A deductive perspective on Minimalism
Hungarian verb movement: a case study

Willemijn Vermaat
Contents

1 Background: Central thesis .......................................................... 3
2 Illustration: Verbal complex formation ......................................... 5
3 A minimalist grammar $\mathcal{MG}$ .................................................. 7
  3.1 Features .............................................................................. 7
  3.2 Lexicon .............................................................................. 8
  3.3 Structure building operations ............................................... 9
4 A $\mathcal{MG}$ approach to VM-climbing ....................................... 11
5 A mapping: MG to MMCG ......................................................... 13
  5.1 Mapping of Operations ......................................................... 15
6 A deductive approach to VM-climbing ....................................... 17
7 Conclusion .............................................................................. 22
1. Background: Central thesis

What controls Movement?

1. The Computational System of Human Language

2. The Lexicon: “Parametric differences between languages are defined as individual, lexical feature information of words that are involved” (Ruys, NRC Handelsblad 23/01/99)
The Computational System of Human Language

![Diagram of the Computational System of Human Language]

- Select
- Merge
- Move
- Move F
- Spell-Out
- LEXICON
- NUMERATION
- MP
- CHL
- B.O.C.
- LF
- B.O.C.
2. Illustration: Verbal complex formation
2. **Illustration: Verbal complex formation**

**Two formation processes**

- *Verb modifier climbing* in neutral sentences
- *Verbal inversion* in non-neutral sentences
2. Illustration: Verbal complex formation

Two formation processes

- *Verb modifier climbing* in neutral sentences
- *Verbal inversion* in non-neutral sentences

Kind of words in verbal complexes:
2. Illustration: Verbal complex formation

Two formation processes

- Verb modifier climbing in neutral sentences
- Verbal inversion in non-neutral sentences

Kind of words in verbal complexes:

- verb modifiers (prefixes, infinitives):
  
  haza (‘home’), be (‘in’), úsz-ni (‘to swim’)

- auxiliaries:
  
  fogok (‘I will’), akar (‘I want’), kezd (‘I begin’)
Examples of VM-climbing
Examples of VM-climbing

haza akarok menni M V1 V2
home want[1sg] go[inf]
‘I want to go home’
Examples of VM-climbing

haza akarok menni  
home want[1sg] go[inf]  
‘I want to go home’

* akarok menni haza (as neutral sentence)  
want[1sg] go[inf] home
Examples of VM-climbing

haza akarok menni
home want[1sg] go[inf]
‘I want to go home’

* akarok menni haza (as neutral sentence)
want[1sg] go[inf] home

* akarok haza menni (as neutral sentence)
want[1sg] home go[inf]
3. A minimalist grammar $MG$
3. A minimalist grammar $MG$

3.1. Features
3. A minimalist grammar \( MG \)

3.1. Features

Category features:

- Basic category features: \( c, t, d, n, v, p, \ldots \)
- Selector features: \( =c, =t, =d, =n, =v, =p, \ldots \)
3. A minimalist grammar \( \mathcal{M}G \)

3.1. Features

Category features:

Basic category features: \( c, t, d, n, v, p, \ldots \)
Selector features: \( =c, =t, =d, =n, =v, =p, \ldots \)

Control features:
3. A minimalist grammar $MG$

3.1. Features

Category features:

- Basic category features: $c, t, d, n, v, p, \ldots$
- Selector features: $=c, =t, =d, =n, =v, =p, \ldots$

Control features:

- Licensor features: $+wh, +case, +focus, \ldots$
3. **A minimalist grammar \( MG \)**

3.1. **Features**

Category features:

- Basic category features: \( c, t, d, n, v, p, \ldots \)
- Selector features: \( =c, =t, =d, =n, =v, =p, \ldots \)

Control features:

- Licensor features: \( +wh, +case, +focus, \ldots \)
- Licensee features: \( -wh, -case, -focus, \ldots \)
3. A minimalist grammar $MG$

3.1. Features

Category features:

- Basic category features: $c, t, d, n, v, p, \ldots$
- Selector features: $=c, =t, =d, =n, =v, =p, \ldots$

Control features:

- Licensor features: $+wh, +case, +focus, \ldots$
- Licensee features: $-wh, -case, -focus, \ldots$

Non-syntactic features
3. A minimalist grammar $\mathcal{MG}$

3.1. Features

Category features:

Basic category features: $c, t, d, n, v, p, \ldots$
Selector features: $=c, =t, =d, =n, =v, =p, \ldots$

Control features:

Licensor features: $+wh, +case, +focus, \ldots$
Licensee features: $-wh, -case, -focus, \ldots$

Non-syntactic features

Semantic features
Phonological features
3.2. Lexicon

**Lexical feature specification**

\[(=f \ (=f)^* \ (+f)) \ f \ (-f)^* \ /f/ \ f\]
3.2. Lexicon

Lexical feature specification

\[(=f (=f)^* (+f)) f (-f)^* /f/ f\]

Finite state automaton

![Finite state automaton diagram]
3.3. Structure building operations

Merge

\[
\text{MERGE}(t_1[c], t_2[c]) = \begin{cases} 
< & \text{if } t_1 \in \text{Lex} \\
\begin{array}{c}
t_1 \\
t_2
\end{array} & \text{otherwise}
\end{cases}
\]
Move is feature driven
Move is feature driven

- *licensor* features $[+f]$ attract *licensee* features $[-f]$
- VM-climbing: the licensee feature $[-m]$ is on the verbal modifier
4. **A $MG$ approach to VM-climbing**

**Lexicon:**

| $=v$ | $+_m$ | $I$ | $=v$ | $v$ | $akarok$ | $=m$ | $v$ | $menni$ | $m$ | $-m$ | $haza$ |
4. A $\mathcal{MG}$ approach to VM-climbing

Lexicon:

| $=v$ | $+m$ | $I$ | $=v$ | $v$ akarok | $=m$ | $v$ menni | $m$ | $-m$ haza |

Derivation:

(1) $\text{LEX: } m \ -m \ haza$
(2) $\text{LEX: } =m \ v \ menni$
4. A \( \mathcal{MG} \) approach to VM-climbing

Lexicon:

\[
\begin{array}{|c|c|c|c|}
\hline
=v & +m & I & =v & v & akarok & =m & v & menni & m & -m & haza \\
\hline
\end{array}
\]

Derivation:

(1) \text{LEX: } m & -m & haza
(2) \text{LEX: } =m & v & menni
(3) \text{MERGE}(2,1):

\[
\begin{array}{c}
< \\
\hline
v & \text{menni} & -m & \text{haza}
\end{array}
\]
(4) **LEX**: =v v akarok
(4) **Lex:** =v v akarok

(5) **Merge(4,3):**

```
<
  
v akarok <
    
menni −m haza
```
(4) **Lex:** =v v akarok

(5) **Merge(4,3):**

```
<
  
  v akarok <

  menni  -m haza
```
(4) **Lex**: =v v akarok
(5) **Merge**(4,3):

```
<

v akarok
```

```
menni  -m haza
```

(6) **Lex**: =v +m I
(7) **Merge**(6,5):

```
<

+ m I
```

```
<

akarok
```

```
menni  - m haza
```
(4) **Lex:** =v v akarok
(5) **Merge**(4,3):

\[
\begin{array}{c}
< \\
v \text{ akarok} \\
< \\
menni \quad \text{−m} \quad \text{haza}
\end{array}
\]

(6) **Lex:** =v +m I
(7) **Merge**(6,5):

\[
\begin{array}{c}
< \\
+m \quad \text{I} \\
< \\
akarok \\
< \\
menni \quad \text{−m} \quad \text{haza}
\end{array}
\]

(8) **Move**(7):

\[
\begin{array}{c}
> \\
haza \\
< \\
I \\
< \\
akarok \quad \text{menni}
\end{array}
\]

Step (8): the licensor feature [+m] requires a licensee feature [−m]. Haza carries the required feature. Under influence of the operation **Move**, the subtree is merged to the main tree as the specifier of the functional category [I].
5. A mapping: MG to MMCG
5. A mapping: MG to MMCG

Mapping of features

**Binary connectives:** $\mathcal{F} ::= A \mid \mathcal{F}/i\mathcal{F} \mid \mathcal{F} \bullet_i \mathcal{F} \mid \mathcal{F}\setminus_i \mathcal{F}$

**Unary Connectives:** $\mathcal{F} ::= \square_f \mathcal{F} \mid \diamond_f \mathcal{F}$
5. A mapping: MG to MMCG

Mapping of features

**Binary connectives:** \( F ::= A \mid F \mathbin{/} i F \mid F \mathbin{\cdot} i F \mid F \mathbin{\backslash} i F \)

**Unary Connectives:** \( F ::= \Box_f F \mid \Diamond_f F \)

**Feature correspondence**

<table>
<thead>
<tr>
<th>Kind of feature</th>
<th>MG</th>
<th>MMCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic categories</td>
<td>( c )</td>
<td>( c^\bullet )</td>
</tr>
<tr>
<td>Selector features</td>
<td>( =c )</td>
<td>( c^\circ ): for example ((c \mathbin{\backslash} \mathbin{\sim})^\bullet), ((\mathbin{-}/ &lt; c)^\bullet)</td>
</tr>
<tr>
<td>Licensee features</td>
<td>( [\neg f] )</td>
<td>( \Box_f ) on a formula with polarity ( ^\bullet )</td>
</tr>
<tr>
<td>Licensor features</td>
<td>( [+f] )</td>
<td>( \Diamond_f ) on a formula with polarity ( ^\circ )</td>
</tr>
</tbody>
</table>
Lexical correspondence

\[(f \ F)_{\alpha} = (F)^{\gamma} f\]
\[=f \ F)_{\alpha} = (F)^{\beta} / f\]
\[=f \ F)_{\beta} = f \backslash (F)^{\beta}\]
\[(f \ F)_{\beta} = (F)^{\gamma} f\]
\[(+f \ g \ F)_{\beta} = (F)^{\gamma} g\]
\[-f \ F)_{\gamma} = (F)^{\gamma} \Box_f\]
\[(-)_{\gamma} = \_\]
Lexical correspondence

\[(f \mathcal{F})^\alpha = (\mathcal{F})^\gamma f\]
\[(= f \mathcal{F})^\alpha = (\mathcal{F})^\beta / < f\]
\[(= f \mathcal{F})^\beta = f \setminus (\mathcal{F})^\beta\]
\[(f \mathcal{F})^\beta = (\mathcal{F})^\gamma f\]
\[(+ f g \mathcal{F})^\beta = (\mathcal{F})^\gamma g\]
\[\text{with } \Box f g \text{ on a } \mathcal{F}^\circ\]
\[(- f \mathcal{F})^\gamma = (\mathcal{F})^\gamma \Box f\]
\[(\quad)^\gamma = -\]

\[= n \text{ d } - \text{wh what } \sim \text{ what } \vdash \Box_{\text{wh}d/ < n}\]
Lexical correspondence

\[
\begin{align*}
(f \, \mathcal{F})^\alpha &= (\mathcal{F})^\gamma \, f \\
(=f \, \mathcal{F})^\alpha &= (\mathcal{F})^\beta / < f \\
(=f \, \mathcal{F})^\beta &= f \! \setminus_\succ (\mathcal{F})^\beta \\
(f \, \mathcal{F})^\beta &= (\mathcal{F})^\gamma \, f \\
(\mathcal{g} \, \mathcal{F})^\beta &= (\mathcal{F})^\gamma \, g \\
(=f \, \mathcal{g} \, \mathcal{F})^\beta &= (\mathcal{F})^\gamma \, \mathcal{g} \\
(=n \, d \, -wh \, what) &\sim what \vdash \Box_{whd/ < n}
\end{align*}
\]

The automaton corresponds with the rules on the left side of the algorithm.
5.1. Mapping of Operations

Merge as Modus Ponens

Leftheaded MERGE:

Rightheaded MERGE:
5.1. Mapping of Operations

**Merge as Modus Ponens**

Leftheaded \texttt{MERGE}:

\[
\text{\texttt{MERGE}(t_1[=A],t_2[A])} \Rightarrow \begin{cases} \text{t}_1 \swarrow \text{t}_2 \\ \searrow \end{cases} \]

Rightheaded \texttt{MERGE}:

\[
\begin{cases} \text{t}_1 \swarrow \text{t}_2 \\ \searrow \end{cases} \Rightarrow \text{\texttt{MERGE}(t_2[=A],t_1[A])} \]
5.1. Mapping of Operations

**Merge as Modus Ponens**

**Leftheaded** \( \text{MERGE}(t_1[=A], t_2[A]) \) \( \Rightarrow \) \( t_1 \leftarrow t_2 \) 

\[ t_1 \vdash B/<A \quad t_2 \vdash A \]
\[ \frac{t_1 \Leftarrow t_2 \vdash B}{t_1 \circ < t_2 \vdash B} [/E] \]

**Rightheaded** \( \text{MERGE}(t_2[=A], t_1[A]) \) \( \Leftarrow \) \( t_1 \leftarrow t_2 \) 

\[ t_1 \vdash A \quad t_2 \vdash A\rhd B \]
\[ \frac{t_1 \rhd t_2 \vdash B}{t_1 \circ > t_2 \vdash B} \setminus [E] \]
Move as Structural Reasoning

\[\diamond f(A \triangleright B) \rightarrow \diamond fA \triangleright B\] \hspace{1cm} [P1]
\[\diamond fA \triangleright (B \bullet_i C) \rightarrow B \bullet_i (C \bullet_< \diamond fA)\] \hspace{1cm} [P2]
\[\diamond fA \triangleright (B \bullet_i C) \rightarrow B \bullet_i (\diamond fA \triangleright C)\] \hspace{1cm} [P3]

Where \(i \in \{<, >\}\)
Move as Structural Reasoning

\[ \diamond f(A \cdot \triangleright B) \rightarrow \diamond f A \cdot \triangleright B \quad [P1] \]
\[ \diamond f A \cdot \triangleright (B \cdot_i C) \rightarrow B \cdot_i (C \cdot_\langle \diamond f A) \quad [P2] \]
\[ \diamond f A \cdot \triangleright (B \cdot_i C) \rightarrow B \cdot_i (\diamond f A \cdot \triangleright C) \quad [P3] \]

Where \( i \in \{<, >\} \)

- Feature information of the specifier projects over the head of the tree
Move as Structural Reasoning

\[ \Diamond f(A \bullet_\Rightarrow B) \rightarrow \Diamond fA \bullet_\Rightarrow B \quad [P1] \]
\[ \Diamond fA \bullet_\Rightarrow (B \bullet_i C) \rightarrow B \bullet_i (C \bullet_\prec \Diamond fA) \quad [P2] \]
\[ \Diamond fA \bullet_\Rightarrow (B \bullet_i C) \rightarrow B \bullet_i (\Diamond fA \bullet_\Rightarrow C) \quad [P3] \]

Where \( i \in \{<, >\} \)

- Feature information of the specifier projects over the head of the tree
- Under control of the licensor feature (\( \Diamond_m \)) the specifier moves to the Spec position of the head \( A \)
6. A deductive approach to VM-climbing

Feature information as lexical formulas

\[ \text{akarok} \vdash v_{fin}/<v_{inf} \]
\[ \text{menni} \vdash v_{inf}/<m \]

\text{MOVE as structural reasoning:}
\[ \text{haza} \vdash \square_{m} m \]
6. A deductive approach to VM-climbing

Feature information as lexical formulas

\[ \text{akarok} \vdash v_{\text{fin}} < v_{\text{inf}} \]
\[ \text{menni} \vdash v_{\text{inf}} < m \]
\[ \text{haza} \vdash \Box_m m \]
\[ \text{haza} \vdash v_{\text{inf}} > (\Diamond_m \Box_m m \setminus v_{\text{inf}}) \]
6. A deductive approach to VM-climbing

Feature information as lexical formulas

MOVE as structural reasoning:
- akarok ⊢ v_{fin} < v_{inf}
- menni ⊢ v_{inf} < m
- haza ⊢ \Box_{m} m

MOVE as abstraction:
- haza ⊢ v_{inf} > (\Diamond_{m} \Box_{m} m > v_{inf})

MERGE over MOVE:
- haza ⊢ \Box_{m} (v_{inf} > (v_{inf} < m))
6. A deductive approach to VM-climbing

Feature information as lexical formulas

\[
\begin{align*}
\text{MOVE as structural reasoning:} & \quad \text{haza} \vdash \Box_m m \\
\text{MOVE as abstraction:} & \quad \text{haza} \vdash v_{inf}/>(\diamond_m \Box_m m \backslash v_{inf}) \\
\text{MERGE over MOVE:} & \quad \text{haza} \vdash \Box_m (v_{inf}/>(v_{inf}/<m))
\end{align*}
\]

- Basic ‘English’ order: * akarok menni haza
- ‘Inverted’ order: * akarok haza menni
- haza always precedes its selecting verb
- “MERGE over MOVE”
Structural postulates for VM-climbing

\[ \Diamond_m (A \rightarrow B) \rightarrow \Diamond_m A \rightarrow B \quad [K1m] \]
\[ \Diamond_m A \rightarrow (B < C) \rightarrow B < (C < \Diamond_m A) \quad [Mm1] \]
\[ \Diamond_m A \rightarrow (B < C) \rightarrow B < (\Diamond_m A \rightarrow C) \quad [Mm2] \]
Structural postulates for VM-climbing

\[ \diamond_m (A \mathbin{\rightarrow} B) \to \diamond_m A \mathbin{\rightarrow} B \quad [K1m] \]
\[ \diamond_m A \mathbin{\rightarrow} (B \mathbin{\leftarrow} C) \to B \mathbin{\leftarrow} (C \mathbin{\leftarrow} \diamond_m A) \quad [Mm1] \]
\[ \diamond_m A \mathbin{\rightarrow} (B \mathbin{\leftarrow} C) \to B \mathbin{\leftarrow} (\diamond_m A \mathbin{\rightarrow} C) \quad [Mm2] \]

- [K1m] = Feature distribution
- [Mm1] = Movement of Comp → Spec
- [Mm2] = Movement of Spec → Spec
Compare this analysis to the $MG$ derivation.

(1) Lexicon: 

\[
\text{haza} \vdash \Box_m (v_{inf} > (v_{inf} / m))
\]
Compare this analysis to the $\mathcal{MG}$ derivation.

(1) Lexicon:

$$\text{haza} \vdash \Box_m (v_{inf}/>(v_{inf}/m))$$

(2) $[\Box_m E]$

$$\Diamond_m \downarrow \text{haza} \vdash v_{inf}/>(v_{inf}/m)$$
Compare this analysis to the MG derivation.

(1) Lexicon: $\text{haza} \vdash \Diamond_m (\text{v}_{\text{inf}}/>(\text{v}_{\text{inf}}/m))$

(2) $[\Box_m E]$:

\[
\Diamond_m \\
| \\
\text{haza} \\
\vdash \text{v}_{\text{inf}}/>(\text{v}_{\text{inf}}/m)
\]

(3) Lexicon: $\text{menni} \vdash \text{v}_{\text{inf}}/m$
Compare this analysis to the MG derivation.

(1) Lexicon:

\[ \text{haza} \vdash \Box_m (v_{inf} > (v_{inf}/m)) \]

(2) \([\Box_m \downarrow E]\):

\[ \text{haza} \vdash v_{inf} > (v_{inf}/m) \]

(3) Lexicon:

\[ \text{menni} \vdash v_{inf}/m \]

(4) \([/E], \text{MERGE}(2,3)\):

\[ \text{haza} \vdash v_{inf} \]
(5) Lexicon: 

\[ \text{akarok} \vdash v_{\text{fin}} / v_{\text{inf}} \]
(5) Lexicon:

\[
\begin{array}{c}
\text{akarok} \\
\vdash v_{\text{fin}}/v_{\text{inf}}
\end{array}
\]

(6) \([/E], \text{MERGE}(4,5)\):

\[
\begin{array}{c}
n\text{menni} \\
\vdash v_{\text{fin}}
\end{array}
\]

\[
\begin{array}{c}
\text{haza}
\end{array}
\]
(5) Lexicon:

\[
\begin{array}{c}
\text{akarok} \\
\vdash v_{\text{fin}}/v_{\text{inf}}
\end{array}
\]

(6) \( /E \), \text{Merge}(4,5):

\[
\begin{array}{c}
\text{menni} \\
\vdash v_{\text{fin}}
\end{array}
\]

\[
\begin{array}{c}
\text{haza}
\end{array}
\]

(7) \text{Move}(6)

\[
\begin{array}{c}
\text{haza} \\
\text{akarok} \\
\text{menni}
\end{array}
\]

\[
\begin{array}{c}
\vdash v_{\text{fin}}
\end{array}
\]
(8) $FD(7)$:

\[
\begin{array}{c}
\diamond_m \\
\downarrow \\
> \\
\hline
\text{haza} < \\
\hline
\text{akarok} \quad \text{menni}
\end{array}
\quad \vdash \quad v_{fin}
\]
(8) \( FD(7) \):

\[
\vdash v_{\text{fin}}
\]

(9) \([\Box_m I]\):

\[
\vdash \Box_m v_{\text{fin}}
\]
7. Conclusion

- Licensee features are triggers for movement
7. Conclusion

- Licensee features are triggers for movement
- Features are defined as lexical formulas
7. Conclusion

- Licensee features are triggers for movement
- Features are defined as lexical formulas
- MOVE is not a primitive operation
7. Conclusion

• Licensee features are triggers for movement

• Features are defined as lexical formulas

• MOVE is **not** a primitive operation

• MOVE is built up of Structural Reasoning and Hypothetical Reasoning w.r.t. Merge
Controlling Movement: Minimalism in a deductive perspective

Willemijn Vermaat, Universiteit Utrecht
April 1999

willemijn.vermaat@let.uu.nl

ftp://ftp.let.uu.nl/pub/users/Willemijn.Vermaat