]

\[ \{ \begin{array}{c}
\beta_A \\
\beta_B \\
\beta_C \\
\end{array} \] 

\( \begin{array}{c}
a \\
A_{X_A} \\
f \\
\end{array} \) 

\( \begin{array}{c}
b \\
B_{X_A} \\
e \\
\end{array} \) 

\( \begin{array}{c}
c \\
C_{X_A} \\
d \\
\end{array} \) 

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

MCTAG and LCFRS (2)

Derivation for $aabbccddeeff$: 

\[ \begin{array}{c}
A \\
\downarrow \\
A_{X_A} \\
f \\
\end{array} \]

\( \begin{array}{c}
a \\
A_{X_A} \\
f \\
\end{array} \) 

\( \begin{array}{c}
b \\
B_{X_A} \\
e \\
\end{array} \) 

\( \begin{array}{c}
c \\
C_{X_A} \\
d \\
\end{array} \)
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 $\Gamma$ and all sets $\Gamma_1, \ldots, \Gamma_k$ that can attach to $\Gamma$ such that all obligatory adjunctions and substitutions are performed, we introduce a rule that tells us how the yield of $\Gamma$ can be obtained from the yields of $\Gamma_1, \ldots, \Gamma_k$ and from the terminals occurring in $\Gamma$.
- For every set $\Gamma$ without substitution nodes or OA constraints, we add a terminating rule that lists only the terminals occurring in $\Gamma$.

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

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

$$S(X) \rightarrow \alpha(X)$$
$$\alpha(\varepsilon) \rightarrow \varepsilon$$
$$\alpha(X_1X_2X_3X_4X_5X_6) \rightarrow \beta_{A,B,C}(X_1, X_2, X_3, X_4, X_5, X_6)$$
$$\beta_{A,B,C}(a, b, c, d, e, f) \rightarrow \varepsilon$$
$$\beta_{A,B,C}(X_1\alpha, X_2\beta, X_3\alpha, dX_4, eX_5, fX_6) \rightarrow \beta_{A,B,C}(X_1, X_2, X_3, X_4, X_5, X_6)$$

MCTAG and LCFRS (5)

Construction of an equivalent MCT AG 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 $A_i$ if $A$ is the lhs symbol and the tree describes the $k$th 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 rhs and with $B_i$ if the lhs element is the $i$th argument of the rhs element $B$.
- The MCT AG has a start symbol, namely $S_1$.

Note: This construction yields an MCTAG without adjunction.
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 (4)

Besides the set \textit{Lex}, MG provides two operations, \textit{merge} and \textit{move} to create new expressions.

- \textit{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.
- \textit{Move} transforms a single tree into a new one. Roughly, it consists of extracting a subtree, replacing it with a trace \(\epsilon\) 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.

Examples of \textit{merge}:
\[
\begin{align*}
=d=d \text{ v like } + d \text{ John } &= \quad =d \text{ v like } \text{ John} \\
d \text{ Mary } + =d \text{ v like } \text{ John } &= \quad \text{ Mary } =d \text{ v like } \text{ John}
\end{align*}
\]

Minimalist Grammar (5)

Example of \textit{move}:

```
\begin{tikzpicture}
  \node (M) {Mary} ;
  \node (W) [above of = M] {will} ;
  \node (J) [right of = M] {John} ;
  \node (V) [left of = M] {v like} ;
  \draw [->] (M) -- (J);
  \draw [->] (V) -- (M);
\end{tikzpicture}
```

```
\begin{tikzpicture}
  \node (M) {Mary} ;
  \node (W) [above of = M] {will} ;
  \node (J) [right of = M] {John} ;
  \node (V) [left of = M] {v like} ;
  \draw [->] (M) -- (J);
  \draw [->] (V) -- (M);
\end{tikzpicture}
```

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.
  \[
  \begin{align*}
  V'(XY) &\rightarrow V(X)DP(Y), \\
  VP(X,Y) &\rightarrow DP(X)V'(Y), \\
  T'(X,Y,Z) &\rightarrow Aux(X)VP(Y,Z)
  \end{align*}
  \]

- Move amounts to a switching of components such that the \(k\)th component (for some \(k > 1\)) becomes the first/is concatenated to the first component.
  \[
  TP(X,Y,Z) \rightarrow T'(Y,X,Z)
  \]

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


