Minimalist Grammars
Formalisme en Dependency Structures

Sjoerd Dost     Myrthe Tielman

Logical Methods in NLP
03/05/12
Inhoud

1. The Minimalist Program
2. Minimalist Grammars
   - Lexicon
3. Merge
   - Merge
   - Move
4. Onion
5. Dependency Structures
   - Aim
   - Merge1
   - Merge2 and 3
   - Move1
   - Move2
6. Block Degree
The Minimalist Program

Chomsky, Universal Grammar
The Minimalist Program

- Chomsky, Universal Grammar
- "effort to attribute as little as possible to UG while still accounting for the apparent diversity of human languages" (Stabler, 2011)
The Minimalist Program

- Chomsky, Universal Grammar
- "effort to attribute as little as possible to UG while still accounting for the apparent diversity of human languages" (Stabler, 2011)
- Endocentricity
"A head X determines certain relevant properties of the phrase XP it is the head of."

Merging head x with xy gives \{x\{xy\}\}
A minimalist grammar $G$ is a lexicon:

$$G \subseteq \text{PhoneticFeatures} \times \text{Features}^*, \text{a finite set}$$

Where $\text{Features}^*$ is the set of finite sequences of syntactic features

Where the elements of the lexicon are combined by the merge operation. \textit{(Stabler, 2011)}
Features

Features:

- Categorial

\[ N, V, A, P, C, T, D, \ldots \]
Features

Features:

- Categorial
  
  \[ N, V, A, P, C, T, D, \ldots \]

- Selector

  \[ = N, = V, = A, = P, = C, = T, = D, \ldots \]
Features

Features:

- **Categorial**

  \[ N, V, A, P, C, T, D, \ldots \]

- **Selector**

  \[ = N, = V, = A, = P, = C, = T, = D, \ldots \]

- **Licensee (goal)**

  \[ -wh, -case, \ldots \]
Features

Features:

- **Categorial**
  
  \[ N, V, A, P, C, T, D, \ldots \]

- **Selector**
  
  \[ = N, = V, = A, = P, = C, = T, = D, \ldots \]

- **Licensee (goal)**
  
  \[ -wh, -case, \ldots \]

- **Licensor (probes)**
  
  \[ +wh, +case, \ldots \]
Example Lexicon

Who Marie praises?

Marie :: D
who :: D -wh
praises :: =D =D V
ε :: =T +wh C
Merge

- From lexical items to trees
Merge

- From lexical items to trees
- Lexical items are 1 node trees
Merge

- From lexical items to trees
- Lexical items are 1 node trees
- Merge (external merge) and Move (internal merge)
Merge

- From lexical items to trees
- Lexical items are 1 node trees
- Merge (external merge) and Move (internal merge)
- Merge to merge 2 trees together
Merge

- From lexical items to trees
- Lexical items are 1 node trees
- Merge (external merge) and Move (internal merge)
- Merge to merge 2 trees together
- Move to restructure trees
Merge rule

\[ em(t_1[x], t_2[x]) = \begin{cases} 
< & \text{if } |t_1| = 1 \\
\begin{array}{c}
t_1 \\
<
\end{array} & \begin{array}{c}
t_2 \\
>
\end{array} \\
\begin{array}{c}
t_2 \\
>
\end{array} & \begin{array}{c}
t_1 \\
<
\end{array} & \text{otherwise}
\end{cases} \]

\[
\begin{align*}
\text{merge1:} & \quad s :: = f \gamma \quad t \cdot f, \alpha_1, \ldots, \alpha_k \\
& \quad \overline{st : \gamma, \alpha_1, \ldots, \alpha_k} \\
\text{merge2:} & \quad s :: = f \gamma, \alpha_1, \ldots, \alpha_k \quad t \cdot f, u_1, \ldots, u_l \\
& \quad \overline{ts : \gamma, \alpha_1, \ldots, \alpha_k, u_1, \ldots, u_l} \\
\text{merge3:} & \quad s :: = f \gamma, \alpha_1, \ldots, \alpha_k \quad t \cdot f \delta, u_1, \ldots, u_l \\
& \quad \overline{s : \gamma, \alpha_1, \ldots, \alpha_k, t : \delta, u_1, \ldots, u_l}
\end{align*}
\]
Merge rule

Merge (external merge)

\[ \text{praises::=} D = D \ V \ + \ \text{Pierre::=} D \rightarrow \] 
\[ \text{praises::=} D \ V \quad \text{Pierre} \]

\[ \text{praises::=} D \ V \quad \text{Pierre} \] 
\[ + \quad \text{Marie::=} D \rightarrow \] 
\[ \text{Marie} \] 
\[ \downarrow \] 
\[ \text{praises::=} V \quad \text{Pierre} \]
Move (internal merge)

- SMC: Exactly one head in the tree has -x as its first feature.
Move (internal merge)

- **SMC**: Exactly one head in the tree has $-x$ as its first feature.
- $t\{t_1 \leftrightarrow t_2\}$: the result of replacing subtree $t_1$ by $t_2$ in $t$. 
Move (internal merge)

- SMC: Exactly one head in the tree has -x as its first feature.
- $t\{t_1 \leftrightarrow t_2\}$: the result of replacing subtree $t_1$ by $t_2$ in $t$.
- $t^M$: the maximal projection of the head of $t$. 
Move

Move (internal merge)

- SMC: Exactly one head in the tree has -x as its first feature.
- \( t \{ t_1 \leftarrow t_2 \} \): the result of replacing subtree \( t_1 \) by \( t_2 \) in \( t \).
- \( t^M \): the maximal projection of the head of \( t \).
- A subtree is a maximal projection, if it is not properly included in any larger subtree that has the same head.
Move

\[ \text{im}(t_1[+x]) = t^M \]

\[ t^M_2 \quad t_1 \{ t_2[-x]^M \mapsto \varepsilon \} \quad \text{if SMC.} \]

\[ s\cdot +f_\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f, \alpha_{i+1}, \ldots, \alpha_k \]

\[ t_\text{move1} : \gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k \]

\[ s\cdot +f_\gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f_\delta, \alpha_{i+1}, \ldots, \alpha_k \]

\[ s : \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : \delta, \alpha_{i+1}, \ldots, \alpha_k \quad \text{move2} \]
Move (internal merge)

\[ \varepsilon:\!+\!wh \ C \]
Structuurregels

Move (internal merge)

```
>  
  < 
  < 
  which  student  ε:C  >
  Marie < praises ε
```
CMG = Conflated Minimalist Grammar, PMG = Phase-based Minimalist Grammar, RMG = Relativized Minimalist Grammar, MCFG = Multiple Context-free Grammar
"An interpretation of Minimalist Grammars in terms of dependency structures." (Boston, Hale Kuhlmann 2010)
### Table 1. Merge and Move

\[
\begin{align*}
  s ::= & f \gamma & t \cdot, f, \alpha_1, \ldots, \alpha_k \\
  & st : \gamma, \alpha_1, \ldots, \alpha_k & \text{merge1} \\
  s ::= & f \gamma, \alpha_1, \ldots, \alpha_k & t \cdot, f, \iota_1, \ldots, \iota_l \\
  & ts : \gamma, \alpha_1, \ldots, \alpha_k, \iota_1, \ldots, \iota_l & \text{merge2} \\
  s ::= & f \gamma, \alpha_1, \ldots, \alpha_k & t \cdot f \delta, \iota_1, \ldots, \iota_l \\
  & s : \gamma, \alpha_1, \ldots, \alpha_k, t : \delta, \iota_1, \ldots, \iota_l & \text{merge3} \\
  s ::= & f \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f, \alpha_{i+1}, \ldots, \alpha_k \\
  & ts : \gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k & \text{move1} \\
  s ::= & f \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : -f \delta, \alpha_{i+1}, \ldots, \alpha_k \\
  & s : \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : \delta, \alpha_{i+1}, \ldots, \alpha_k & \text{move2}
\end{align*}
\]
Table 2. Merge in terms of dependency trees

\[
(\{\lambda\}, \langle\lambda\rangle) \to f \gamma \quad (t, x) \cdot f, \alpha_1, \ldots, \alpha_k \quad \text{merge1}\text{DG}
\]
\[
(\{\lambda\} \cup \uparrow_i t, \langle\lambda\rangle \cdot \uparrow_i x) : \gamma, \uparrow_i \alpha_1, \ldots, \uparrow_i \alpha_k \quad \text{merge2}\text{DG}
\]
\[
(s, x) : = f \gamma, \alpha_1, \ldots, \alpha_k \quad (t, y) \cdot f, \upsilon_1, \ldots, \upsilon_l \quad \text{merge3}\text{DG}
\]
\[
(s, x) : \gamma, \alpha_1, \ldots, \alpha_k \quad (t, y) : \delta, \uparrow_i \upsilon_1, \ldots, \uparrow_i \upsilon_l
\]
where \( i = \text{next}((s, x) : = f \gamma, \alpha_1, \ldots, \alpha_k) \)

Table 3. Move in terms of dependency trees

\[
(s, x) : + f \gamma, \alpha_1, \ldots, \alpha_{i-1}, (t, y) : - f, \alpha_{i+1}, \ldots, \alpha_k \quad \text{move1}\text{DG}
\]
\[
(s \cup t, yx) : \gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k
\]
\[
(s \cdot + f \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : - f \delta, \alpha_{i+1}, \ldots, \alpha_k \quad \text{move2}\text{DG}
\]
\[
s : \gamma, \alpha_1, \ldots, \alpha_{i-1}, t : \delta, \alpha_{i+1}, \ldots, \alpha_k
\]
The Minimalist Program
Minimalist Grammars
Merge
Onion
Dependency Structures
Block Degree
Nestedness

Aim
Merge 1
Merge 2 and 3
Move 1
Move 2

\[
\begin{align*}
  s ::=& f\gamma \quad t \cdot f, \alpha_1, \ldots, \alpha_k \\
  \frac{s t : \gamma, \alpha_1, \ldots, \alpha_k}{merge 1} \\

  s ::=& f\gamma, \alpha_1, \ldots, \alpha_k \quad t \cdot f, \upsilon_1, \ldots, \upsilon_i \\
  \frac{ts : \gamma, \alpha_1, \ldots, \alpha_k, \upsilon_1, \ldots, \upsilon_i}{merge 2} \\

  s ::=& f\gamma, \alpha_1, \ldots, \alpha_k \quad t \cdot f\delta, \upsilon_1, \ldots, \upsilon_i \\
  \frac{s : \gamma, \alpha_1, \ldots, \alpha_k, t : \delta, \upsilon_1, \ldots, \upsilon_i}{merge 3}
\end{align*}
\]

Dost, Tielman
Minimalist Grammars
Fig. 1. $\text{merge}^1_{DG}$ applies to two simple dependency trees
The Minimalist Program
Minimalist Grammars
Merge
Onion
Dependency Structures
Block Degree
Nestedness

Aim
Merge1
Merge2 and 3
Move1
Move2

Fig. 2. merge2_{DG} and merge3_{DG}
Dost, Tielman

Minimalist Grammars

The Minimalist Program
Minimalist Grammars
Merge
Onion

Dependency Structures
Block Degree
Nestedness

Aim
Merge1
Merge2 and 3
Move1
Move2

\[
\frac{s : +\gamma, \alpha_1, \ldots, \alpha_i, \ldots}{t : -f, \alpha_{i+1}, \ldots, \alpha_k}
\]

\[
\frac{s : +\gamma, \alpha_1, \ldots, \alpha_i, \ldots}{\gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k}
\]

\[
\frac{t : f_\gamma, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k}{s : \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k}
\]

\[
\frac{t : f_\delta, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k}{s : \delta, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k}
\]
### The Minimalist Program

**Minimalist Grammars**

- Merge
- Onion

**Dependency Structures**

- Block Degree
- Nestedness

---

**Aim**

- Merge 1
- Merge 2 and 3
- Move 1
- Move 2

---

**Figure 3.** move1DG
The Minimalist Program
Minimalist Grammars
Merge
Onion
Dependency Structures
Block Degree
Nestedness

Aim
Merge1
Merge2 and 3
Move1
Move2

Dost, Tielman

Minimalist Grammars
Aim
Merge1
Merge2 and 3
Move1
Move2

Fig. 4. move2_{DG}
Dost, Tielman

**Minimalist Grammars**

*Fig. 4. move2_{DG}*

(SMC) None of $\alpha_1, \alpha_i, \ldots, \alpha_{i+1}, \ldots, \alpha_k$ has $-f$ as its first feature.
- Projectivity: subtrees span intervals
- Projectivity: subtrees span intervals
- Non-projective structures violate this constraint
- Projectivity: subtrees span intervals
- Non-projective structures violate this constraint
- Block degree: maximum number of intervals spanned

Fig. 5. The block degree of this structure is 2
Merge always forms dependency relations between roots of subtrees by construction
- Merge always forms dependency relations between roots of subtrees by construction
- Move1 can create non-projective structures
- Merge always forms dependency relations between roots of subtrees by construction
- Move1 can create non-projective structures
- Movement is triggered by lincensor-licensee pairs: block degrees bounded by \#licensees
Wellnested structures prohibit the crossing of disjoint subtree intervals.
Wellnested structures prohibit the crossing of disjoint subtree intervals.

Not all mildly context-sensitive formalisms can derive illnested structures (i.e. TAG).
MGs can derive illnested structures (see example)