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.

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.

Outline

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.	

constraints ⟷ operations

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)

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.	

constraints ⟷ operations

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
  $\Rightarrow$ 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

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]

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^-, ...)
- Derivation tree differs only with respect to labels

Merge: Example 1

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

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^-, ...)
- Derivation tree differs only with respect to labels

Merge: Example 1

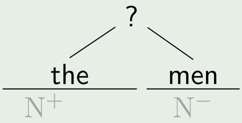
Assembling $[_{\text{DP}}$ the men$]$

$$\frac{\text{the}}{N^{+}} \quad \frac{\text{men}}{N^{-}}$$

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

Merge: Example 1

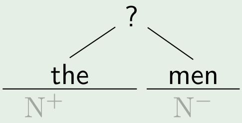
Assembling [$_{\text{DP}}$ the men]



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

Merge: Example 1

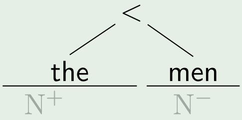
Assembling [$_{\text{DP}}$ the men]



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

Merge: Example 1

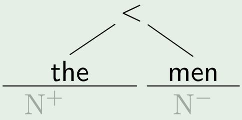
Assembling [$_{\text{DP}}$ the men]



- 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

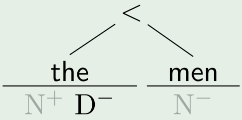
Assembling [$_{\text{DP}}$ the men]



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

Merge: Example 1

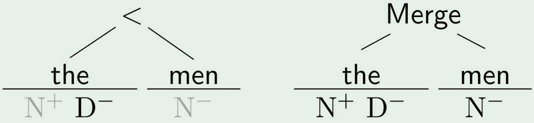
Assembling [$_{\text{DP}}$ the men]



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

Merge: Example 1

Assembling [$_{DP}$ 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^-, ...)
- 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^{-}, ...)
- Derivation tree differs only with respect to labels

Merge: Example 2

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

the	men	like	which	men
$\mathrm{N}^{+}\ \mathrm{D}^{-}$	N^{-}			

- *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]

- *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

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

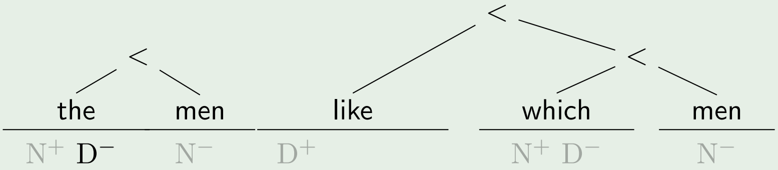
Merge: Example 2

Assembling [$_{\mathrm{VP}}$ 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

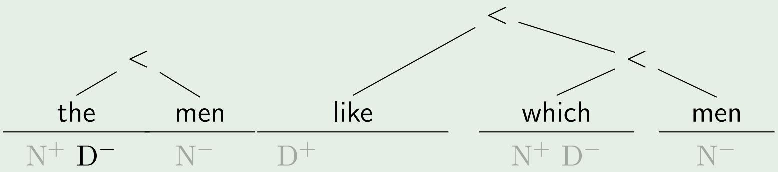
Assembling [$_{\text{VP}}$ the men like which men]



- *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

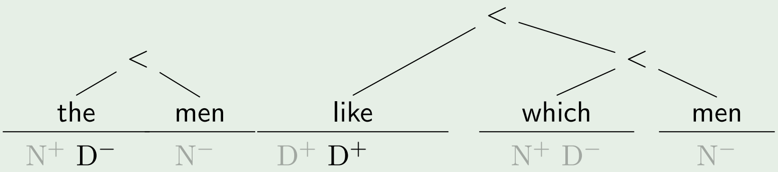
Assembling [$_{\text{VP}}$ the men like which men]



- *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

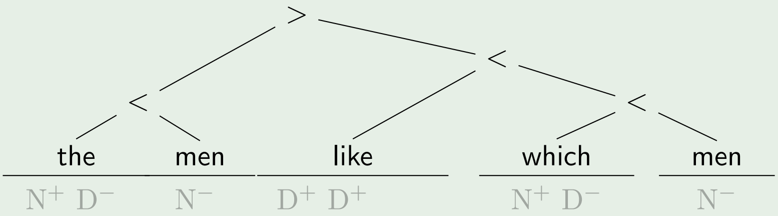
Assembling [$_{\text{VP}}$ the men like which men]



- *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

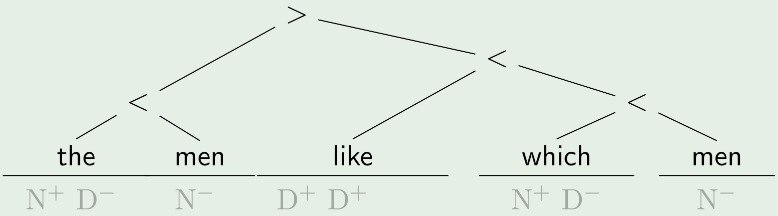
Assembling [$_{VP}$ the men like which men]



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

Merge: Example 2

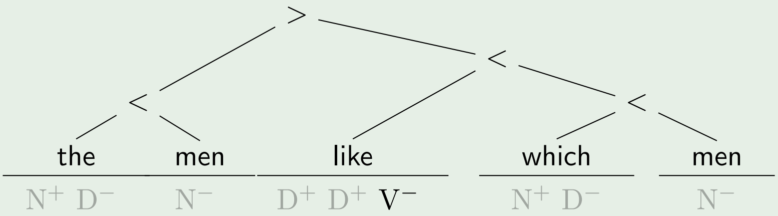
Assembling [$_{\text{VP}}$ the men like which men]



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

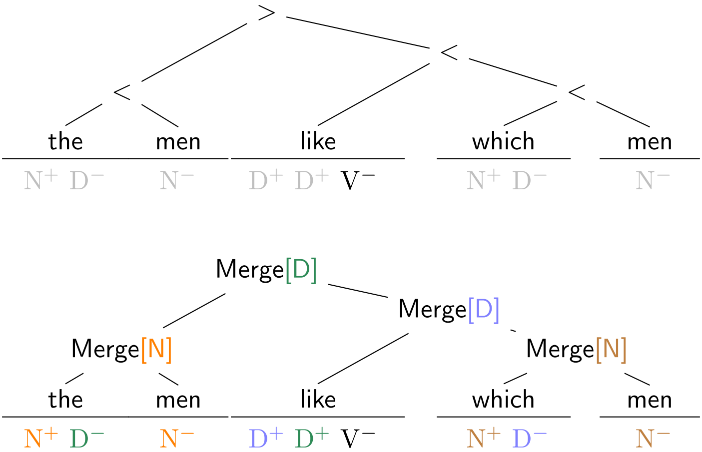
Merge: Example 2

Assembling [$_{VP}$ the men like which men]



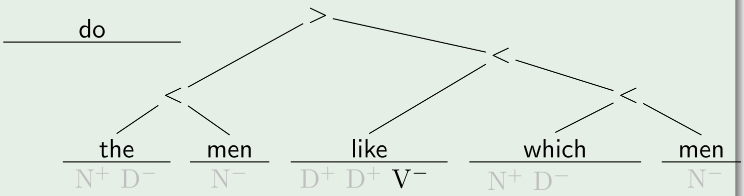
- *the* and *men* merged as before
- same steps for *which men*
- *like* selects *which men*
- *like* selects *the men*
- *like* needs a category feature

Merge: Example 2 [cont.]



Move (Multi-Dominance Implementation)

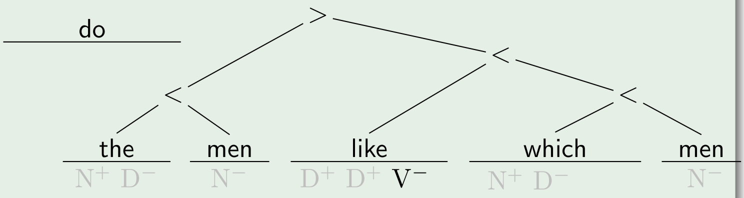
Assembling "which men do the men like?"



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

Move (Multi-Dominance Implementation)

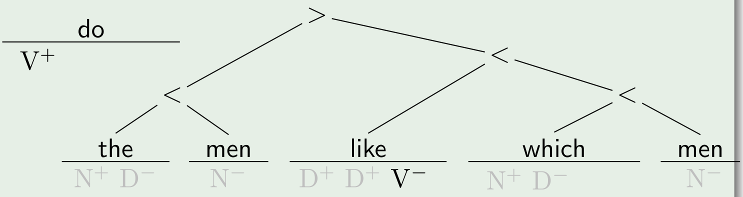
Assembling "which men do the men like?"



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

Move (Multi-Dominance Implementation)

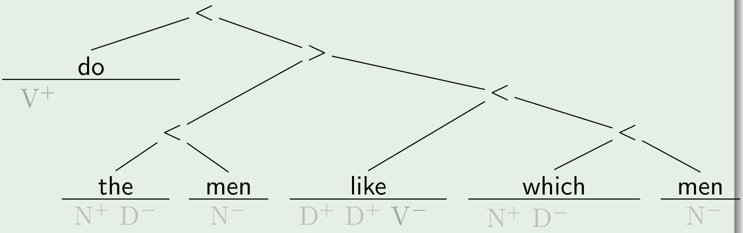
Assembling "which men do the men like?"



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

Move (Multi-Dominance Implementation)

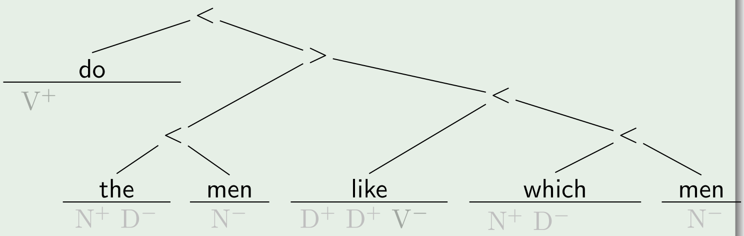
Assembling "which men do the men like?"



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

Move (Multi-Dominance Implementation)

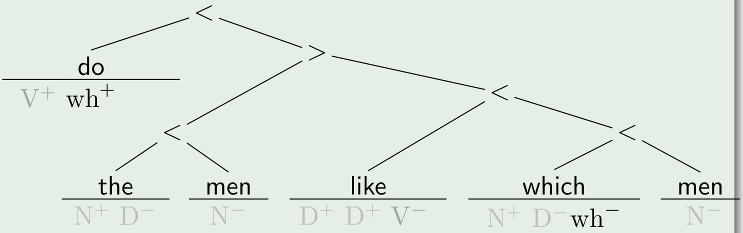
Assembling "which men do the men like?"



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

Move (Multi-Dominance Implementation)

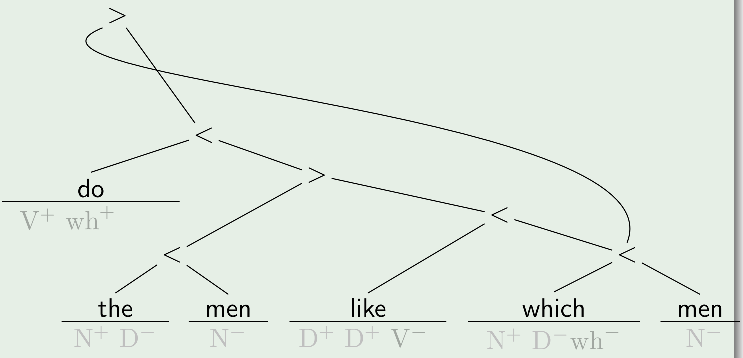
Assembling "which men do the men like?"



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

Move (Multi-Dominance Implementation)

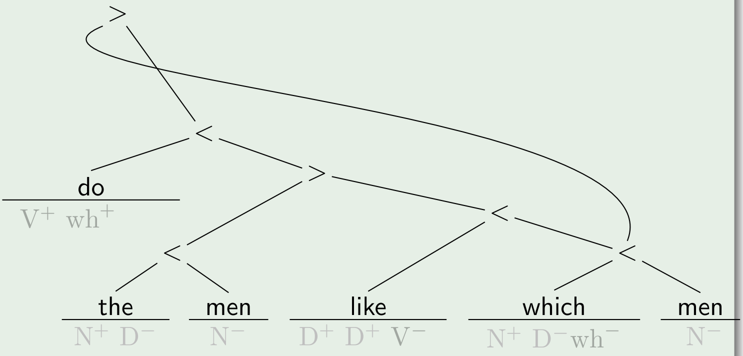
Assembling "which men do the men like?"



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

Move (Multi-Dominance Implementation)

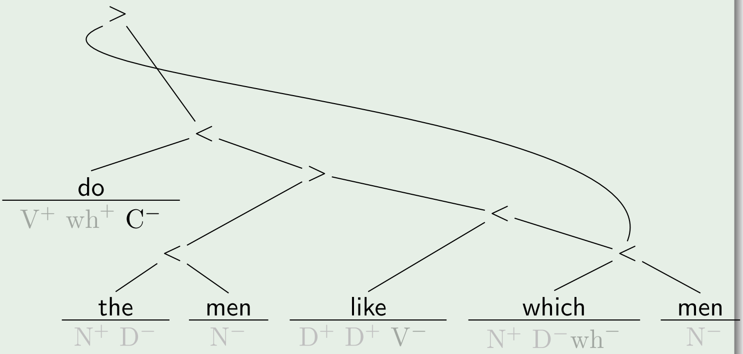
Assembling “which men do the men like?”



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

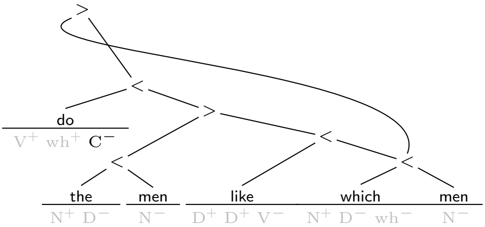
Move (Multi-Dominance Implementation)

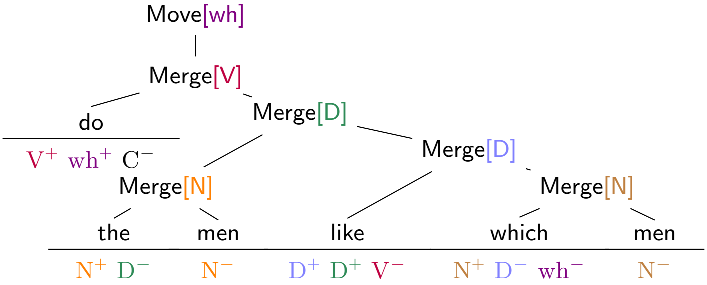
Assembling "which men do the men like?"



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

Derivation Trees with Move





MG Summary

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

$$\frac{\text{men}}{N^-} \qquad \frac{\text{the}}{D^+\ N^-} \qquad \frac{\text{which}}{D^+\ N^-\ \mathrm{wh}^-}$$

$$\frac{\text{like}}{D^+\ D^+\ V^-} \qquad \frac{\text{do}}{V^+\ \mathrm{wh}^+\ C^-} \qquad \frac{\varepsilon}{V^+\ C^-}$$

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 ϕ that must be satisfied in order for a structure to be a model of ϕ

$\Rightarrow$ **Constraints $\equiv$ Logical Formulas**

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

First-Order Logic for Trees (FO)

$x, y, z, \ldots$	variables $x, y, z, \ldots$
$\wedge, \vee, \neg, \rightarrow, \leftrightarrow$	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 ϕ that must be satisfied in order for a structure to be a model of ϕ

$\Rightarrow$ **Constraints $\equiv$ Logical Formulas**

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

First-Order Logic for Trees (FO)

$x, y, z, \dots$	variables $x, y, z, \dots$
$\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$	and, or, not, implies, iff
$\exists$, $\forall$	there is, for all
$l(x)$	x has label l
$\triangleleft$	dominance
$\approx$	equivalence

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

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

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

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

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor

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

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff

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

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 *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.

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself*

or

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

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself*
or x is labeled *herself*

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

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself*
or x is labeled *herself* or

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

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself* or x is labeled *herself* or x is labeled *itself*."

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

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself* or x is labeled *herself* or x is labeled *itself*."

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

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself* or x is labeled *herself* or x is labeled *itself*."

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

"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.

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself* or x is labeled *herself* or x is labeled *itself*."

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

"x c-commands y iff

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself*
or x is labeled *herself* or x is labeled *itself*."

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

"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.

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself*
or x is labeled *herself* or x is labeled *itself*."

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

"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.

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself*
or x is labeled *herself* or x is labeled *itself*."

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

"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.

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself* or x is labeled *herself* or x is labeled *itself*."

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

"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

Example: Stating Principle A

Principle A (slightly simplified)

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

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself*
or x is labeled *herself* or x is labeled *itself*."

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

"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.

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself* or x is labeled *herself* or x is labeled *itself*."

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

"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.

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself* or x is labeled *herself* or x is labeled *itself*."

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

"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.

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself* or x is labeled *herself* or x is labeled *itself*."

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

"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.

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself* or x is labeled *herself* or x is labeled *itself*."

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

"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.

$$\text{anaphor}(x) \leftrightarrow \text{himself}(x) \vee \text{herself}(x) \vee \text{itself}(x)$$

"x is an anaphor iff x is labeled *himself* or x is labeled *herself* or x is labeled *itself*."

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

"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 dominates y."

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg \exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

"X is the binding domain of y

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

"X is the binding domain of y iff

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

"X is the binding domain of y iff X is a TP

Set Quantification: Talking About Domains

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

Monadic Second-Order Logic for Trees (MSO)

Extension of FO with

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

"X is the binding domain of y iff X is a TP and

Set Quantification: Talking About Domains

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

Monadic Second-Order Logic for Trees (MSO)

Extension of FO with

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

"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** $\Rightarrow$ move beyond FO

Monadic Second-Order Logic for Trees (MSO)

Extension of FO with

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

"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** $\Rightarrow$ move beyond FO

Monadic Second-Order Logic for Trees (MSO)

Extension of FO with

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

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

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

"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** $\Rightarrow$ move beyond FO

Monadic Second-Order Logic for Trees (MSO)

Extension of FO with

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

"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** $\Rightarrow$ move beyond FO

Monadic Second-Order Logic for Trees (MSO)

Extension of FO with

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

"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** $\Rightarrow$ move beyond FO

Monadic Second-Order Logic for Trees (MSO)

Extension of FO with

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg\exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

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

$$\text{b-dom}(X, y) \leftrightarrow \mathrm{TP}(X) \wedge y \in X \wedge \neg \exists Z[y \in Z \wedge \mathrm{TP}(Z) \wedge Z \subset X]$$

"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."

Principle A Again

Principle A (slightly simplified)

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

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

Principle A Again

Principle A (slightly simplified)

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

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

Principle A Again

Principle A (slightly simplified)

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

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

"For every x

Principle A Again

Principle A (slightly simplified)

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

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

"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.

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

"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.

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

"For every x that is an anaphor it holds that
there is a y

Principle A Again

Principle A (slightly simplified)

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

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

"For every x that is an anaphor it holds that
there is a y that c-commands x

Principle A Again

Principle A (slightly simplified)

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

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

"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.

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

"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.

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

"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.

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

"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.

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

"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.

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

"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.

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

"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)

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"
  $\Rightarrow$ 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"
  $\Rightarrow$ 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

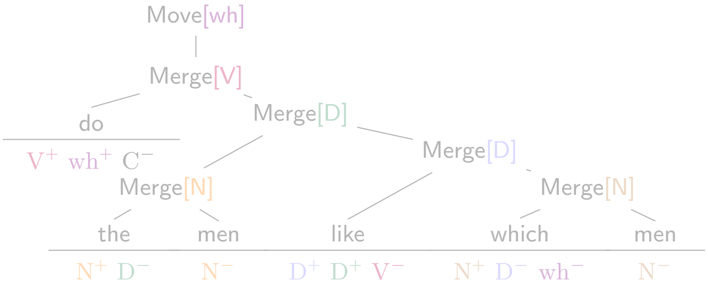
Computing MSO-Constraints

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**.



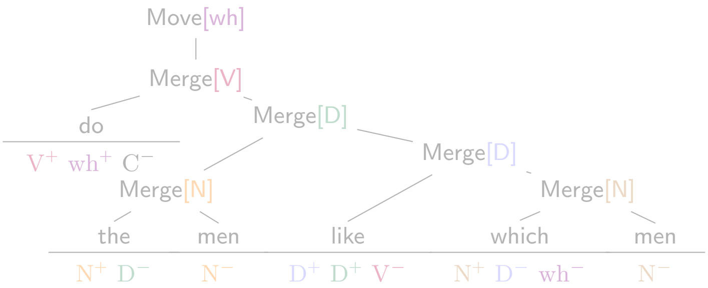
Computing MSO-Constraints

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**.



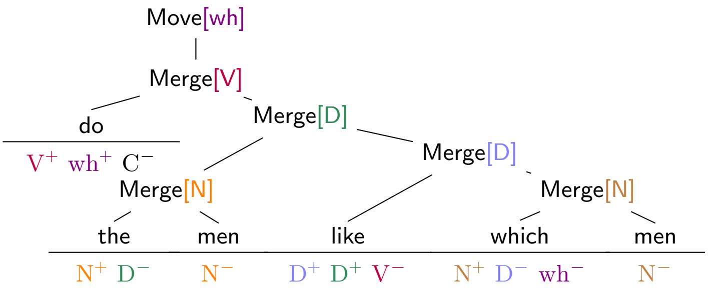
Computing MSO-Constraints

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**.



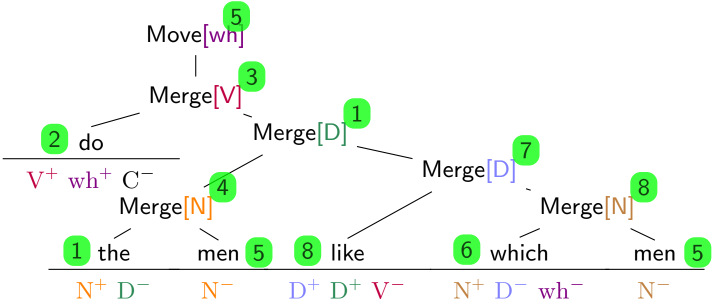
Computing MSO-Constraints

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**.



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**

Outline

The Central Result

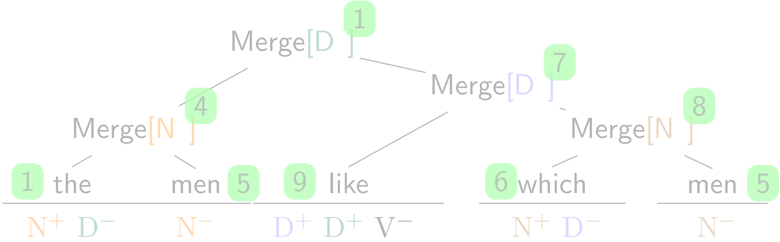
MSO-Constraints ≡ Merge (Graf 2011; Kobele 2011)

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**



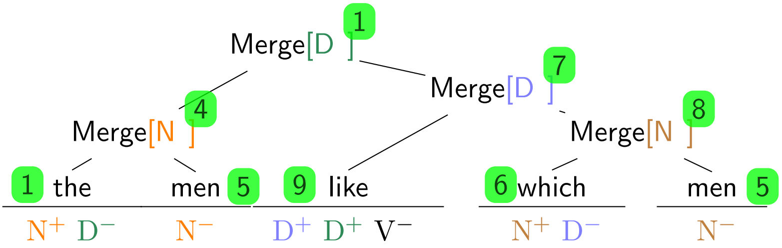
The Central Result

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**



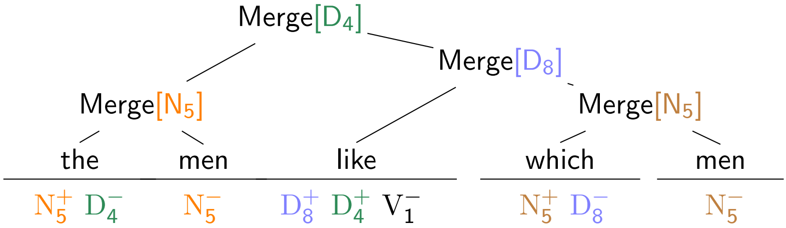
The Central Result

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**



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 $\equiv$ local**
  solves problem of parsing long-distance dependencies
  in a local, incremental manner (cf. Alcocer and Phillips 2012)
- **derivational $\equiv$ representational**
  allows parsing derivations rather than phrase structure trees
  $\Rightarrow$ 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

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
  $\Rightarrow$ no upper bound on number of referents
  $\Rightarrow$ 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
  $\Rightarrow$ no upper bound on number of referents
  $\Rightarrow$ 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
  $\Rightarrow$ 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.

Syntactic Binding: No Discourse

- Discourse-mechanisms arguably not part of syntax
  $\Rightarrow$ 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
  $\Rightarrow$ SE/SELF anaphors no problem
- What about long distance reflexives?
  - Icelandic type: usually allows local binding $\Rightarrow$ like SE/SELF

(2) Jón$_i$ segir að María$_j$ elski sig$_{i/j}$.
Jon says that Maria loves.SUBJ SE
'Jon says that Maria loves him/herself.'

  - Swedish type: no local binding allowed $\Rightarrow$ like pronouns

(3) Generalen$_i$ tvingade översten$_j$ PRO$_j$ att hjälpa sig$_{i/*j}$.
General.the forced colonel.the PRO to help 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 $\approx$ binding/obviation domain
- binding domain $\leq$ CP
- Within a single CP, there are three ways of introducing an unbounded number of pronominal DPs:
  - adjuncts
  - nested TPs/*v*Ps, 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).

Adjuncts

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/vPs*
  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]].

Nested DPs without Possessors

There is **no obviation 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

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.

A Counterexample in ASL? (Graf and Abner 2012)

Coordination of bound pronouns is grammatical in ASL.

(11) ALL$_{\mathrm{i}}$ WRESTLER$_{\mathrm{i}}$ INFORM$_{\mathrm{j}}$ SOMEONE$_{\mathrm{j}}$ SWIMMER$_{\mathrm{j}}$ THAT IX$_{\mathrm{i/j}}$ IX$_{\mathrm{j/i}}$ 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) EACH$_{\mathrm{i}}$ WRESTLER$_{\mathrm{i}}$ TELL$_{\mathrm{j}}$ SOMEONE$_{\mathrm{j}}$ SWIMMER$_{\mathrm{j}}$ THAT SOMEONE$_{\mathrm{k}}$ FOOTBALL$_{\mathrm{k}}$ PLAYER$_{\mathrm{k}}$ ASK CAN IX$_{\mathrm{i}}$ IX$_{\mathrm{j}}$ IX$_{\mathrm{k}}$ 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).

A Counterexample in ASL? (Graf and Abner 2012)

Coordination of bound pronouns is grammatical in ASL.

(11) $\mathtt{ALL}_\mathrm{i}$ $\mathtt{WRESTLER}_\mathrm{i}$ $\mathtt{INFORM}_\mathrm{j}$ $\mathtt{SOMEONE}_\mathrm{j}$ $\mathtt{SWIMMER}_\mathrm{j}$ THAT $\mathtt{IX}_{\mathrm{i/j}}$ $\mathtt{IX}_{\mathrm{j/i}}$ 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) $\mathtt{EACH}_\mathrm{i}$ $\mathtt{WRESTLER}_\mathrm{i}$ $\mathtt{TELL}_\mathrm{j}$ $\mathtt{SOMEONE}_\mathrm{j}$ $\mathtt{SWIMMER}_\mathrm{j}$ THAT $\mathtt{SOMEONE}_\mathrm{k}$ $\mathtt{FOOTBALL}_\mathrm{k}$ $\mathtt{PLAYER}_\mathrm{k}$ ASK CAN $\mathtt{IX}_\mathrm{i}$ $\mathtt{IX}_\mathrm{j}$ $\mathtt{IX}_\mathrm{k}$ 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 $\text{him}_{deictic}$ and $\text{him}_{deictic}$ 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?

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.

Section Summary

Are MSO-definable constraints sufficient?
Probably yes, if semantics isn't involved:

- Constraint results only hold for syntax
  $\Rightarrow$ 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 $\Rightarrow$ **Limited Obviation**
- Satisfied in English and (probably) ASL
- A new descriptive universal for binding theory?

Conclusion

- **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

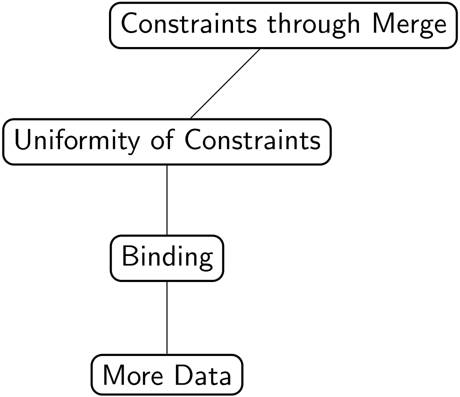
Outlook

Constraints through Merge

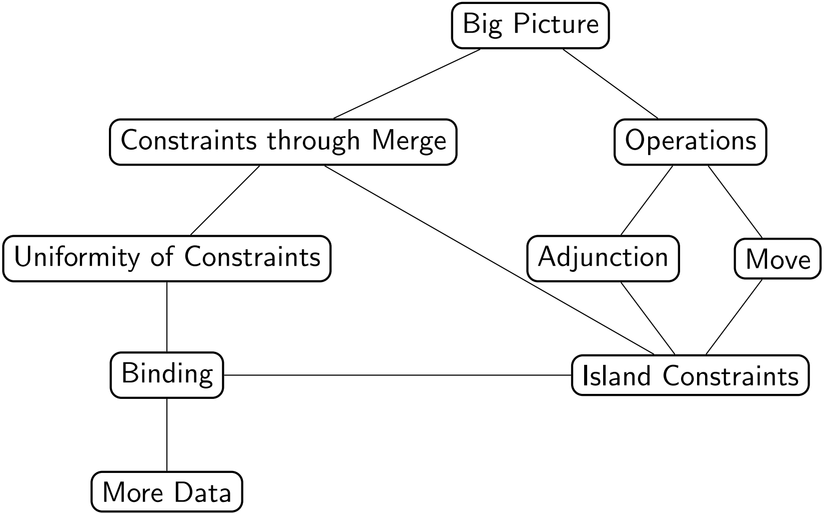
Uniformity of Constraints

Binding

Outlook


Constraints through Merge
Uniformity of Constraints
Binding
More Data

Outlook


Big Picture
Constraints through Merge
Operations
Uniformity of Constraints
Adjunction
Move
Binding
Island Constraints
More Data

References I

Alcocer, Pedro, and Colin Phillips. 2012. Using relational syntactic constraints in content-addressable memory architectures for sentence parsing. Ms., University of Maryland.

Bailyn, John F. 2010. Kinds of derivational binding. In *Formal studies in slavic linguistics*, ed. Gerhild Zybatow, volume 25 of *Linguistik International*, 11–30. Frankfurt am Main: Peter Lang.

Brody, Michael. 2002. On the status of representations and derivations. In *Derivation and explanation in the minimalist program*, ed. Samuel D. Epstein and Daniel T. Seely, 19–41. Oxford: Blackwell.

Conroy, Anastasia, Eri Takahashi, Jeffrey Lidz, and Colin Phillips. 2009. Equal treatment for all antecedents: How children succeed with Principle B. *Linguistc Inquiry* 40:446–486.

Doner, John. 1970. Tree acceptors and some of their applications. *Journal of Computer and System Sciences* 4:406–451.

Elbourne, Paul. 2005. On the acquisition of Principle B. *Linguistic Inquiry* 36:333–365.

Epstein, Samuel D., Erich M. Groat, Ruriko Kawashima, and Hisatsugu Kitahara. 1998. *A derivational approach to syntactic relations*. Oxford: Oxford University Press.

References II

Epstein, Samuel D., and Daniel T. Seely. 2002. Rule applications as cycles in a level-free syntax. In *Derivation and explanation in the minimalist program*, ed. Samuel D. Epstein and Daniel T. Seely, 65–89. Oxford: Blackwell.

Frey, Werner, and Hans-Martin Gärtner. 2002. On the treatment of scrambling and adjunction in minimalist grammars. In *Proceedings of the Conference on Formal Grammar (FGTrento)*, 41–52. Trento.

Graf, Thomas. 2010. A tree transducer model of reference-set computation. *UCLA Working Papers in Linguistics* 15:1–53.

Graf, Thomas. 2011. Closure properties of minimalist derivation tree languages. In *LACL 2011*, ed. Sylvain Pogodalla and Jean-Philippe Prost, volume 6736 of *Lecture Notes in Artificial Intelligence*, 96–111. Heidelberg: Springer.

Graf, Thomas. 2012a. Concealed reference-set computation: How syntax escapes the parser's clutches. In *Towards a biolinguistic understanding of grammar. Essays on interfaces*, ed. Anna Maria Di Sciullo, 339–362. Amsterdam: John Benjamins.

Graf, Thomas. 2012b. Movement-generalized minimalist grammars. In *LACL 2012*, ed. Denis Béchet and Alexander J. Dikovsky, volume 7351 of *Lecture Notes in Computer Science*, 58–73.

Graf, Thomas, and Natasha Abner. 2012. Is syntactic binding rational? In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, 189–197.

References III

Grodzinsky, Yosef, and Tanja Reinhart. 1993. The innateness of binding and coreference. *Linguistic Inquiry* 24:69–102.

Kiparsky, Paul. 2002. Disjoint reference and the typology of pronouns. In *More than words*, ed. Ingrid Kaufmann and Barbara Stiebels, volume 53 of *Studia Grammatica*, 179–226. Berlin: Akademie Verlag.

Kisseberth, Charles. 1970. On the functional unity of phonological rules. *Linguistic Inquiry* 1:291–306.

Kobele, Gregory M. 2002. Formalizing mirror theory. *Grammars* 5:177–221.

Kobele, Gregory M. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. In *LACL 2011*, ed. Sylvain Pogodalla and Jean-Philippe Prost, volume 6736 of *Lecture Notes in Artificial Intelligence*, 129–144.

Kobele, Gregory M. 2012. Idioms and extended transducers. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, 153–161. Paris, France. URL `http://www.aclweb.org/anthology-new/W/W12/W12-4618`.

Kobele, Gregory M., Sabrina Gerth, and John T. Hale. 2012. Memory resource allocation in top-down minimalist parsing. In *Proceedings of Formal Grammar 2012*.

References IV

Kracht, Marcus. 1995. Is there a genuine modal perspective on feature structures? *Linguistics and Philosophy* 18:401–458.

Müller, Gereon. 2005. Constraints in syntax. Lecture Notes, Universität Leipzig.

Müller, Gereon, and Wolfgang Sternefeld. 2000. The rise of competition in syntax: A synopsis. In *Competition in syntax*, ed. Wolfgang Sternefeld and Gereon Müller, 1–68. Berlin: Mouton de Gruyter.

Papafragou, Anna, and Julien Musolino. 2003. Scalar implicatures: Experiments at the semantics-pragmatics interface. *Cognition* 86:253–282.

Papafragou, Anna, and Niki Tantalou. 2004. Children's computation of implicatures. *Language Acquisition* 12:71–82.

Potts, Christopher. 2001. Three kinds of transderivational constraints. In *Syntax at Santa Cruz*, ed. Séamas Mac Bhloscaidh, volume 3, 21–40. Santa Cruz: Linguistics Department, UC Santa Cruz.

Pullum, Geoffrey K. 2007. The evolution of model-theoretic frameworks in linguistics. In *Model-Theoretic Syntax @ 10*, ed. James Rogers and Stephan Kepser, 1–10.

Pullum, Geoffrey K., and Barbara C. Scholz. 2001. On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In *Logical aspects of computational linguistics: 4th international conference*, ed. Philippe de Groote, Glyan Morril, and Christian Retoré, number 2099 in Lecture Notes in Artificial Intelligence, 17–43. Berlin: Springer.

References V

Pullum, Geoffrey K., and Barbara C. Scholz. 2005. Contrasting applications of logic in natural language syntactic description. In *Logic, Methodology and Philosophy of Science: Proceedings of the Twelfth International Congress*, ed. Petr Hájek, Luis Valdés-Villanueva, and Dag Westerståhl, 481–503. London: College Publications Department of Computer Science, KCL, The Strand, London.

Rogers, James. 1998. *A descriptive approach to language-theoretic complexity*. Stanford: CSLI.

Stabler, Edward P. 1997. Derivational minimalism. In *Logical aspects of computational linguistics*, ed. Christian Retoré, volume 1328 of *Lecture Notes in Computer Science*, 68–95. Berlin: Springer.

Stabler, Edward P. 2003. Comparing 3 perspectives on head movement. In *Syntax at sunset 3: Head movement and syntactic theory*, ed. A. Mahajan, volume 10 of *UCLA Working Papers in Linguistics*, 178–198. Los Angeles, CA: UCLA.

Stabler, Edward P. 2006. Sidewards without copying. In *Formal Grammar '06, Proceedings of the Conference*, ed. Gerald Penn, Giorgio Satta, and Shuly Wintner, 133–146. Stanford: CSLI Publications.

Stabler, Edward P. 2011. Computational perspectives on minimalism. In *Oxford handbook of linguistic minimalism*, ed. Cedric Boeckx, 617–643. Oxford: Oxford University Press.

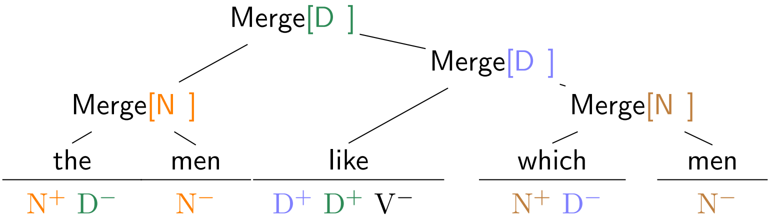
References VI

Stabler, Edward P. 2012. Bayesian, minimalist, incremental syntactic analysis. Ms., UCLA.

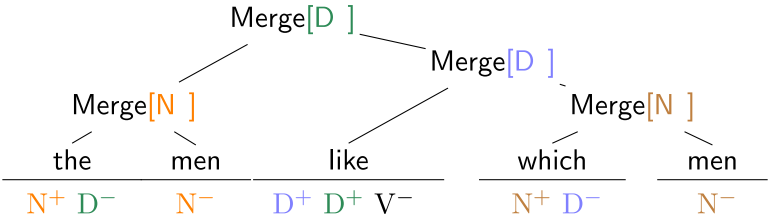
Szendrői, Kriszta. 2004. Acquisition evidence for an interface theory of focus. In *Proceedings of GALA 2003*, ed. Jaqueline van Kampen and Sergio Bauuw, 457–468. Utrecht: LOT.

Thatcher, James W., and J. B. Wright. 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory* 2:57–81.

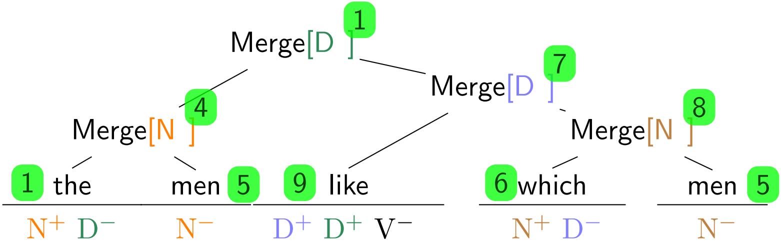
Compiling States into Features



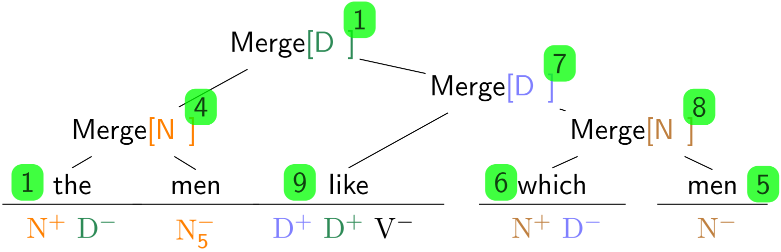
Compiling States into Features



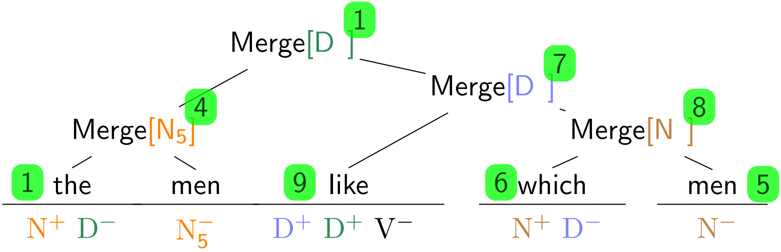
Compiling States into Features



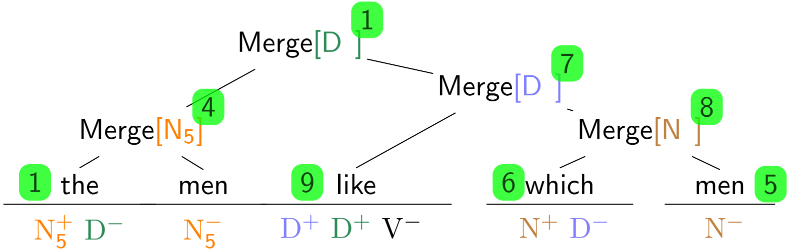
Compiling States into Features



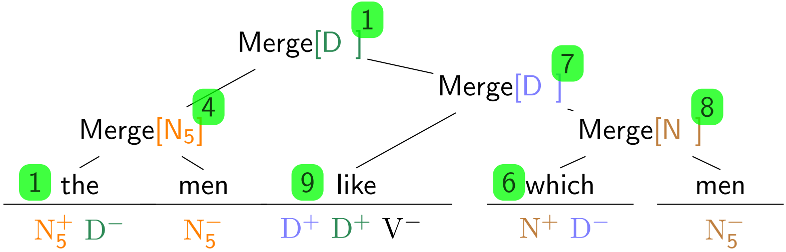
Compiling States into Features



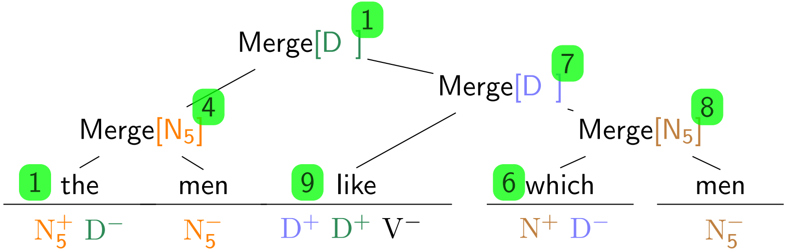
Compiling States into Features



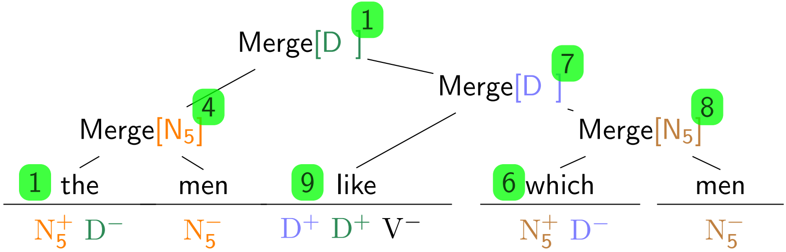
Compiling States into Features



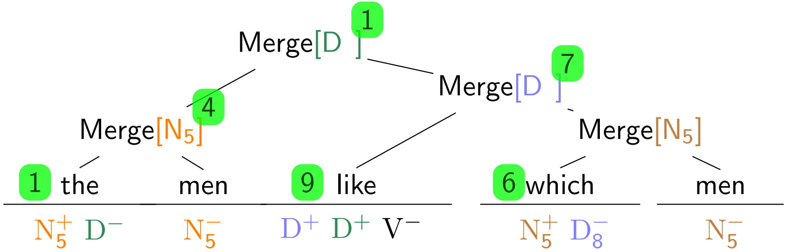
Compiling States into Features



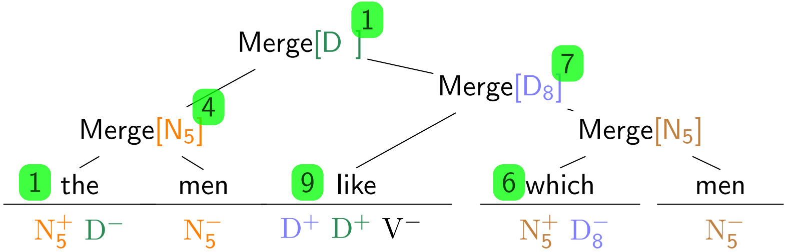
Compiling States into Features



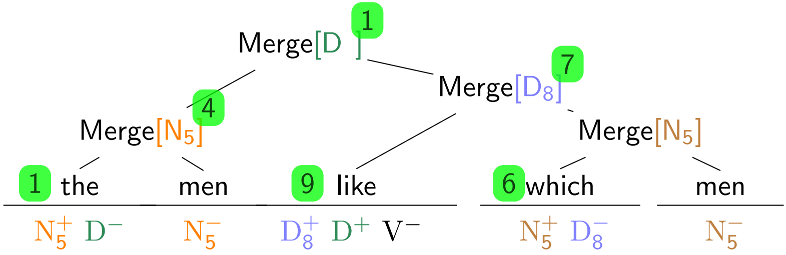
Compiling States into Features



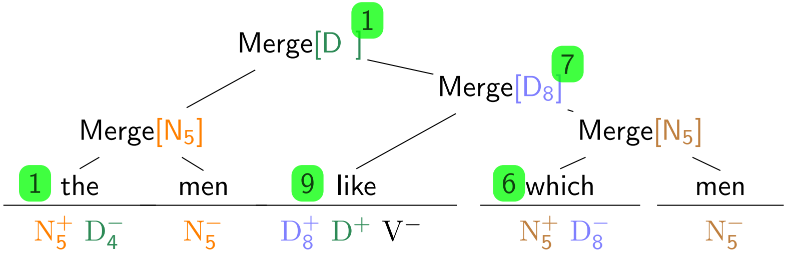
Compiling States into Features



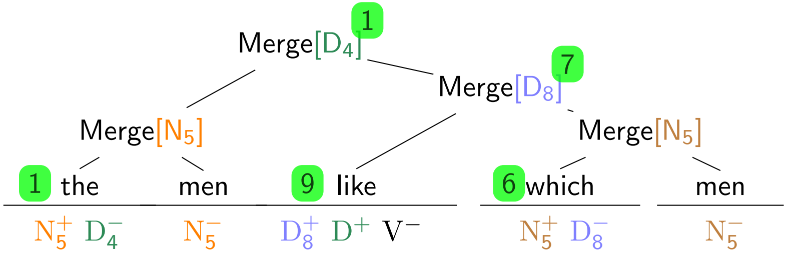
Compiling States into Features



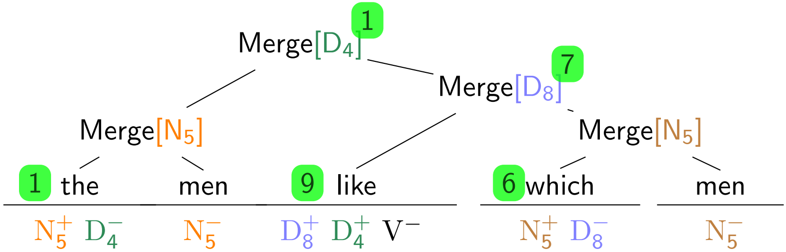
Compiling States into Features



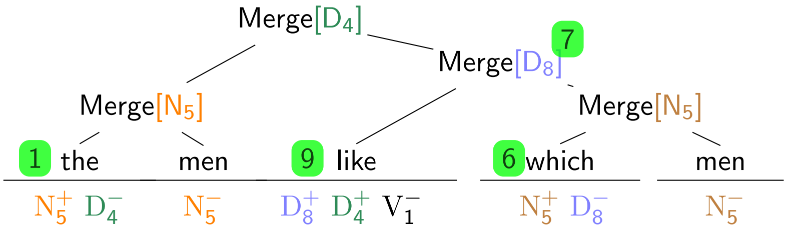
Compiling States into Features



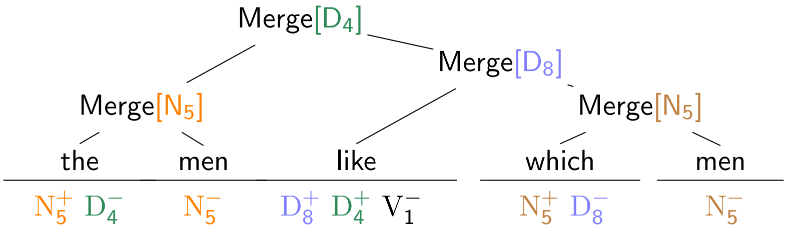
Compiling States into Features



Compiling States into Features



Compiling States into Features



Representational $\equiv$ Derivational

MSO over Representations	MSO over Derivations
$\wedge, \vee, \neg, \rightarrow$	$\wedge, \vee, \neg, \rightarrow$
$\exists, \forall$	$\exists, \forall$
$\equiv$	$\equiv$
standard dominance $\triangleleft$	derivational dominance $\blacktriangleleft$

- $x \triangleleft y$ iff $\phi(x, y)$, where ϕ 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 ψ 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 $\equiv$ 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
    $\Rightarrow$ delay in acquisition due to insufficient working memory
- Implausible explanation if transderivational $\equiv$ 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.

More on Coordination in English

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 disprefered compared to *every* and *some*.
- There is no c-command requirement (even in configurations where QR is bounded).

More on Coordination in English [cont.]

(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.