Constraints Emerge from Merge

Thomas Graf
tgraf@ucla.edu
tgraf.bol.ucla.edu

University of California, Los Angeles

February 12, 2013
Take-Home Message

Questions

- How do constraints fit into syntax?
- What are their properties?

This Talk

- Constraints are closely related to operations.
- A formal perspective makes this connection explicit.
- Linking constraints to operations limits their power and makes new empirical predictions.
Questions

- How do constraints fit into syntax?
- What are their properties?

This Talk

- Constraints are closely related to operations.
- A formal perspective makes this connection explicit.
- Linking constraints to operations limits their power and makes new empirical predictions.
1. The Status of Constraints in Linguistics
2. Formal Concepts
   - Minimalist Grammars
   - Constraints as Logical Formulas
3. Results
   - Main Result: Constraints $\equiv$ Merge
   - Corollary: Uniformity of Constraint Classes
   - Linguistic Implications
4. Are MSO-Constraints Enough? A Look at Binding
   - Syntactic Binding: No Semantics, No Discourse
   - Computing Principle B
   - English
   - American Sign Language (ASL)
5. Conclusion & Outlook
Constraints and Operations

The two essential tools of linguistics: **Constraints** and **Operations**

- GB
- Minimalism
- TAG
- HPSG
- EST
- CCG
- LFG
- Gov. Phon.
- SPE
- OT
- Harm. Serial.

Some Naive Questions

- What distinguishes constraints from operations?
  - (Kisseberth 1970; Brody 2002; Epstein and Seely 2002)
- Is one superior to the other?
  - (Epstein et al. 1998; Bailyn 2010)
- How do they interact?
  - (Pullum and Scholz 2001, 2005)
- Does it make a difference for empirical work?
The two essential tools of linguistics: **Constraints** and **Operations**

- GB Minimalism TAG
- HPSG EST CCG
- LFG Gov. Phon. SPE
- OT Harm. Serial.

**Some Naive Questions**

- What distinguishes constraints from operations?
- Is one superior to the other?  
  (Kisseberth 1970; Brody 2002; Epstein and Seely 2002)
- How do they interact?  
  (Epstein et al. 1998; Bailyn 2010)
- Does it make a difference for empirical work?  
  (Pullum and Scholz 2001, 2005)
Müller-Sternefeld Hierarchy of Constraints

- **Representational** (output filters/interface conditions)
  - stated over phrase structure trees
  - ECP: “Every trace is properly governed at LF.”
  - Full Interpretation: “No uninterpretable features at LF.”
  - Linearizability: “All leaves are linearly ordered at PF.”

- **Derivational**
  - stated over derivations
  - Relativized Minimality: “\( X \) moves to \( Z \) only if there is no \( Y \) closer to \( Z \) that could have moved there.”

- **Transderivational** (economy conditions; cf. OT)
  - picks optimal tree out of set of competing candidates
  - Shortest Derivation Principle: “Given a set of competing derivations, pick the one with the fewest instances of Move.”

MS-Hierarchy (Müller and Sternefeld 2000; Müller 2005)

representational < derivational < transderivational
Shortcomings of the MS-Hierarchy

- based on case studies of specific constraints ⇒ lack of generality
- no clear characterization of power of constraints
- Why do constraints exist in the first place?
  What does this tell us about language?
Outline

1. The Status of Constraints in Linguistics

2. Formal Concepts
   - Minimalist Grammars
   - Constraints as Logical Formulas

3. Results
   - Main Result: Constraints $\equiv$ Merge
   - Corollary: Uniformity of Constraint Classes
   - Linguistic Implications

4. Are MSO-Constraints Enough? A Look at Binding
   - Syntactic Binding: No Semantics, No Discourse
   - Computing Principle B
   - English
   - American Sign Language (ASL)

5. Conclusion & Outlook
Minimalist Grammars (MGs)

- formalization of Minimalist syntax without Agree/phases (Stabler 1997)
- many extensions to make them more faithful (Frey and Gärtner 2002; Graf 2012b; Kobele 2002, 2012; Stabler 2003, 2006, 2011, among others)
- original version suffices for our purposes

Core Idea of MGs

- Operations: **Merge** and **Move**
- lexical items annotated with features
- features come in two polarities
- each operation must check two features of opposite polarity
Minimalist Grammars (MGs)

- formalization of Minimalist syntax without Agree/phases (Stabler 1997)
- many extensions to make them more faithful (Frey and Gärtner 2002; Graf 2012b; Kobele 2002, 2012; Stabler 2003, 2006, 2011, among others)
- original version suffices for our purposes

Core Idea of MGs

- Operations: **Merge** and **Move**
- lexical items annotated with features
- features come in two polarities
- each operation must check two features of opposite polarity
Merge: Example 1

Assembling \([\text{DP the men}]\)

- Merge triggered by features of opposite polarities
- Label points to branch leading to projecting head
- Head must have a category feature \((\text{N}^-, \text{D}^-, \text{V}^-, \ldots)\)
- Derivation tree differs only with respect to labels
Merge: Example 1

Assembling $[\text{DP the men}]$

- Merge triggered by features of opposite polarities
- Label points to branch leading to projecting head
- Head must have a category feature ($N^-, D^-, V^-, \ldots$)
- Derivation tree differs only with respect to labels
Merge: Example 1

Assembling $[\text{DP the men}]$

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>men</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{N}^+$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{N}^-$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Merge triggered by features of opposite polarities
- Label points to branch leading to projecting head
- Head must have a category feature ($\text{N}^-, \text{D}^-, \text{V}^-, \ldots$)
- Derivation tree differs only with respect to labels
Merge: Example 1

Assembling $[\text{DP the men}]$

- Merge triggered by features of opposite polarities
- Label points to branch leading to projecting head
- Head must have a category feature ($N^-, D^-, V^-, \ldots$)
- Derivation tree differs only with respect to labels
Merge: Example 1

Assembling \([_{DP} \text{ the men}]\]

- Merge triggered by features of opposite polarities
- Label points to branch leading to projecting head
- Head must have a category feature (\(N^-, D^-, V^-, \ldots\))
- Derivation tree differs only with respect to labels
Merge: Example 1

Assembling $[[\text{DP the men}]]$

- Merge triggered by features of opposite polarities
- Label points to branch leading to projecting head
- Head must have a category feature ($N^-, D^-, V^-, \ldots$)
- Derivation tree differs only with respect to labels
Merge: Example 1

Assembling $[\text{DP the men}]$

- Merge triggered by features of opposite polarities
- Label points to branch leading to projecting head
- Head must have a category feature ($N^-, D^-, V^-, \ldots$)
- Derivation tree differs only with respect to labels
Merge: Example 1

Assembling $[\text{DP the men}]$

- Merge triggered by features of opposite polarities
- Label points to branch leading to projecting head
- **Head must have a category feature** ($N^-, D^-, V^-, \ldots$)
- Derivation tree differs only with respect to labels
Merge: Example 1

Assembling \([\text{DP the men}]\)

- Merge triggered by features of opposite polarities
- Label points to branch leading to projecting head
- Head must have a category feature \((N^-, D^-, V^-, \ldots)\)
- Derivation tree differs only with respect to labels
Assembling \([DP \text{ the men}]\)

- Merge triggered by features of opposite polarities
- Label points to branch leading to projecting head
- Head must have a category feature (\(N^-, D^-, V^-, \ldots\))
- Derivation tree differs only with respect to labels
Merge: Example 2

Assembling $[\text{VP the men like which men}]$

- *the* and *men* merged as before
- same steps for *which men*
- *like* selects *which men*
- *like* selects *the men*
- *like* needs a category feature
Merge: Example 2

Assembling $[\text{VP the men like which men}]$

- the  men  like
- which  men

- the and men merged as before
- same steps for which men
- like selects which men
- like selects the men
- like needs a category feature
### Merge: Example 2

**Assembling \([VP \text{ the men like which men}]\)

<table>
<thead>
<tr>
<th>the</th>
<th>men</th>
<th>like</th>
<th>which</th>
<th>men</th>
</tr>
</thead>
<tbody>
<tr>
<td>N(^+) D(^-)</td>
<td>N(^-)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- *the* and *men* merged as before
- same steps for *which men*
- *like* selects *which men*
- *like* selects *the men*
- *like* needs a category feature
Assembling $[\text{VP the men like which men}]$

- the and men merged as before
- same steps for which men
- like selects which men
- like selects the men
- like needs a category feature
### Merge: Example 2

#### Assembling $[\text{VP the men like which men}]$

- $\text{the}$ and $\text{men}$ merged as before
- *same steps for which men*
- *like selects which men*
- *like selects the men*
- *like needs a category feature*
Assembling \[\text{[VP the men like which men]}\]

- The and men merged as before
- Same steps for which men
- Like selects which men
- Like selects the men
- Like needs a category feature
Merge: Example 2

Assembling $[\text{VP the men like which men}]$

- The and men merged as before
- Same steps for which men
- Like selects which men
- Like selects the men
- Like needs a category feature
Merge: Example 2

Assembling \([\text{VP the men like which men}]\)

- \(\text{the}\) and \(\text{men}\) merged as before
- same steps for \(\text{which men}\)
- \text{like} selects \(\text{which men}\)
- \text{like} selects \(\text{the men}\)
- \text{like} needs a category feature
Assembling $[\text{VP the men like which men}]$:

- The and men merged as before.
- Same steps for which men.
- Like selects which men.
- Like selects the men.
- Like needs a category feature.
Merge: Example 2

Assembling \([\text{VP the men like which men}]\)

- **the** and **men** merged as before
- same steps for **which men**
- **like** selects **which men**
- **like** selects **the men**
- **like** needs a category feature
Merge: Example 2

Assembling \([VP \ the \ men \ like \ which \ men]\)

- The and men merged as before
- Same steps for which men
- Like selects which men
- Like selects the men
- Like needs a category feature
Assembling $[VP \text{ the } men \text{ like } which \text{ men}]$

- **the** and **men** merged as before
- **same steps for** **which men**
- **like** selects **which men**
- **like** selects **the men**
- **like** needs a category feature
Merge: Example 2

Assembling \([_{VP} \text{ the men like which men}]\)

- \(\text{the} \) and \(\text{men}\) merged as before
- same steps for \(\text{which men}\)
- \(\text{like}\) selects \(\text{which men}\)
- \(\text{like}\) selects \(\text{the men}\)
- \(\text{like}\) needs a category feature
Assembling \([\text{VP the men like which men}]\)

- \(\text{the}\) and \(\text{men}\) merged as before
- same steps for \(\text{which men}\)
- \(\text{like}\) selects \(\text{which men}\)
- \(\text{like}\) selects \(\text{the men}\)
- \(\text{like}\) needs a category feature
Assembling \([VP \ the \ men \ like \ which \ men]\)

- the and men merged as before
- same steps for which men
- like selects which men
- like selects the men
- like needs a category feature
Merge: Example 2 [cont.]

the  men  like
N⁺  D⁻  N⁻  D⁺  D⁺  V⁻

which  men
N⁺  D⁻  N⁻
Move (Multi-Dominance Implementation)

Assembling “which men do the men like?”

- Merge *do*
- Move triggered by features of opposite polarity
- *do* must have a category feature
Move (Multi-Dominance Implementation)

Assembling “which men do the men like?”

- Merge *do*
- Move triggered by features of opposite polarity
- *do* must have a category feature
Move (Multi-Dominance Implementation)

Assembling “which men do the men like?”

- **Merge** *do*
- Move triggered by features of opposite polarity
- *do* must have a category feature
Move (Multi-Dominance Implementation)

Assembling “which men do the men like?”

- **Merge** *do*
- Move triggered by features of opposite polarity
- *do* must have a category feature
Move (Multi-Dominance Implementation)

Assembling “which men do the men like?”

- Merge *do*
- Move triggered by features of opposite polarity
- *do* must have a category feature
Move (Multi-Dominance Implementation)

Assembling “which men do the men like?”

- **Merge** *do*
- **Move triggered by features of opposite polarity**
- *do* must have a category feature
Assembling “which men do the men like?”

- **Merge** *do*
- **Move triggered by features of opposite polarity**
- *do* must have a category feature
Move (Multi-Dominance Implementation)

Assembling “which men do the men like?”

- Merge *do*
- Move triggered by features of opposite polarity
- *do* must have a category feature
Move (Multi-Dominance Implementation)

Assembling “which men do the men like?”

- Merge *do*
- Move triggered by features of opposite polarity
- *do* must have a category feature
Derivation Trees with Move

\[ \text{do} \quad < \quad \text{the} \quad \text{men} \quad < \quad \text{like} \quad \text{which} \quad \text{men} \]

\[ \text{V}^+ \text{wh}^+ \text{C}^- \]

Move[wh]

\[ \text{Merge}[V] \]

\[ \text{do} \]

\[ \text{V}^+ \text{wh}^+ \text{C}^- \]

\[ \text{Merge}[N] \]

\[ \text{the} \quad \text{men} \]

\[ \text{N}^+ \text{D}^- \quad \text{N}^- \]

\[ \text{Merge}[D] \]

\[ \text{like} \]

\[ \text{D}^+ \text{D}^+ \text{V}^- \]

\[ \text{N}^+ \text{D}^- \text{wh}^- \quad \text{N}^- \]

\[ \text{Merge}[D] \]

\[ \text{which} \]

\[ \text{N}^+ \text{D}^- \text{wh}^- \quad \text{N}^- \]

\[ \text{Merge}[N] \]

\[ \text{men} \]
An MG is given by a **set of feature-annotated lexical items**. It generates all (multi-dominance) trees that are CPs built from the available lexical items.

**Example**

- **men**: \[N^-\]
- **the**: \[D^+ N^-\]
- **which**: \[D^+ N^- wh^-\]
- **like**: \[D^+ D^+ V^-\]
- **do**: \[V^+ wh^+ C^-\]
- **ε**: \[V^+ C^-\]

**Generated sentences:**
- The men like the men.
- Which men do the men like.
- Which men do like the men.
Formalizing Constraints

Constraint statement \( c \) that must be satisfied in order for a tree to be well-formed

Logical Formula statement \( \phi \) that must be satisfied in order for a structure to be a model of \( \phi \)

\[ \Rightarrow \text{Constraints} \equiv \text{Logical Formulas} \]

(Kracht 1995; Rogers 1998; Potts 2001; Pullum 2007)

First-Order Logic for Trees (FO)

\( x, y, z, \ldots \) variables \( x, y, z, \ldots \)
\( \land, \lor, \lnot, \rightarrow, \leftrightarrow \) and, or, not, implies, iff
\( \exists, \forall \) there is, for all
\( l(x) \) \( x \) has label \( l \)
\( \triangleleft \) dominance
\( \approx \) equivalence
Formalizing Constraints

Constraint statement $c$ that must be satisfied in order for a tree to be well-formed

Logical Formula statement $\phi$ that must be satisfied in order for a structure to be a model of $\phi$

$\Rightarrow$ Constraints $\equiv$ Logical Formulas

(Kracht 1995; Rogers 1998; Potts 2001; Pullum 2007)

First-Order Logic for Trees (FO)

- $x, y, z, \ldots$ variables $x, y, z, \ldots$
- $\land, \lor, \neg, \rightarrow, \leftrightarrow$ and, or, not, implies, iff
- $\exists, \forall$ there is, for all
- $l(x)$ $x$ has label $l$
- $\triangle$ dominance
- $\approx$ equivalence
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[
anaphor(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

\[
c-com(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \text{anaphor}(x) \leftrightarrow \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x) \]

\[ \text{c-com}(x, y) \leftrightarrow \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y] \]
Example: Stating Principle A

Principle A (slightly simplified)
Every anaphor must be c-commanded by some DP within its binding domain.

\[ \text{anaphor}(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x) \]

"x is an anaphor

\[ \text{c-com}(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \implies z \triangleleft y] \]
Example: Stating Principle A

**Principle A (slightly simplified)**

Every anaphor must be c-commanded by some DP within its binding domain.

\[
anaphor(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff

\[
c-\text{com}(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \text{anaphor}(x) \leftrightarrow \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x) \]

“\( x \) is an anaphor iff \( x \) is labeled \textit{himself}"

\[ \text{c-com}(x, y) \leftrightarrow \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y] \]
Example: Stating Principle A

**Principle A (slightly simplified)**

Every anaphor must be c-commanded by some DP within its binding domain.

\[
\text{anaphor}(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself}"

or

\[
\text{c-com}(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]
Example: Stating Principle A

Principle A (slightly simplified)
Every anaphor must be c-commanded by some DP within its binding domain.

\[
anaphor(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself} or \(x\) is labeled \textit{herself}.

\[
c\text{-com}(x, y) \iff \neg (x \equiv y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[
\text{anaphor}(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself}

or \(x\) is labeled \textit{herself} or

\[
\text{c-com}(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z [z \triangleleft x \rightarrow z \triangleleft y]
\]
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[
anaphor(x) \leftrightarrow \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself}

or \(x\) is labeled \textit{herself} or \(x\) is labeled \textit{itself}.”

\[
c-com(x, y) \leftrightarrow \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]
Example: Stating Principle A

**Principle A (slightly simplified)**

Every anaphor must be c-commanded by some DP within its binding domain.

\[
anaphor(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself}

or \(x\) is labeled \textit{herself} or \(x\) is labeled \textit{itself}.”

\[
c-\text{com}(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[
anaphor(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself}

or \(x\) is labeled \textit{herself} or \(x\) is labeled \textit{itself}.”

\[
c-com(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]

“\(x\) c-commands \(y\)”
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[
\text{anaphor}(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself}

or \(x\) is labeled \textit{herself} or \(x\) is labeled \textit{itself}.”

\[
\text{c-com}(x, y) \iff \neg (x \approx y) \land \neg (x \lhd y) \land \forall z [z \lhd x \rightarrow z \lhd y]
\]

“\(x\) c-commands \(y\) iff

---

12
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \text{anaphor}(x) \leftrightarrow \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x) \]

“\( x \) is an anaphor iff \( x \) is labeled \textit{himself}

or \( x \) is labeled \textit{herself} or \( x \) is labeled \textit{itself}.”

\[ \text{c-com}(x, y) \leftrightarrow \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y] \]

“\( x \) c-commands \( y \) iff

it is not the case that
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[
anaphor(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \text{himself}

or \(x\) is labeled \text{herself} or \(x\) is labeled \text{itself}.”

\[
c-com(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]

“\(x\) c-commands \(y\) iff

it is not the case that \(x\) and \(y\) are the same node,”
Example: Stating Principle A

**Principle A (slightly simplified)**

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \text{anaphor}(x) \leftrightarrow \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x) \]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself}

or \(x\) is labeled \textit{herself} or \(x\) is labeled \textit{itself}.”

\[ \text{c-com}(x, y) \leftrightarrow \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y] \]

“\(x\) c-commands \(y\) iff

it is not the case that \(x\) and \(y\) are the same node,

and
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \text{anaphor}(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x) \]

“\( x \) is an anaphor iff \( x \) is labeled \textit{himself}

or \( x \) is labeled \textit{herself} or \( x \) is labeled \textit{itself}."

\[ \text{c-com}(x, y) \iff \neg (x \approx y) \land \neg (x \lessdot y) \land \forall z \ [z \lessdot x \rightarrow z \lessdot y] \]

“\( x \) c-commands \( y \) iff

it is not the case that \( x \) and \( y \) are the same node,

and \textit{it is not the case that}
Example: Stating Principle A

**Principle A (slightly simplified)**

Every anaphor must be c-commanded by some DP within its binding domain.

\[
anaphor(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \(\text{himself}\) or \(x\) is labeled \(\text{herself}\) or \(x\) is labeled \(\text{itself}\).”

\[
c-com(x, y) \iff \neg(x \approx y) \land \neg(x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]

“\(x\) c-commands \(y\) iff
it is not the case that \(x\) and \(y\) are the same node, and it is not the case that \(x\) dominates \(y\),
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[
anaphor(x) \leftrightarrow \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself}

or \(x\) is labeled \textit{herself} or \(x\) is labeled \textit{itself}.”

\[
c-com(x, y) \leftrightarrow \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]

“\(x\) c-commands \(y\) iff

it is not the case that \(x\) and \(y\) are the same node,

and it is not the case that \(x\) dominates \(y\),

and
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[
anaphor(x) \leftrightarrow \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself}
or \(x\) is labeled \textit{herself} or \(x\) is labeled \textit{itself}.”

\[
c-com(x, y) \leftrightarrow \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [(z \triangleleft x \rightarrow z \triangleleft y]
\]

“\(x\) c-commands \(y\) iff
it is not the case that \(x\) and \(y\) are the same node,
and it is not the case that \(x\) dominates \(y\),
and every \(z\)”
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[
\text{anaphor}(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x)
\]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself}

or \(x\) is labeled \textit{herself} or \(x\) is labeled \textit{itself}.”

\[
\text{c-com}(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y]
\]

“\(x\) c-commands \(y\) iff

it is not the case that \(x\) and \(y\) are the same node,

and it is not the case that \(x\) dominates \(y\),

and every \(z\) that dominates \(x\)
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \text{anaphor}(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x) \]

“\(x\) is an anaphor iff \(x\) is labeled \(\text{himself}\) or \(x\) is labeled \(\text{herself}\) or \(x\) is labeled \(\text{itself}\).”

\[ \text{c-com}(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y] \]

“\(x\) c-commands \(y\) iff
it is not the case that \(x\) and \(y\) are the same node,
and it is not the case that \(x\) dominates \(y\),
and every \(z\) that dominates \(x\) also
Example: Stating Principle A

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \text{anaphor}(x) \iff \text{himself}(x) \lor \text{herself}(x) \lor \text{itself}(x) \]

“\(x\) is an anaphor iff \(x\) is labeled \textit{himself} or \(x\) is labeled \textit{herself} or \(x\) is labeled \textit{itself}.”

\[ \text{c-com}(x, y) \iff \neg (x \approx y) \land \neg (x \triangleleft y) \land \forall z \ [z \triangleleft x \rightarrow z \triangleleft y] \]

“\(x\) c-commands \(y\) iff
\(x\) is not the case that \(x\) and \(y\) are the same node, and \(x\) dominates \(y\), and every \(z\) that dominates \(x\) also dominates \(y\).”
Set Quantification: Talking About Domains

**Problem:** domains are _sets of nodes_ ⇒ move beyond FO

---

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with:

- \( X, Y, Z, \ldots \) set variables \( X, Y, Z, \ldots \)
- \( \text{YP}(X) \) set \( X \) is a YP
- \( \exists, \forall \) there is a set, for all sets
- \( \in, \subset \) set containment, proper subset

\[
b\text{-dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z[y \in Z \land \text{TP}(Z) \land Z \subset X]
\]
Set Quantification: Talking About Domains

**Problem**: domains are **sets of nodes** ⇒ move beyond FO

### Monadic Second-Order Logic for Trees (MSO)

Extension of FO with

- $X, Y, Z, \ldots$ set variables $X, Y, Z, \ldots$
- $YP(X)$ set $X$ is a YP
- $\exists, \forall$ there is a set, for all sets
- $\in, \subset$ set containment, proper subset

\[ b\text{-dom}(X, y) \iff TP(X) \land y \in X \land \neg \exists Z[y \in Z \land TP(Z) \land Z \subset X] \]
Set Quantification: Talking About Domains

**Problem:** domains are *sets of nodes* ⇒ move beyond FO

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with

- \( X, Y, Z, \ldots \) set variables \( X, Y, Z, \ldots \)
- \( \text{YP}(X) \) set \( X \) is a YP
- \( \exists, \forall \) there is a set, for all sets
- \( \in, \subset \) set containment, proper subset

\[
\text{b-dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z[y \in Z \land \text{TP}(Z) \land Z \subset X]
\]

“\( X \) is the binding domain of \( y \)”
Set Quantification: Talking About Domains

**Problem**: domains are *sets of nodes* ⇒ move beyond FO

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with

- \( X, Y, Z, \ldots \) set variables \( X, Y, Z, \ldots \)
- \( \text{YP}(X) \) set \( X \) is a YP
- \( \exists, \forall \) there is a set, for all sets
- \( \in, \subset \) set containment, proper subset

\[
b\text{-dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z[y \in Z \land \text{TP}(Z) \land Z \subset X]
\]

“\( X \) is the binding domain of \( y \) iff
Set Quantification: Talking About Domains

**Problem:** domains are **sets of nodes** ⇒ move beyond FO

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with

- $X, Y, Z, \ldots$ set variables $X, Y, Z, \ldots$
- $\text{YP}(X)$ set $X$ is a YP
- $\exists, \forall$ there is a set, for all sets
- $\in, \subset$ set containment, proper subset

\[ b\text{-dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z [y \in Z \land \text{TP}(Z) \land Z \subset X] \]

“$X$ is the binding domain of $y$ iff $X$ is a TP”
Set Quantification: Talking About Domains

**Problem:** domains are *sets of nodes* ⇒ move beyond FO

**Monadic Second-Order Logic for Trees (MSO)**

**Extension of FO with**

- $X, Y, Z, \ldots$ set variables $X, Y, Z, \ldots$
- $\text{YP}(X)$ set $X$ is a YP
- $\exists, \forall$ there is a set, for all sets
- $\in, \subset$ set containment, proper subset

**Definition:**

\[
\text{b-dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z[y \in Z \land \text{TP}(Z) \land Z \subset X]
\]

"$X$ is the binding domain of $y$ iff $X$ is a TP and"
Set Quantification: Talking About Domains

**Problem:** domains are **sets of nodes** ⇒ move beyond FO

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with:

- $X, Y, Z, \ldots$ set variables $X, Y, Z, \ldots$
- $YP(X)$ set $X$ is a YP
- $\exists, \forall$ there is a set, for all sets
- $\in, \subset$ set containment, proper subset

**b-dom($X, y$) $\iff$**

$$TP(X) \land y \in X \land \neg \exists Z[y \in Z \land TP(Z) \land Z \subset X]$$

"$X$ is the binding domain of $y$ iff $X$ is a TP and $X$ contains $y"
Set Quantification: Talking About Domains

**Problem:** domains are **sets of nodes** ⇒ move beyond FO

---

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with

- \(X, Y, Z, \ldots\) set variables \(X, Y, Z, \ldots\)
- \(YP(X)\) set \(X\) is a YP
- \(\exists, \forall\) there is a set, for all sets
- \(\in, \subset\) set containment, proper subset

\[
b\text{-}\text{dom}(X, y) \iff TP(X) \land y \in X \land \neg\exists Z[y \in Z \land TP(Z) \land Z \subset X]
\]

“\(X\) is the binding domain of \(y\) iff \(X\) is a TP and \(X\) contains \(y\) and
Set Quantification: Talking About Domains

**Problem:** domains are sets of nodes ⇒ move beyond FO

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with

- \( X, Y, Z, \ldots \) set variables \( X, Y, Z, \ldots \)
- \( \text{YP}(X) \) set \( X \) is a \( \text{YP} \)
- \( \exists, \forall \) there is a set, for all sets
- \( \in, \subset \) set containment, proper subset

\[
\text{b-dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z[y \in Z \land \text{TP}(Z) \land Z \subset X]
\]

“\( X \) is the binding domain of \( y \) iff \( X \) is a \( \text{TP} \) and \( X \) contains \( y \) and there is no \( Z \) that
Set Quantification: Talking About Domains

**Problem:** domains are **sets of nodes** ⇒ move beyond FO

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with:

- $X, Y, Z, \ldots$ set variables $X, Y, Z, \ldots$
- $\text{YP}(X)$ set $X$ is a YP
- $\exists, \forall$ there is a set, for all sets
- $\in, \subset$ set containment, proper subset

\[
b-\text{dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z[y \in Z \land \text{TP}(Z) \land Z \subset X]
\]

“$X$ is the binding domain of $y$ iff $X$ is a TP and $X$ contains $y$ and there is no $Z$ that contains $y$”
Set Quantification: Talking About Domains

Problem: domains are sets of nodes ⇒ move beyond FO

Monadic Second-Order Logic for Trees (MSO)

Extension of FO with

\[ X, Y, Z, \ldots \quad \text{set variables } X, Y, Z, \ldots \]
\[ \text{YP}(X) \quad \text{set } X \text{ is a YP} \]
\[ \exists, \forall \quad \text{there is a set, for all sets} \]
\[ \in, \subset \quad \text{set containment, proper subset} \]

\[ \text{b-dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z[y \in Z \land \text{TP}(Z) \land Z \subset X] \]

“\( X \) is the binding domain of \( y \) iff \( X \) is a TP and \( X \) contains \( y \) and there is no \( Z \) that contains \( y \) and
Set Quantification: Talking About Domains

**Problem:** domains are **sets of nodes** ⇒ move beyond FO

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with

- \( X, Y, Z, \ldots \) set variables \( X, Y, Z, \ldots \)
- \( \text{YP}(X) \) set \( X \) is a YP
- \( \exists, \forall \) there is a set, for all sets
- \( \in, \subset \) set containment, proper subset

\[
b\text{-dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z [y \in Z \land \text{TP}(Z) \land Z \subset X]
\]

“\( X \) is the binding domain of \( y \) iff \( X \) is a TP and \( X \) contains \( y \) and there is no \( Z \) that contains \( y \) and is a TP
Set Quantification: Talking About Domains

**Problem:** domains are *sets of nodes* ⇒ move beyond FO

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with

- $X, Y, Z, \ldots$ set variables $X, Y, Z, \ldots$
- $\text{YP}(X)$ set $X$ is a YP
- $\exists, \forall$ there is a set, for all sets
- $\in, \subset$ set containment, proper subset

\[
\text{b-dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z[y \in Z \land \text{TP}(Z) \land Z \subset X]
\]

"$X$ is the binding domain of $y$ iff $X$ is a TP and $X$ contains $y$ and there is no $Z$ that contains $y$ and is a TP and"
Set Quantification: Talking About Domains

**Problem:** domains are *sets of nodes* $\Rightarrow$ move beyond FO

**Monadic Second-Order Logic for Trees (MSO)**

Extension of FO with

- $X, Y, Z, \ldots$ set variables $X, Y, Z, \ldots$
- $\text{YP}(X)$ set $X$ is a YP
- $\exists, \forall$ there is a set, for all sets
- $\in, \subset$ set containment, proper subset

\[
b\text{-dom}(X, y) \iff \text{TP}(X) \land y \in X \land \neg \exists Z[y \in Z \land \text{TP}(Z) \land Z \subset X]
\]

"$X$ is the binding domain of $y$ iff $X$ is a TP and $X$ contains $y$ and there is no $Z$ that contains $y$ and is a TP and is a proper subset of $X$."

13
Principle A Again

Principle A (slightly simplified)
Every anaphor must be c-commanded by some DP within its binding domain.

∀x [anaphor(x) → ∃y [c-com(y, x) ∧ DP(y) ∧ ∃Z [b-dom(Z, x) ∧ y ∈ Z]]]
Principle A Again

Principle A (slightly simplified)
Every anaphor must be c-commanded by some DP within its binding domain.

∀x[anaphor(x) → \exists y [c-com(y, x) ∧ DP(y) ∧ 
\exists Z [b-dom(Z, x) ∧ y ∈ Z]]]
Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[\forall x \left[ \text{anaphor}(x) \rightarrow \exists y \left[ \text{c-com}(y, x) \land \text{DP}(y) \land \exists Z \left[ \text{b-dom}(Z, x) \land y \in Z \right] \right] \right] \]

“For every $x$
Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[
\forall x \left[ \text{anaphor}(x) \rightarrow \exists y \left[ \text{c-com}(y, x) \land \text{DP}(y) \land \exists Z \left[ \text{b-dom}(Z, x) \land y \in Z \right] \right] \right]
\]

“For every \( x \) that is an anaphor
Principle A Again

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \forall x \left[ \text{anaphor}(x) \rightarrow \exists y \left[ \text{c-com}(y, x) \land \text{DP}(y) \land \exists Z \left[ \text{b-dom}(Z, x) \land y \in Z \right] \right] \right] \]

“For every \( x \) that is an anaphor it holds that”
Principle A Again

Principle A (slightly simplified)
Every anaphor must be c-commanded by some DP within its binding domain.

$$\forall x \left[ \text{anaphor}(x) \rightarrow \exists y \left[ \text{c-com}(y, x) \land \text{DP}(y) \land \exists Z \left[ \text{b-dom}(Z, x) \land y \in Z \right] \right] \right]$$

“For every \( x \) that is an anaphor it holds that there is a \( y \)”
Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \forall x \left[ \text{anaphor}(x) \rightarrow \exists y \left[ \text{c-com}(y, x) \land \text{DP}(y) \land \exists Z \left[ \text{b-dom}(Z, x) \land y \in Z \right] \right] \right] \]

“For every \( x \) that is an anaphor it holds that there is a \( y \) that c-commands \( x \)"
Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

∀x[anaphor(x) → ∃y[c-com(y, x) ∧ DP(y) ∧ ∃Z[b-dom(Z, x) ∧ y ∈ Z]]]

“For every x that is an anaphor it holds that there is a y that c-commands x and
Principle A Again

Principle A (slightly simplified)
Every anaphor must be c-commanded by some DP within its binding domain.

\[ \forall x \left[ \text{anaphor}(x) \rightarrow \exists y \left[ \text{c-com}(y, x) \land \text{DP}(y) \land \right. \right. \]
\[ \exists Z \left[ \text{b-dom}(Z, x) \land y \in Z \left. \right] \right] \]

“For every \( x \) that is an anaphor it holds that there is a \( y \) that c-commands \( x \) and is labeled DP,
Principle A Again

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \forall x \left[ \text{anaphor}(x) \rightarrow \exists y \left[ \text{c-com}(y, x) \land \text{DP}(y) \land \exists Z \left[ \text{b-dom}(Z, x) \land y \in Z \right] \right] \right] \]

“For every \( x \) that is an anaphor it holds that there is a \( y \) that c-commands \( x \) and is labeled DP, and...
Principle A Again

Principle A (slightly simplified)

Every anaphor must be c-commanded by some DP within its binding domain.

\[ \forall x \left[ \text{anaphor}(x) \rightarrow \exists y \left[ \text{c-com}(y, x) \land \text{DP}(y) \land \exists Z \left[ \text{b-dom}(Z, x) \land y \in Z \right] \right] \right] \]

“For every \( x \) that is an anaphor it holds that there is a \( y \) that c-commands \( x \) and is labeled DP, and there is a \( Z \)”
Principle A Again

Principle A (slightly simplified)
Every anaphor must be c-commanded by some DP within its binding domain.

\[ \forall x \left[ \text{anaphor}(x) \rightarrow \exists y \left[ \text{c-com}(y, x) \land \text{DP}(y) \land \exists Z \left[ \text{b-dom}(Z, x) \land y \in Z \right] \right] \right] \]

“For every \( x \) that is an anaphor it holds that there is a \( y \) that c-commands \( x \) and is labeled DP, and there is a \( Z \) that is the binding domain of \( x \)
Principle A Again

Principle A (slightly simplified)
Every anaphor must be c-commanded by some DP within its binding domain.

∀x [anaphor(x) → ∃y [c-com(y, x) ∧ DP(y) ∧ ∃Z [b-dom(Z, x) ∧ y ∈ Z]]]

“For every x that is an anaphor it holds that there is a y that c-commands x and is labeled DP, and there is a Z that is the binding domain of x and..."
Principle A Again

Principle A (slightly simplified)
Every anaphor must be c-commanded by some DP within its binding domain.

\[\forall x \left[ \text{anaphor}(x) \rightarrow \exists y \left[ \text{c-com}(y, x) \land \text{DP}(y) \land \exists Z \left[ \text{b-dom}(Z, x) \land y \in Z \right] \right] \right]\]

“For every \(x\) that is an anaphor it holds that there is a \(y\) that c-commands \(x\) and is labeled DP, and there is a \(Z\) that is the binding domain of \(x\) and contains \(y\).”
Further Remarks on the Power of MSO

MSO provides good fit for syntactic constraints, but not perfect:
- can state many unnatural constraints (too strong)
- cannot state some natural constraints (too weak)

<table>
<thead>
<tr>
<th>Definable Unnatural Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetric analog: “Every anaphor c-commands a DP”,</td>
</tr>
<tr>
<td>Unrelated properties: “The subject is plural if T is [+past]”,</td>
</tr>
<tr>
<td>Analogs from phonology: “The total number of nodes is even”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Undefinable Natural Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Subtrees A and B are identical”</td>
</tr>
</tbody>
</table>
  ⇒ problematic for ellipsis as deletion under (syntactic) identity |
| “The meaning of subtree A implies the meaning of B” |
  ⇒ problematic for semantic constraints in syntax |
Further Remarks on the Power of MSO

MSO provides good fit for syntactic constraints, but not perfect:

- can state many unnatural constraints (too strong)
- cannot state some natural constraints (too weak)

**Definable Unnatural Constraints**

- Symmetric analog: “Every anaphor c-commands a DP”,
- Unrelated properties: “The subject is plural if T is [+past]”,
- Analogs from phonology: “The total number of nodes is even”

**Undefinable Natural Constraints**

- “Subtrees A and B are identical”
  $\Rightarrow$ problematic for ellipsis as deletion under (syntactic) identity
- “The meaning of subtree A implies the meaning of B”
  $\Rightarrow$ problematic for semantic constraints in syntax
Further Remarks on the Power of MSO

MSO provides good fit for syntactic constraints, but not perfect:
- can state many unnatural constraints (too strong)
- cannot state some natural constraints (too weak)

Definable Unnatural Constraints
- Symmetric analog: “Every anaphor c-commands a DP”,
- Unrelated properties: “The subject is plural if T is [+past]”,
- Analogs from phonology: “The total number of nodes is even”

Undefinable Natural Constraints
- “Subtrees A and B are identical”
  ⇒ problematic for ellipsis as deletion under (syntactic) identity
- “The meaning of subtree A implies the meaning of B”
  ⇒ problematic for semantic constraints in syntax
A constraint is MSO-definable iff it can be computed by a \textit{(finite-state) tree automaton}.

A tree automaton

- assigns each node in a tree one of finitely many \textit{states}, and
- accepts the tree iff its root is assigned a \textit{final state}.
MSO & Tree Automata (Thatcher and Wright 1968; Doner 1970)

A constraint is MSO-definable iff it can be computed by a (finite-state) tree automaton.

A tree automaton
- assigns each node in a tree one of finitely many states, and
- accepts the tree iff its root is assigned a final state.
A constraint is MSO-definable iff it can be computed by a \textit{(finite-state) tree automaton}.

A tree automaton

- assigns each node in a tree one of finitely many \textit{states}, and
- accepts the tree iff its root is assigned a \textit{final state}.
A constraint is MSO-definable iff it can be computed by a (finite-state) tree automaton.

A tree automaton
- assigns each node in a tree one of finitely many states, and
- accepts the tree iff its root is assigned a final state.
Section Summary

Minimalist Syntax

- Formalized in terms of MGs
- Operations: Merge and Move (Agree omitted for convenience)
- Triggered by checking features of opposite polarities

Constraints

- Constraints $\equiv$ MSO formulas $\equiv$ tree automata
- MSO can talk about both nodes and sets of nodes
  $\Rightarrow$ expressive enough for syntax
- Tree automata compute constraints in a local way using finitely bounded number of states ($\approx$ working memory)
  $\Rightarrow$ cognitive plausibility
1. The Status of Constraints in Linguistics

2. Formal Concepts
   - Minimalist Grammars
   - Constraints as Logical Formulas

3. Results
   - Main Result: Constraints $\equiv$ Merge
   - Corollary: Uniformity of Constraint Classes
   - Linguistic Implications

4. Are MSO-Constraints Enough? A Look at Binding
   - Syntactic Binding: No Semantics, No Discourse
   - Computing Principle B
   - English
   - American Sign Language (ASL)

5. Conclusion & Outlook
The Central Result

MSO-Constraints $\equiv$ Merge (Graf 2011; Kobele 2011)

A constraint $C$ can be expressed by an MG iff $C$ is MSO-definable.

Proof idea

- convert constraint $C$ into tree automaton $A$
- incorporate states of $A$ into feature calculus
  $\Rightarrow$ “refined” grammar expresses $C$ via Merge
The Central Result

**MSO-Constraints ≡ Merge (Graf 2011; Kobele 2011)**

A constraint $C$ can be expressed by an MG iff $C$ is MSO-definable.

**Proof idea**
- convert constraint $C$ into tree automaton $A$
- incorporate states of $A$ into feature calculus
  $\Rightarrow$ “refined” grammar expresses $C$ via Merge

![Diagram of Merge and feature calculus expressions]
The Central Result

MSO-Constraints $\equiv$ Merge (Graf 2011; Kobele 2011)

A constraint $C$ can be expressed by an MG iff $C$ is MSO-definable.

Proof idea

- convert constraint $C$ into tree automaton $A$
- incorporate states of $A$ into feature calculus
  $\Rightarrow$ “refined” grammar expresses $C$ via Merge
Uniformity of Constraint Classes

Reminder: Müller-Sternefeld Hierarchy of Constraints

representational $<$ derivational $<$ transderivational

Formal Result: Uniformity of MSO-Constraints

For every MG

representational $\equiv$ derivational $\equiv$ transderivational ($\equiv$ local)

(Graf 2010, 2011, 2012a,b; Kobele 2011)
Uniformity of Constraint Classes

Reminder: Müller-Sternefeld Hierarchy of Constraints

representational $<$ derivational $<$ transderivational

Formal Result: Uniformity of MSO-Constraints

For every MG

representational $\equiv$ derivational $\equiv$ transderivational ($\equiv$ local)

(Graf 2010, 2011, 2012a,b; Kobele 2011)
Recap: What Just Happened?

- Monadic Second-Order logic as description language:
  - powerful enough for stating syntactic constraints
  - computable with finite working-memory
- Every MSO-constraint expressible purely through Merge
  
  **Metaphor**: Put memory states into category features

- Corollary: constraint types conflate into one

**The New Perspective on Constraints**

- Existence of MSO-definable constraints unsurprising given power of Merge
- But are there really only MSO-constraints in syntax? What would the implications be?
Recap: What Just Happened?

- Monadic Second-Order logic as description language:
  - powerful enough for stating syntactic constraints
  - computable with finite working-memory

- Every MSO-constraint expressible purely through Merge
  **Metaphor**: Put memory states into category features

- Corollary: constraint types conflate into one

**The New Perspective on Constraints**

- Existence of MSO-definable constraints unsurprising given power of Merge

- But are there really only MSO-constraints in syntax? What would the implications be?
Implications for Processing

- **derivational ≡ local**
  solves problem of parsing long-distance dependencies in a local, incremental manner (cf. Alcocer and Phillips 2012)

- **derivational ≡ representational**
  allows parsing derivations rather than phrase structure trees
  ⇒ increased performance (Stabler 2012)
  correctly predicts processing difficulties (Kobele et al. 2012)
Implications for Processing

- **derivational ≡ local**
  solves problem of parsing long-distance dependencies in a local, incremental manner (cf. Alcocer and Phillips 2012)

- **derivational ≡ representational**
  allows parsing derivations rather than phrase structure trees
  ⇒ increased performance (Stabler 2012)
  correctly predicts processing difficulties (Kobele et al. 2012)
Outline

1 The Status of Constraints in Linguistics

2 Formal Concepts
   - Minimalist Grammars
   - Constraints as Logical Formulas

3 Results
   - Main Result: Constraints $\equiv$ Merge
   - Corollary: Uniformity of Constraint Classes
   - Linguistic Implications

4 Are MSO-Constraints Enough? A Look at Binding
   - Syntactic Binding: No Semantics, No Discourse
   - Computing Principle B
   - English
   - American Sign Language (ASL)

5 Conclusion & Outlook
Syntactic Binding: No Semantics

- **Canonical Binding Theory:**
  is sentence grammatical with respect to *specific reading*?
- requires storing referent for each pronoun
- number of pronouns per sentence unbounded
  ⇒ no upper bound on number of referents
  ⇒ needs unbounded amount of working memory

Syntactic Binding (preliminary)
Does a given sentence have *some grammatical reading*?
Syntactic Binding: No Semantics

- **Canonical Binding Theory:**
  - is sentence grammatical with respect to *specific reading*?
  - requires storing referent for each pronoun
  - number of pronouns per sentence unbounded
    - no upper bound on number of referents
    - needs unbounded amount of working memory

**Syntactic Binding (preliminary)**

Does a given sentence have *some grammatical reading*?
Syntactic Binding: No Discourse

- Discourse-mechanisms arguably not part of syntax
  ⇒ syntax only regulates pronouns that must be syntactically bound (cf. aapan in Marathi; Kiparsky 2002)
- **Technical assumption**
  English has discourse-bound $him_d$ and syntactically bound $him_s$; only the latter is of interest here

## Syntactic Binding (final)
Is there some *syntactically* grammatical reading?

### Example

(1) a. Every patient said that $he_d$ should sedate $him_s$.
b. *Every patient said that $he_s$ should sedate $him_s$.
c. Every patient told some doctor that $he_s$ should sedate $him_s$. 
Syntactic Binding: No Discourse

- Discourse-mechanisms arguably not part of syntax
  ⇒ syntax only regulates pronouns that must be syntactically bound (cf. *aapaṇ* in Marathi; Kiparsky 2002)
- **Technical assumption**
  English has discourse-bound *him*\(_d\) and syntactically bound *him*\(_s\); only the latter is of interest here

Syntactic Binding (final)

Is there some **syntactically** grammatical reading?

Example

(1) a. Every patient said that he\(_d\) should sedate him\(_s\).
   b. * Every patient said that he\(_s\) should sedate him\(_s\).
   c. Every patient told some doctor that he\(_s\) should sedate him\(_s\).
Principle A is Easy

- already saw how Principle A can be expressed in MSO ⇒ SE/SELF anaphors no problem
- What about long distance reflexives?
  - Icelandic type: usually allows local binding ⇒ like SE/SELF
    
    (2) Jóni segir að Maríaj elski sigi/j.  
    Jon says that Maria loves.SUBJ SE  
    ‘Jon says that Maria loves him/herself.’
  
- Swedish type: no local binding allowed ⇒ like pronouns

(3) Generalen; tvingade överstenj PROj att hjälpa  
    General.the forced colonel.the PRO to help  
    sigi/*j.  
    SE  
    ‘The general forced the colonel to help him(*self).’
Principle B: Limited Obviation

While Principle A is easy, Principle B is difficult because of its obviation requirement (= no local binding).

Syntactic Binding and MSO

Syntactic Binding is MSO-definable iff Limited Obviation holds.

Limited Obviation

For every binding domain, its syntactically bound pronouns need at most a total of $n$ antecedents to yield a grammatical reading.

So, what does that mean?

If a binding domain contains more than $n$ bound pronouns, those additional pronouns can be coreferent with pronouns in the same domain $\Rightarrow$ Principle B exceptions.
Principle B: Limited Obviation

While Principle A is easy, Principle B is difficult because of its **obviation requirement** (= no local binding).

**Syntactic Binding and MSO**

Syntactic Binding is MSO-definable iff **Limited Obviation** holds.

**Limited Obviation**

For every binding domain, its syntactically bound pronouns need at most a total of $n$ antecedents to yield a grammatical reading.

So, what does that mean?

If a binding domain contains more than $n$ bound pronouns, those additional pronouns can be coreferent with pronouns in the same domain $\Rightarrow$ Principle B exceptions.
Principle B: Limited Obviation

While Principle A is easy, Principle B is difficult because of its **obviation requirement** (= no local binding).

**Syntactic Binding and MSO**

Syntactic Binding is MSO-definable iff **Limited Obviation** holds.

**Limited Obviation**

For every binding domain, its syntactically bound pronouns need at most a total of $n$ antecedents to yield a grammatical reading.

**So, what does that mean?**

If a binding domain contains more than $n$ bound pronouns, those additional pronouns can be coreferent with pronouns in the same domain $\Rightarrow$ Principle B exceptions.
How would one Falsify Limited Obviation?

- All binding proposals agree that there is some domain within which pronouns may not be syntactically bound ≈ binding/obviation domain
- binding domain ≤ CP
- Within a single CP, there are three ways of introducing an unbounded number of pronominal DPs:
  - adjuncts
  - nested TPs/νPs, VPs, and DPs
  - coordination

**Limited Obviation** is violated only if the pronouns in these configurations all obviate each other (i.e. are mandatorily disjoint in reference).
Pronouns contained by adjuncts usually lack obviation.

(4) Every/No/Some woman put the box down in front of her.

But even when obviation can be observed, pronouns contained by distinct adjuncts do not obviate each other.

(5) a. * Every/No/Some priest sacrificed a goat for him.
    b. Every/No/Some Egyptian goddess asked of some priest that he sacrifice a goat for her in honor of her.

Hence adjuncts increase the required number of antecedents only by a limited amount.
Nested TPs/VPs

- **Nested TPs/\(vP\)s**
  In English, TPs establish **new obviation domains** (although overlap is possible for Spec,TP).

  (6) a. * Every/No/Some patient said that he wants him to sedate him.
       b. Every/No/Some patient told some doctor that he wants him (to convince him) to sedate him.

- **Nested VPs**
  Nested VPs, if they exist at all in English, behave like nested TPs.

  (7) a. * Every/No/Some patient said that he made him operate on him.
       b. Every/No/Some doctor told some patient that he made him (watch him) operate on him.
Nested DPs with Possessors

Depending on your choice of binding theory, one of the two holds:

- possessed DPs establish a **new obviation domain**
- pronouns inside possessed DPs are **not obviative**

Either way **Limited Obviation** is satisfied.

(8) a. Every/No/Some politician liked the photographer’s picture of him.

b. Every/No/Some politician complained about [the reporter’s article on him and [the photographer’s picture of him]].
There is **no obviatiom effect** with non-possessed DPs.

(9)  

a. Every/No/Some post-modern artist must paint at least one [picture of [him and a picture of him]].  
b. Every/No/Some client wanted to see a [presentation of [a presentation to him] to him].
Coordination involving bound pronouns is **ungrammatical if the two pronouns are identical.**

(10)  

a. Every/No/Some football player told every/no/some cheerleader that the coach wants to see him and her in the office.

b. * Every/No/Some football player told every/no/some masseur that the coach wants to see him and him in the office.

Since every language has only a finite number of distinct pronouns, coordination can only introduce a bounded number of pronouns that obviate each other.
Coordination of bound pronouns is grammatical in ASL.

(11) ALL\textsubscript{i} WRESTLER\textsubscript{i} INFORM\textsubscript{j} SOMEONE\textsubscript{j} SWIMMER\textsubscript{j} THAT IX\textsubscript{i}/j IX\textsubscript{j}/i WILL RIDE-IN-VEHICLE LIMO GO-TO DANCE
Every wrestler\textsubscript{i} told some swimmer\textsubscript{j} that him\textsubscript{i}/j and him\textsubscript{j}/i would ride in a limo to the dance.

(12) EACH\textsubscript{i} WRESTLER\textsubscript{i} TELL\textsubscript{j} SOMEONE\textsubscript{j} SWIMMER\textsubscript{j} THAT SOMEONE\textsubscript{k} FOOTBALL\textsubscript{k} PLAYER\textsubscript{k} ASK CAN IX\textsubscript{i} IX\textsubscript{j} IX\textsubscript{k} THREE-HUMANS-GO-TO DANCE (TOGETHER)
Each wrestler\textsubscript{i} told some swimmer\textsubscript{j} that some football player\textsubscript{k} asked if him\textsubscript{i} and him\textsubscript{j} and him\textsubscript{k} could go to the dance together.

**Binding in ASL**

- Every DP can be assigned a *locus* in space.
- Pronominal binding is realized by *pointing at the locus* which a DP has been assigned to (transcribed as IX).
Coordination of bound pronouns is grammatical in ASL.

(11) \[ \text{ALL}_i \text{ WRESTLER}_i \text{ INFORM}_j \text{ SOMEONE}_j \text{ SWIMMER}_j \text{ THAT} \]
\[ \text{IX}_{i/j} \text{ IX}_{j/i} \text{ WILL RIDE-IN-VEHICLE LIMO GO-TO DANCE} \]
Every wrestler \( i \) told some swimmer \( j \) that him \( i/j \) and him \( j/i \) would ride in a limo to the dance.

(12) \[ \text{EACH}_i \text{ WRESTLER}_i \text{ TELL}_j \text{ SOMEONE}_j \text{ SWIMMER}_j \text{ THAT} \]
\[ \text{SOMEONE}_k \text{ FOOTBALL}_k \text{ PLAYER}_k \text{ ASK CAN IX}_i \text{ IX}_j \text{ IX}_k \text{ THREE-HUMANS-GO-TO DANCE (TOGETHER)} \]
Each wrestler \( i \) told some swimmer \( j \) that some football player \( k \) asked if him \( i \) and him \( j \) and him \( k \) could go to the dance together.

**Binding in ASL**

- Every DP can be assigned a **locus** in space.
- Pronominal binding is realized by **pointing at the locus** which a DP has been assigned to (transcribed as IX).
The Role of Deixis

Pointing at referents in space resembles deictic pronouns in English. And deictic pronouns can easily be coordinated.

(13) Every/No/Some football player told every/some/no masseur that the coach wants to see him\textit{deictic} and him\textit{deictic} in his office.

Since \textbf{Limited Obviation} only applies to syntactic binding, (13) does not constitute a counterexample.

The Big Question

Are the coordinated pronouns in ASL \textit{syntactically} bound?
The Role of Deixis

Pointing at referents in space resembles deictic pronouns in English. And deictic pronouns can easily be coordinated.

(13) Every/No/Some football player told every/some/no masseur that the coach wants to see him$_{deictic}$ and him$_{deictic}$ in his office.

Since **Limited Obviation** only applies to syntactic binding, (13) does not constitute a counterexample.

**The Big Question**

Are the coordinated pronouns in ASL **syntactically** bound?
Non-empty Domain Restrictions

While pronouns can be discourse-bound by quantifiers in English, the extension of the quantified DP must be non-empty.

(14) a. Every player is handed a card. He then has to role a dice.
    b. # No player is handed a card. He then has to role a dice.

A similar pattern emerges for pronouns in ASL.

(15) EACH POLITICS PERSON$_i$ TELL- STORY (IX$_i$) WANT WIN
    Each politician$_i$ said he$_i$ wants to win.

(16) NO POLITICS PERSON$_i$ TELL- STORY (?*IX$_i$) WANT WIN
    No politician$_i$ said he$_i$ wants to win.
Are MSO-definable constraints sufficient?
Probably yes, if semantics isn’t involved:

- Constraint results only hold for syntax
  ⇒ Syntactic fragment of binding theory
- No discourse binding, no evaluation of specific readings
- Even then a limit on the number of required antecedents per binding domain is mandatory for MSO-computability ⇒ **Limited Obviation**
- Satisfied in English and (probably) ASL
- A new descriptive universal for binding theory?
Questions
- Why do constraints exist at all?
- What kinds of constraints are there?
- How powerful are they?
- How do they fit into syntax?
- What are the empirical implications?

Formal Answer
- Constraints are a natural by-product of Merge.
- The MG-expressible constraints are exactly those that can be computed with a finitely bounded amount of working memory.
- Within this class, all subtypes are interchangeable.

Implications
- offers new solutions to processing problems
- prompts new descriptive universal for pronominal binding
Outlook

Constraints through Merge

Uniformity of Constraints

Binding
Outlook

Constraints through Merge

Uniformity of Constraints

Binding

More Data


References IV


Compiling States into Features

\[
\text{the} \quad \text{men} \quad \text{like} \quad \text{which} \quad \text{men}
\]

\[
\begin{array}{c}
\text{Merge[D ]}
\
\text{Merge[N ]}
\
\text{Merge[D ]}
\
\text{Merge[N ]}
\end{array}
\]

\[
\begin{array}{c}
\text{N}^+ \quad \text{D}^- \\
\text{N}^- \\
\text{D}^+ \quad \text{D}^+ \quad \text{V}^- \\
\text{N}^+ \quad \text{D}^- \\
\text{N}^-
\end{array}
\]
Compiling States into Features

```
the       men
N+ D-     N-

like
D+ D+ V-

which
N+ D-

men
N-
```
Compiling States into Features

Merge[N 4]  
  Merge[D 1]  
    Merge[N 5]  
      1 the  
      N+ D-  
    Merge[D 7]  
      Merge[N 8]  
        6 which  
        N+ D-  
  Merge[D 4]  
    Merge[N 9]  
      9 like  
      D+ D+ V-  
    Merge[D 7]  
      Merge[N 8]  
        8 men  
        N-  

Compiling States into Features

References

Constraints

Binding

Compiling States into Features

Merge[D]

Merge[N]

Merge[D]

Merge[N]

1 the

N^+ D^-

4 men

N^- D^- men

7 like

D^+ D^+ V^-

8 which

N^+ D^- men

1

4

9

6

8

5
Compiling States into Features

1 the N⁺ D⁻ men N⁻₅
2 like D⁺ D⁺ V⁻
6 which N⁺ D⁻ men N⁻
Compiling States into Features

- Merge[$N_5^+$]
  - Merge[$N_5^-$]
    - the
    - men
    - like
  - Merge[$D^+$]
    - Merge[$D^-$]
      - Merge[$N_5^-$]
        - Merge[$N_5^+$]
          - Merge[$D^+$]
            - Merge[$D^-$]
              - Merge[$V^-$]
    - Merge[$D^-$]
      - Merge[$N^+$]
        - Merge[$D^-$]
          - Merge[$N^-$]
    - Merge[$D^-$]
      - Merge[$N^+$]
        - Merge[$N^-$]

References
Constraints
Binding
Compiling States into Features

1. the
   \[ N^+_5 \] \[ D^- \]

4. men
   \[ N^-_5 \]

7. Merge[D]

8. Merge[N]
   \[ N^+_5 \] \[ D^- \] \[ N^-_5 \]

9. like
   \[ D^+ \] \[ D^+ \] \[ V^- \]
Compiling States into Features

```
1 the
N_5^+ D^-

4 men
N_5^-

9 like
D^+ D^+ V^-

7
Merge[D]

8
Merge[N_5]

6 which
N^+ D^-

men
N_5^-
```
Compiling States into Features

- Merge[$D^+$] - Merge[$D^+$] - Merge[$V^-$]
- Merge[$D^+$] - Merge[$D^+$] - Merge[$V^-$]
- Merge[$D^+$] - Merge[$D^+$] - Merge[$V^-$]
- Merge[$D^+$] - Merge[$D^+$] - Merge[$V^-$]
Compiling States into Features

Merge[$N_5^+$] the men

Merge[$N_5^-$] D$^-$

Merge[$D_8^-$] which men

Merge[$N_5^-$] D$^+$ D$^+$ V$^-$

Merge[$D_8^+$]
Compiling States into Features

References

Constraints

Binding
Compiling States into Features

Merge[N₅]

Merge[D]

Merge[D₈]

Merge[N₅]

1 the  
N₅⁺ D⁻  
N₅⁻

4 men  
D₈⁺ D⁺ V⁻

9 like  
N₅⁺ D₈⁻

7  
N₅⁻

6 which  
N₅⁺

men  
N₅⁻
Compiling States into Features

Merge[$N_5^+$] 1 the $D_4^-$

Merge[$N_5^-]$ men

Merge[$D_8^+$] 9 like $D^+ V^-$

Merge[$D_8^-]$ 7

Merge[$N_5^+$] 6 which $D_8^-$

Merge[$N_5^-]$ men

$N_5^+$ $D_4^-$ $N_5^-$ $D_8^+$ $D^+$ $V^-$

$N_5^- D_8^-$ $N_5^-$
Compiling States into Features

Merge[$D_4$]

Merge[$N_5$]

1. the

$N_5^+ \quad D_4^-$

2. men

$N_5^-$

9. like

$D_8^+ \quad D^+ \quad V^-$

Merge[$D_8$]

6. which

$N_5^+ \quad D_8^-$

7. men

$N_5^-$
Compiling States into Features

```
Merge[D_4]

1
/
Merge[N_5]

1 the

N_5^+ D_4^- N_5^- 

9 like

D_8^+ D_4^+ V^-

Merge[D_8]

7
/
Merge[N_5]

6 which

N_5^+ D_8^- 

men

N_5^- 
```
Compiling States into Features

\[
\text{Merge}[D_4] \quad \text{Merge}[D_8] \\
\text{Merge}[N_5] \quad \text{Merge}[N_5]
\]

1. the
\[N_5^+ \quad D_4^- \quad N_5^-\]
2. men
3. like
\[D_8^+ \quad D_4^+ \quad V_1^-\]
4. which
\[N_5^+ \quad D_8^-\]
5. men
\[N_5^-\]
Compiling States into Features

Merge[$D_4$]

<table>
<thead>
<tr>
<th>Merge[$N_5$]</th>
<th>the</th>
<th>men</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_5^+$</td>
<td>$D_4^-$</td>
<td>$N_5^-$</td>
</tr>
</tbody>
</table>

Merge[$D_8$]

<table>
<thead>
<tr>
<th>Merge[$N_5$]</th>
<th>like</th>
<th>which</th>
<th>men</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_5^+$</td>
<td>$D_8^-$</td>
<td>$D_4^+$</td>
<td>$V_1^-$</td>
</tr>
</tbody>
</table>

Merge[$N_5$]
Representational $\equiv$ Derivational

<table>
<thead>
<tr>
<th>MSO over Representations</th>
<th>MSO over Derivations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\land, \lor, \neg, \rightarrow$</td>
<td>$\land, \lor, \neg, \rightarrow$</td>
</tr>
<tr>
<td>$\exists, \forall$</td>
<td>$\exists, \forall$</td>
</tr>
<tr>
<td>$\equiv$</td>
<td>$\equiv$</td>
</tr>
</tbody>
</table>

- $x \triangleleft y$ iff $\phi(x, y)$, where $\phi$ uses $\blacktriangleleft$ but not $\triangleleft$
  $\Rightarrow$ replacing each occurrence of $x \triangleleft y$ by $\phi(x, y)$
in representational constraint $C$ yields derivational $C'$
- $x \blacktriangleleft y$ iff $\psi(x, y)$, where $\psi$ uses $\triangleleft$ but not $\blacktriangleleft$
  $\Rightarrow$ replacing each occurrence of $x \blacktriangleleft y$ by $\psi(x, y)$
in derivational constraint $C$ yields representational $C'$
Derivational ≡ Transderivational

A Different Perspective on Transderivationality

Transderivational constraints do not filter out suboptimal trees. They **rewrite suboptimal trees into optimal ones**.

- Rewrite procedure carried out by **linear tree transducer**
- Given a set of Minimalist derivations as inputs, transducer produces set of outputs that can be computed by a tree automaton
- Compile said automaton into the features as usual
(Dis)Advantages of Constraint Classes

Are constraints redundant? Should we just use feature checking?

- **Shortcomings of Local Constraints**
  less succinct, often incomprehensible, hide generalizations

- **Shortcomings of Derivational Constraints**
  some constraints are significantly more complicated when stated over derivations (e.g. ECP)

- **Advantages of Transderivational Constraints**
  can state generalizations *across grammars* that are not expressible with derivational/representational constraints

**Methodological Moral of the Story**

Even though the constraint classes have the same power, they each have their own advantages and disadvantages.
⇒ Use the type of constraint that is best suited to the task!
A Purely Transderivational Generalization

**Shortest Derivation Principle**

Given a set of competing derivations, pick the one with the fewest instances of Move.

### Toy Grammar 1

- At least one DP moves out of VP.
- Two options:
  - Move to SpecYP, and YP then moves to SpecZP (roll-up)
  - Move directly to SpecZP (one-fell-swoop)
- Result: Exactly one DP must move from VP to SpecZP.

### Toy Grammar 2

- At most one DP moves out of VP, directly into SpecZP.
- Result: No DP may move from VP to SpecZP.
## A Purely Transderivational Generalization

### Shortest Derivation Principle

Given a set of competing derivations, pick the one with the fewest instances of Move.

### Toy Grammar 1

- At least one DP moves out of VP.
- Two options:
  - Move to SpecYP, and YP then moves to SpecZP (roll-up)
  - Move directly to SpecZP (one-fell-swoop)
- Result: Exactly one DP must move from VP to SpecZP.

### Toy Grammar 2

- At most one DP moves out of VP, directly into SpecZP.
- Result: No DP may move from VP to SpecZP.
A Purely Transderivational Generalization

Shortest Derivation Principle

Given a set of competing derivations, pick the one with the fewest instances of Move.

Toy Grammar 1

- At least one DP moves out of VP.
- Two options:
  - Move to SpecYP, and YP then moves to SpecZP (roll-up)
  - Move directly to SpecZP (one-fell-swoop)
- Result: Exactly one DP must move from VP to SpecZP.

Toy Grammar 2

- At most one DP moves out of VP, directly into SpecZP.
- Result: No DP may move from VP to SpecZP.
Implications for Acquisition

- “cognitive load explanations” for acquisition delays of Principle B and Focus Projection (Grodzinsky and Reinhart 1993; Szendrői 2004)
  - Certain processes involve transderivational constraints.
  - Comparing multiple trees too computationally demanding for young children
    ⇒ delay in acquisition due to insufficient working memory
- Implausible explanation if transderivational ≡ derivational (no extra processing load)
- Recent findings: Principle B delay an artifact of experimental setup
  (Papafragou and Musolino 2003; Papafragou and Tantalou 2004; Elbourne 2005; Conroy et al. 2009)
- Same problem with Focus experiment?
  A pilot study is in preparation.
Two Common Questions on ASL Binding

What about other obviation domains in ASL?
Coordination and nested VP/TP domains provide the only true ASL parallel to their English counterparts, the latter of which also introduce new binding domains in ASL. Nested DP structures are not well-attested in the language and comparable adjunct structures are expressed in ASL through the use of complex locative and classifier morphology.

Could spatial reference just be an elaborate case or gender system?
Then the grammatical coordination examples parallel the coordination of *him* and *her* in English and are not a problem for Limited Obviation. However, there is no sense in which spatial loci are inherently associated with (pro)nominals in ASL, as is typical of gender systems, nor are spatial loci reliably assigned in specific syntactic environments, as is typical of case systems.
Some speakers accept (17) as grammatical.

(17) ?? Every/No/Some football player told every/no/some masseur that the coach wants him to run six laps and him to prepare the massage room.

If this pattern is a productive instance of coordinating syntactically bound pronouns, it would falsify **Limited Obviation**. But just like in ASL, the binding mechanism at play here arguably isn’t (purely) syntactic in nature.

- Most speakers need to put (contrastive) stress on the respective pronouns.
- *No* seems to be dispreferred compared to *every* and *some*.
- There is no c-command requirement (even in configurations where QR is bounded).
(18)  a. A coach of every/some football player told a receptionist of every/some masseur that the team’s president wants him to get a massage and him to give it.

b. An agent of every/some actress told a bodyguard of every/some first lady that he wants her to do a movie about Jackie Kennedy and her to be on the set as a consultant.

c. An interview that every/some football player liked included a quib, which every/some masseur had related to the reporter at some point, that the coach always ordered him to run six laps and him to prepare the massage chair.