Mathematical structures in syntax

Meaghan Fowlie
mfowlie@ucla.edu
http://meaghanfowlie.com

UCLA/McGill

Concordia Linguistics
January 22, 2016

Motivation

Why approach syntax mathematically/computationally?

- Forces precision
- Forces you to make choices, at least temporarily (is this good or bad?)
- Removes some human error in testing consequences of theories
- Mathematical properties of grammars probably relate to mathematical properties of learning mechanisms
- If you figure out something in language is an example of a well-understood mathematical object, suddenly you know way more about the linguistic object as well

Overview

Apply computational thinking to three things:

1. Minimalist Interface Conditions
2. Minimalism
3. Adjuncts

Minimalism (Chomsky, 1995)

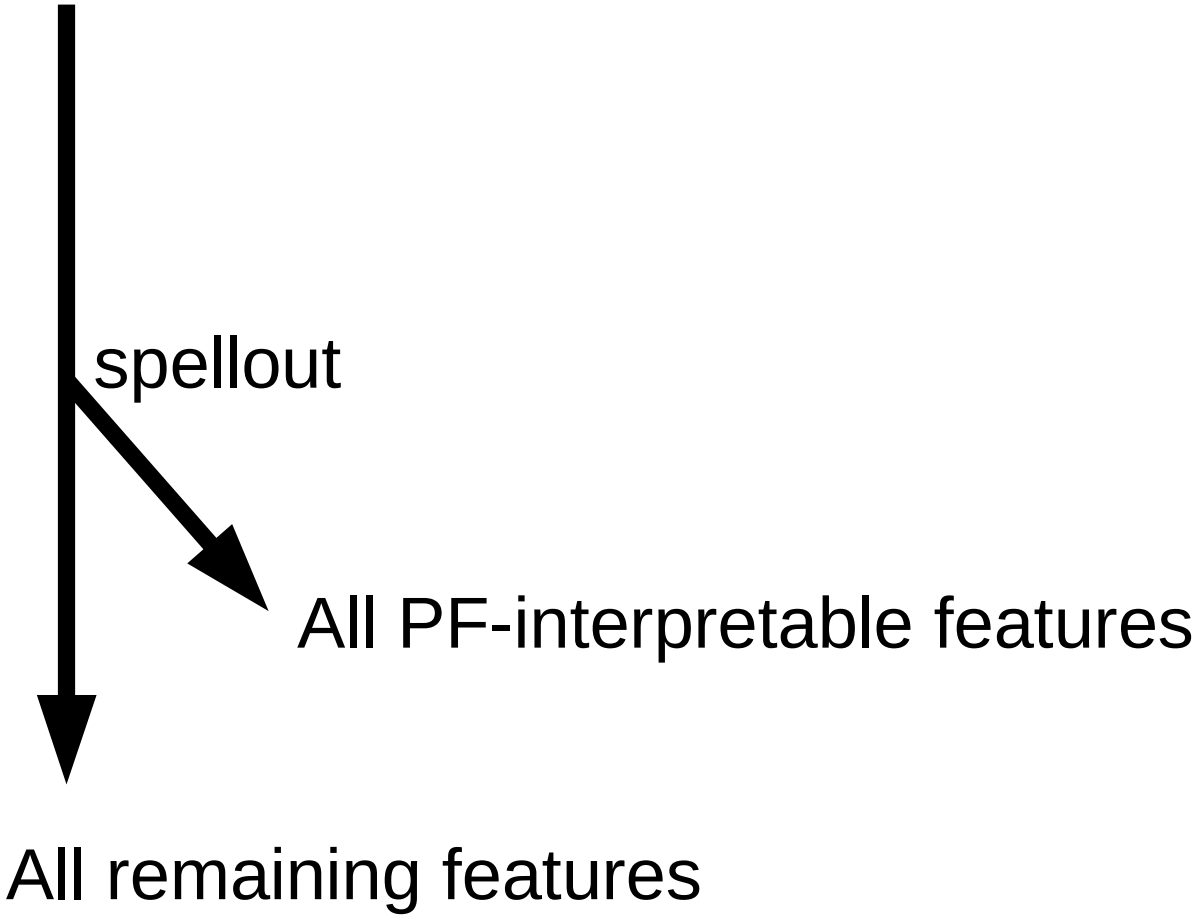
- Atoms of the derivation (words? morphemes?) have (are?) sets of features
- Basic operation: Merge := take two things and put them together
- Two Merges:
  - External Merge (or *Merge*): two things start separate
  - Internal Merge (or *Move*): one thing is a sub-part of the other
- Move/Internal Merge is driven by features. When you Move (or Agree?) you get to cancel pairs of matching features. (or just the uninterpretable one?)
- (External) Merge driven by features too? Or maybe it just has to create something interpretable by LF?

Interface conditions in Minimalism

- Words have/are sets of features
- Two interfaces: Articulatory-Perceptual (PF), Conceptual-Interpretive (LF)
- Full Interpretation: all features are interpreted by an interface
- Intuition: if an interface gets a feature it doesn't understand, the derivation crashes

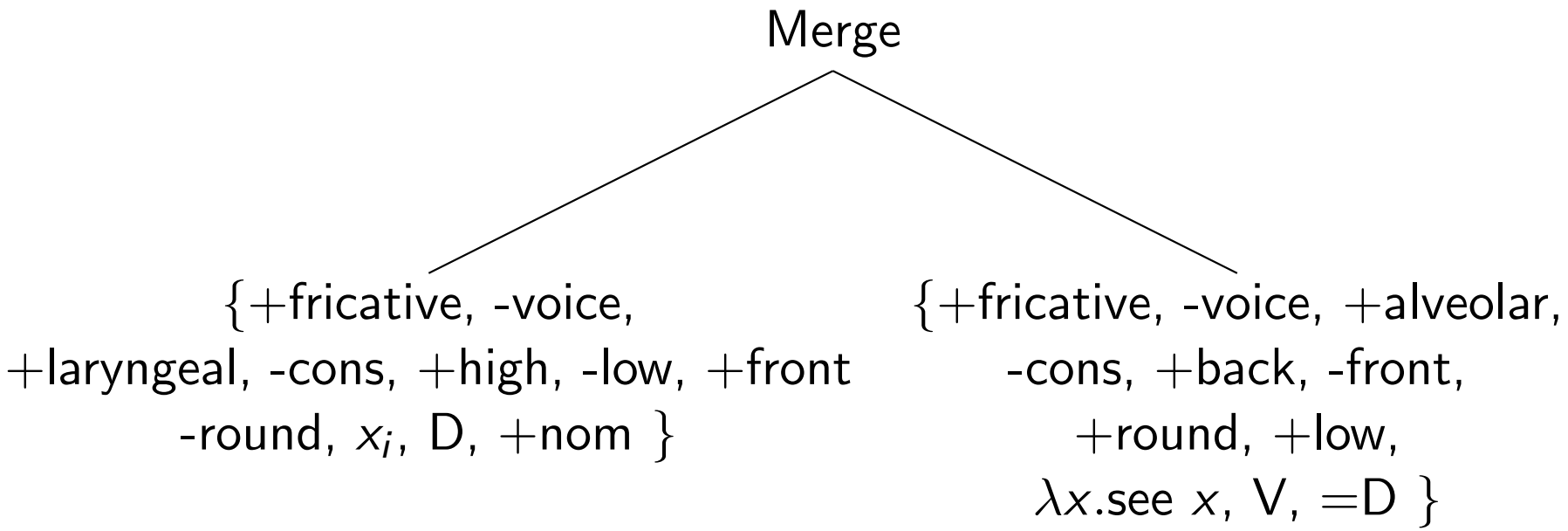
Query: *how can we best implement the intuition?* Imagine we want to write a program or a mathematical function that builds sentences and spells them out to the interfaces.

Interface – y-model



Derivation tree for *He saw*

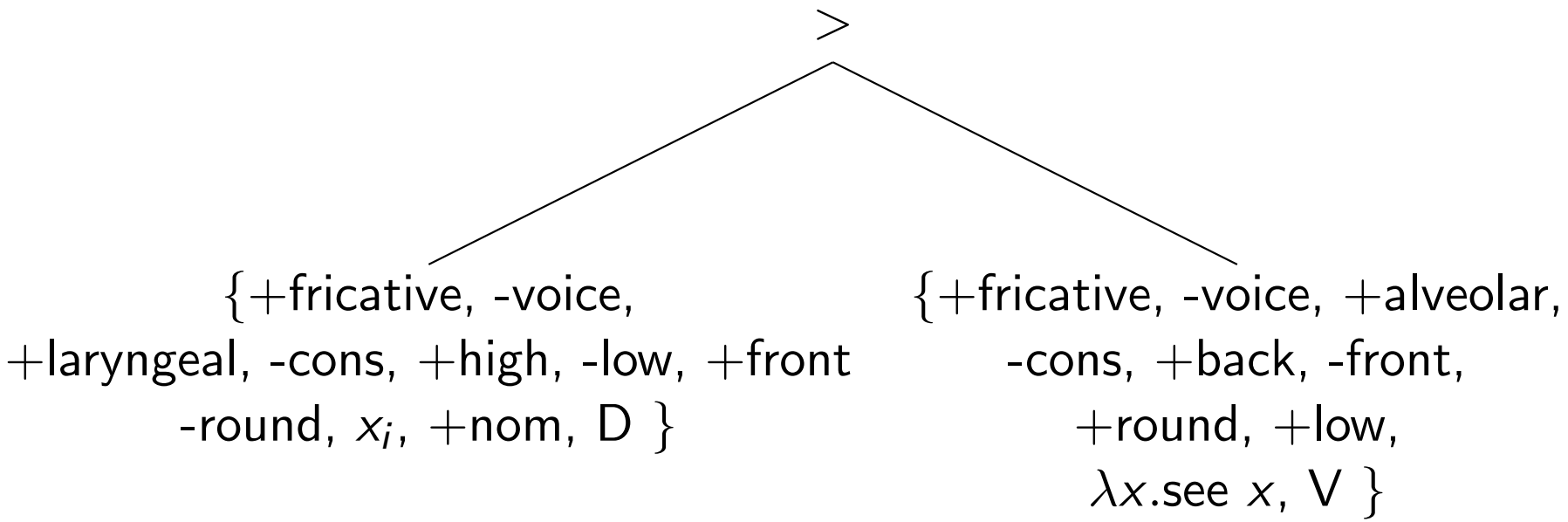
Describes what we did: "we Merged these two items"



Let's say Merge is feature-driven and this application of Merge is licensed by the matching features D (category D) and =D (uninterpretable D). Suppose we just delete the =D, and say D is interpretable by LF.

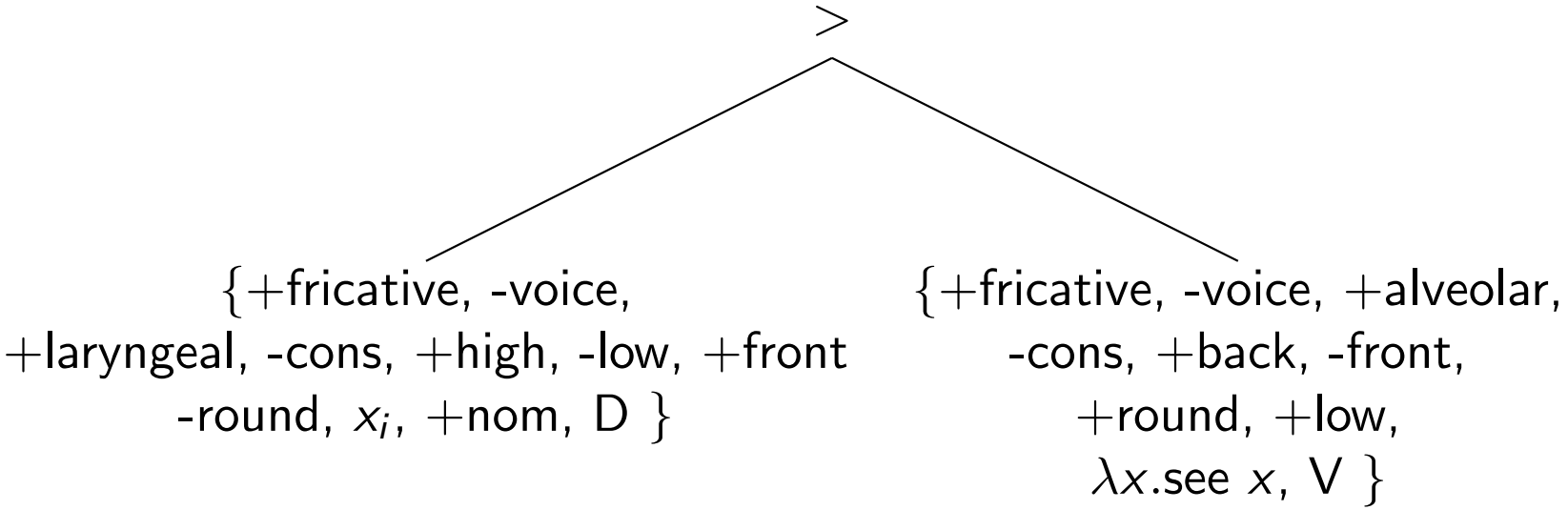
Bare phrase structure tree for *He saw*

The thing we built using Merge
(> or < points to head)
uninterpretable =D feature deletes on Merge?

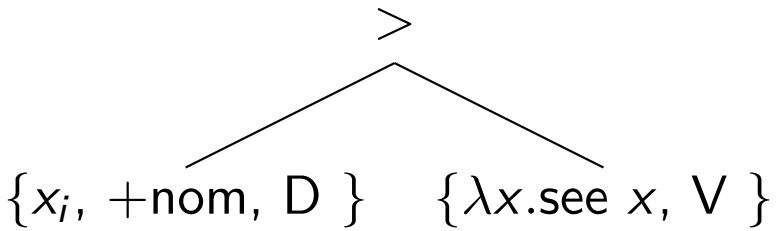


Spellout

Give subset of features to PF



$\rightarrow$



PF

⟨{+fricative, -voice, +laryngeal, -cons, +high, -low, +front, -round},

{+fricative, -voice, +alveolar, -cons,

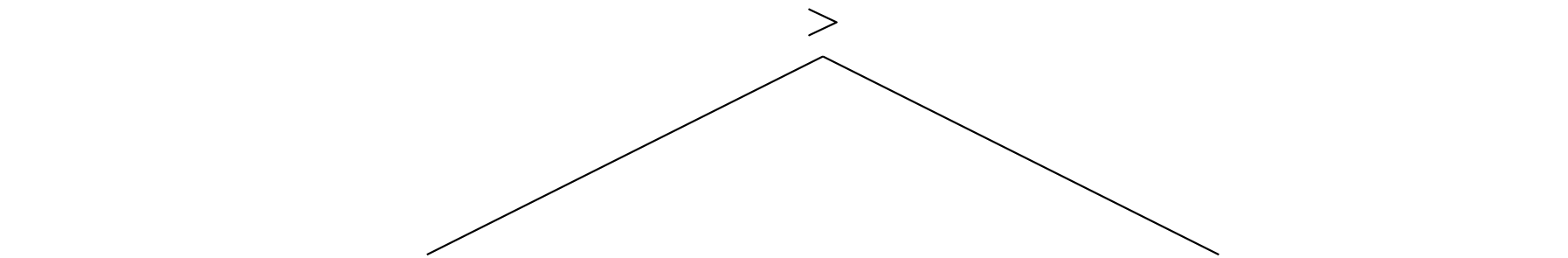
+back, -front, +round, +low} ⟩

What's wrong with this picture?

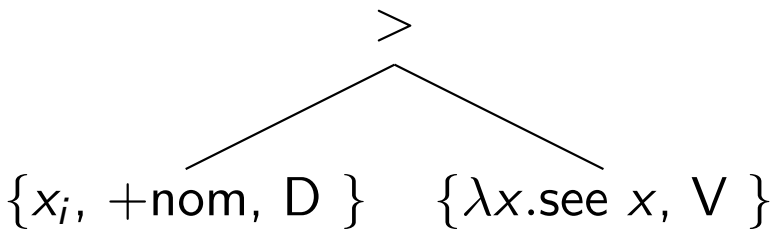
- Phonological features can't be just floating around in an (unordered) set. Sounds in a word come in an order

Spellout

Give subset of features to PF



→



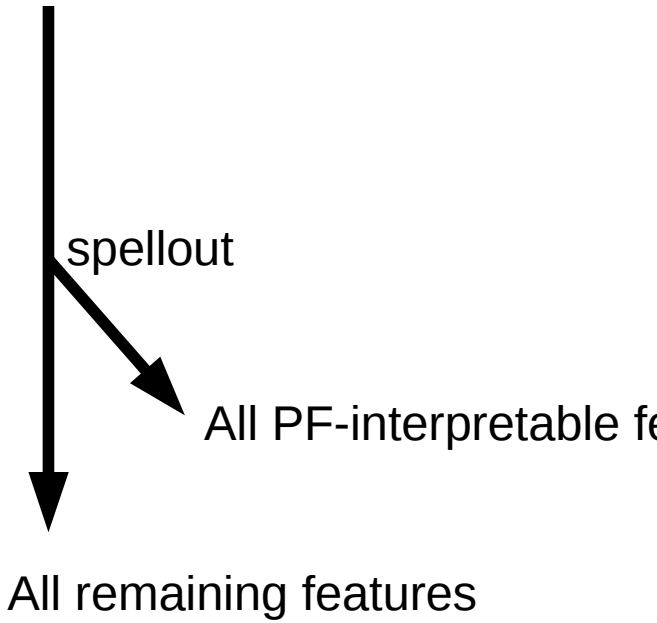
PF
⟨⟨{+fricative, -voice, +laryngeal},
{-cons, +high, -low, +front, -round}⟩,
⟨{+fricative, -voice, +alveolar},
{-cons, +back, -front, +round,
+low}⟩⟩

Revised system

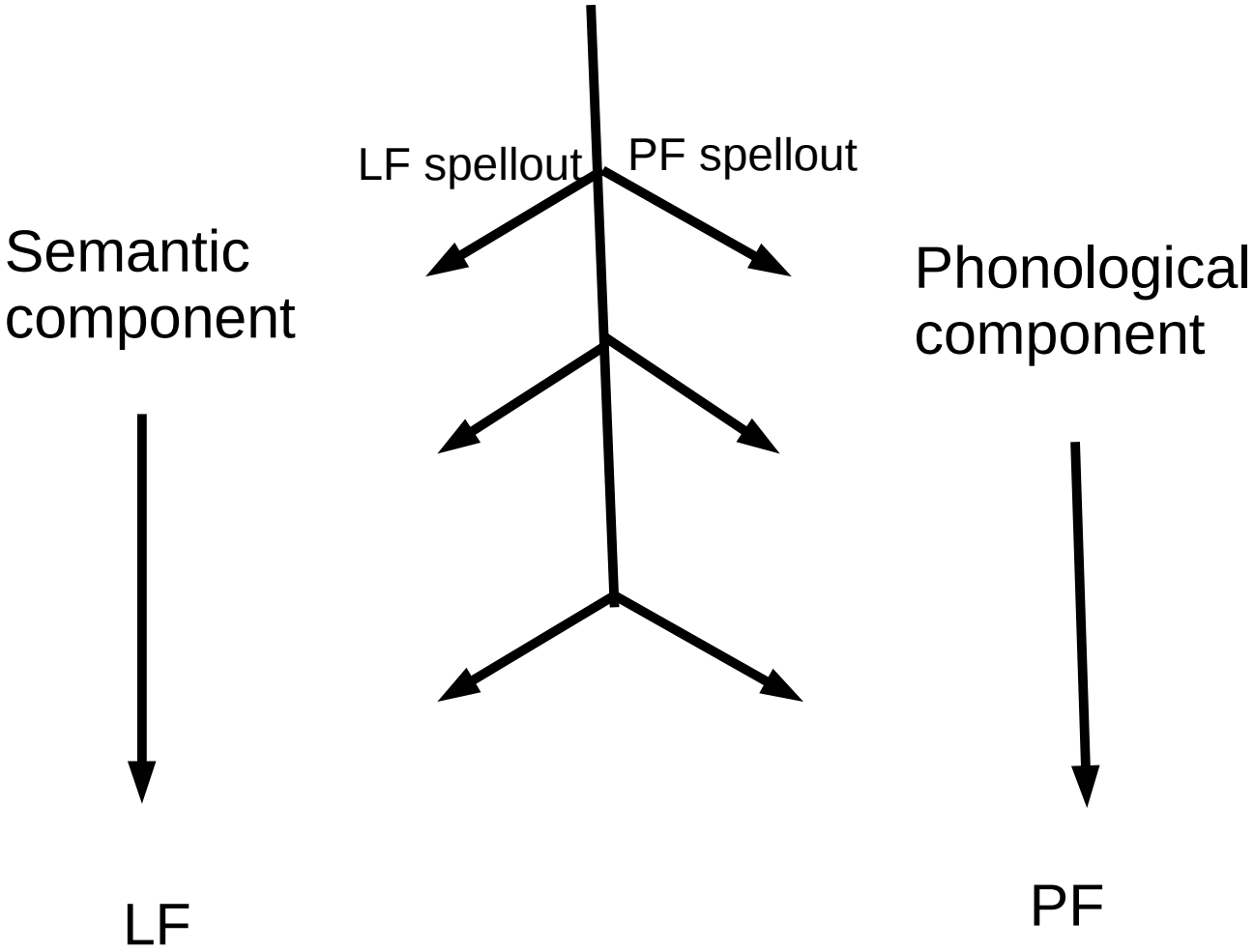
PF features are (include?) a sequence of sets of features.

Y-model properties

- LF gets whatever features are left over after Spellout
- All leftover features had better be interpretable by LF or else the derivation crashes
- At least some PF-interpretable features need to be sequences of sets of features
- Full Interpretation (all features left over at the end of the derivation are interpretable by an interface) is a consequence of the model under the mechanism of "crash if you get an uninterpretable feature"



Interface conditions – multiple spellout with LF spellout

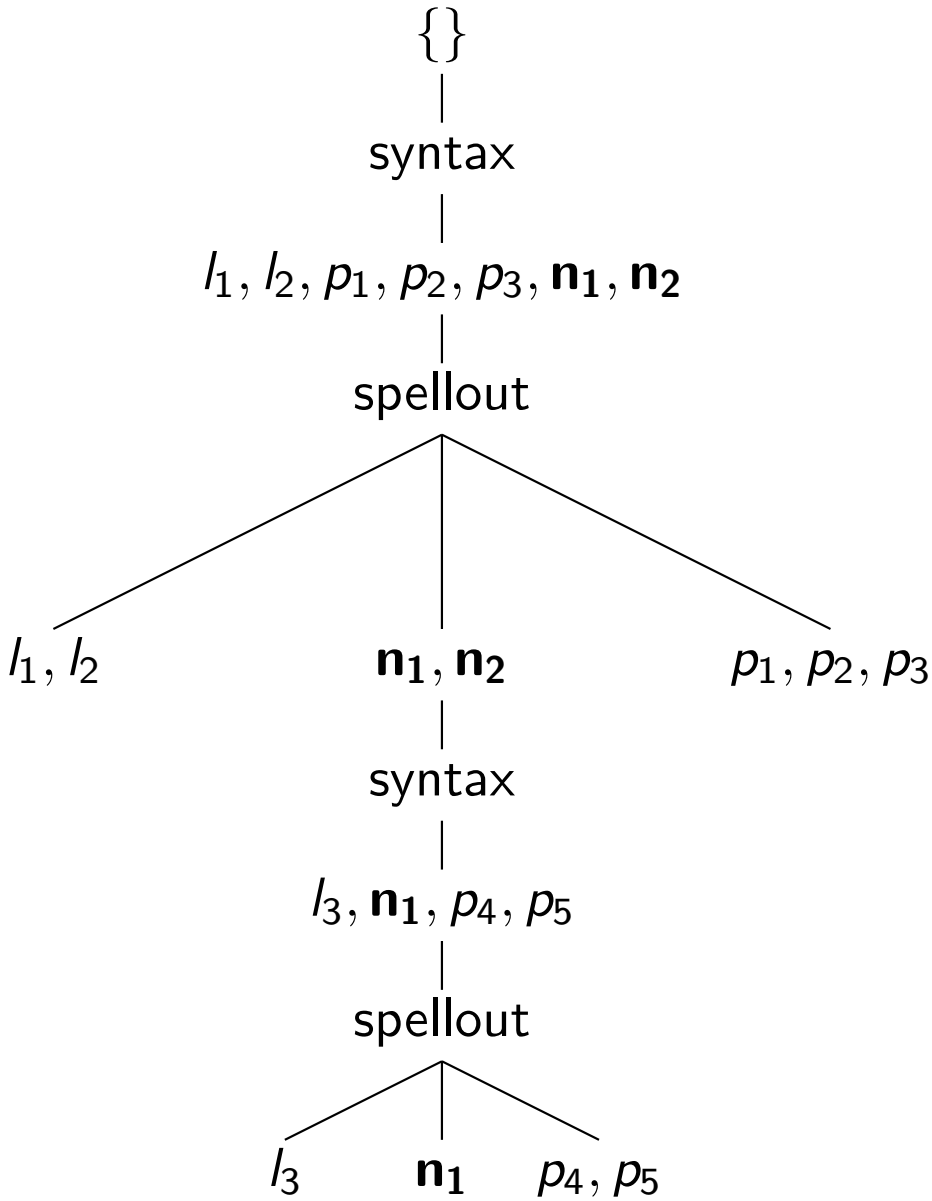


What is LF-spellout?

- PF spellout: give PF phonological features
- LF spellout: give LF semantic features

Notice neither of these say "give the interface whatever's leftover"

l_i = a feature interpretable by LF
p_i = a feature interpretable by PF
$\mathbf{n_i}$ = a feature interpretable at **neither** interface



What about that leftover feature n_1?

Interface conditions

- Spellout:= give each interface its interpretable features
- Crash if an interface gets uninterpretable features

$\rightarrow n_1$ *is fine!*
$\rightarrow$*There's nothing wrong with uninterpretable features staying in the structure*
$\rightarrow$*Why bother checking features?*

$\rightarrow$**Power of feature checking mechanism is lost**

Revised Interface conditions

- Spellout:= give each interface its interpretable features
- Crash if an interface gets uninterpretable features
- Crash if there are any features left over at the end

OR

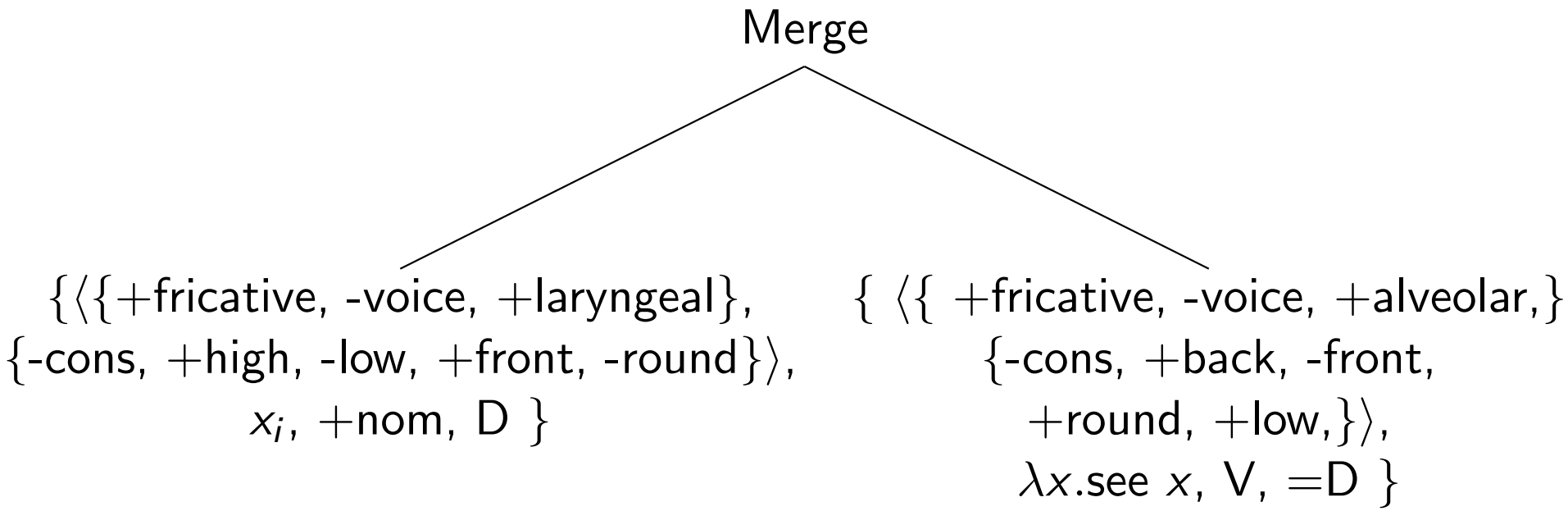
- Spellout:= give each interface its interpretable features
- Crash if an interface gets uninterpretable features
- Final Spellout := give PF its features and give LF the remaining features

Minimalism (Chomsky, 1995)

- Atoms of the derivation (words? morphemes?) have (are?) sets of features
- Basic operation: Merge := take two things and put them together
- Two Merges:
  - External Merge (or *Merge*): two things start separate
  - Internal Merge (or *Move*): one thing is a sub-part of the other
- Query: take two things and put them together to form *what*?
  - Chomsky (pc 2015): DEFINITELY (multi-)sets!!!!
  - Sequences?
  - Trees?
- Move/Internal Merge is driven by features. When you Move (or Agree?) you get to cancel pairs of matching features. (or just the uninterpretable one?)
- (External) Merge driven by features too? Or maybe it just has to create something interpretable by LF?

Derivation tree for *He saw*

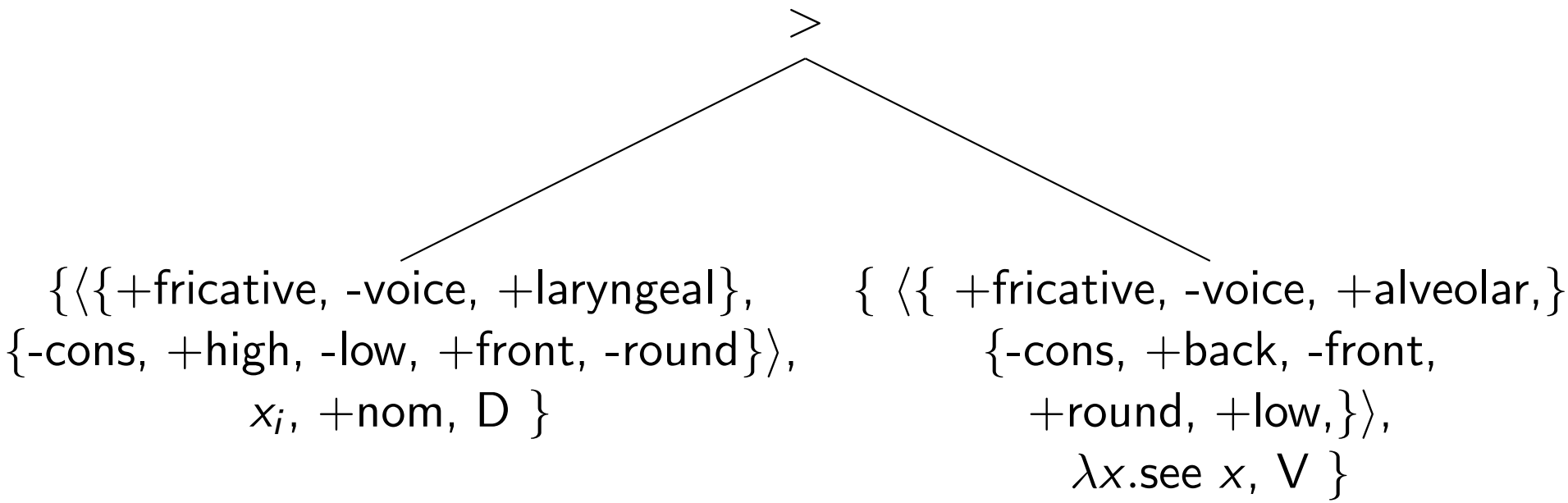
Describes what we did: "we Merged these two items"



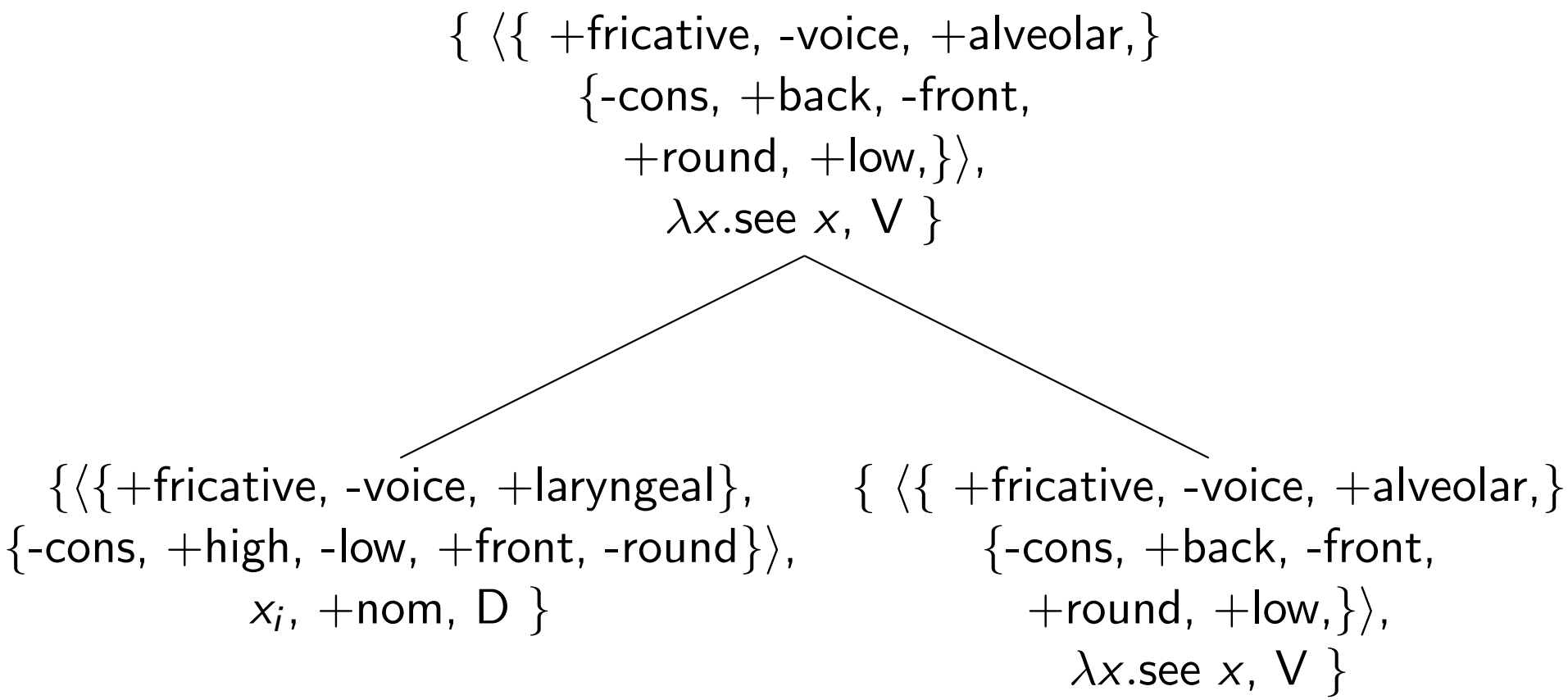
Let's say Merge is feature-driven and this application of Merge is licensed by the matching features D (category D) and =D (uninterpretable D). Suppose we just delete the =D, and say D is interpretable by LF.

Bare phrase structure tree for *He saw*

The thing we built using Merge
(> or < points to head)

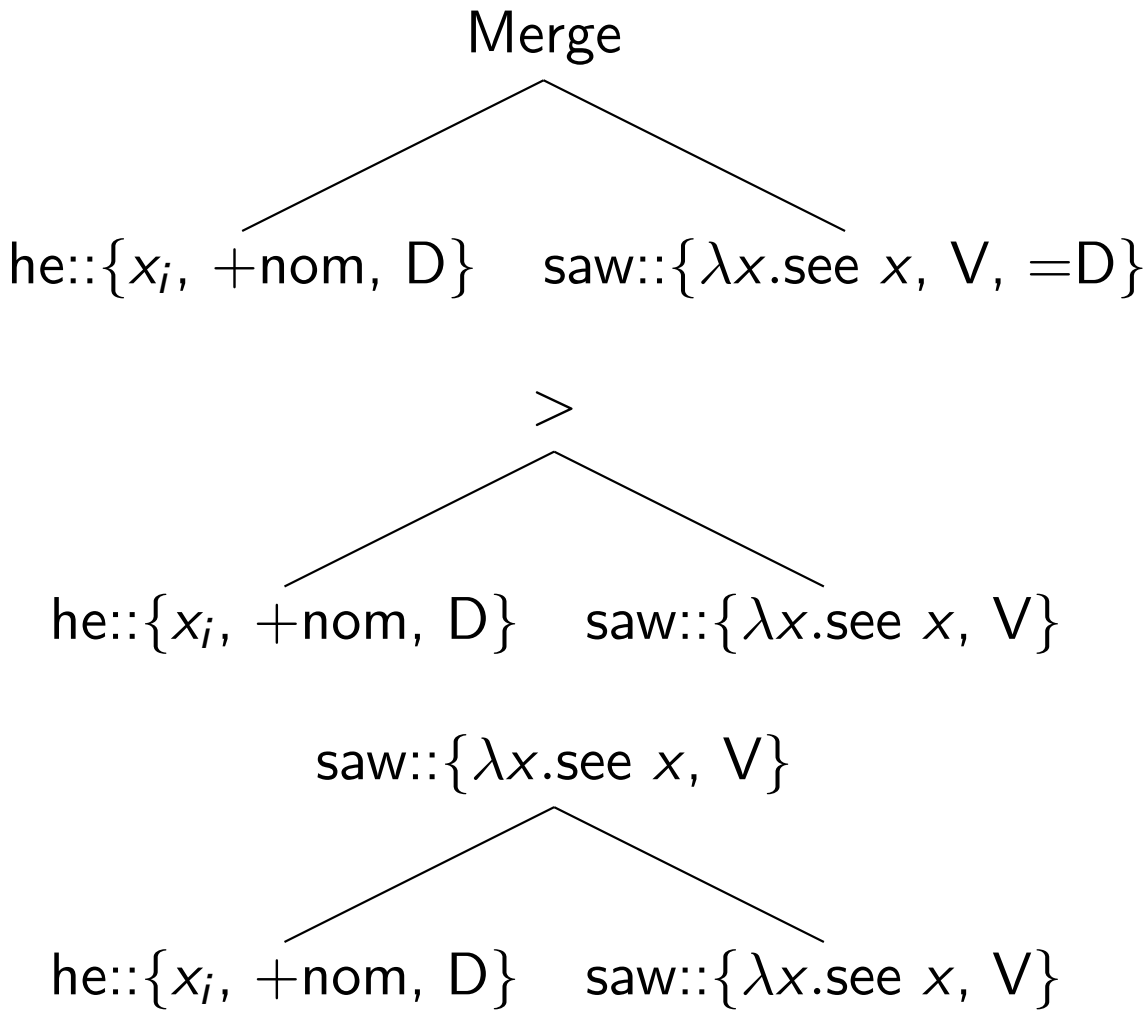


Set inclusion tree + head



Nicer trees

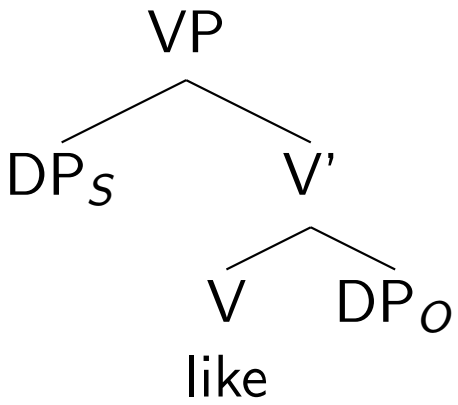
Recall PF features needed to be bundled anyway.



Subcategorisation frames

Recall from P& P era:

Word	meaning	Cat			
sleep	$\lambda x.sleep\ x$	V	DP		
like	$\lambda y(\lambda x.like\ x)y$	V	DP	DP	
put	$\lambda z(\lambda y(\lambda x.put\ x)y)z$	V	DP	DP	PP



Subcategorisation frames

To get subcategorisation within a minimalist framework, all you need to do is

1. Separate the LF features from the pure syntax features
   - Probably okay since we don't really need "leftover" features to be syntactic. If we do want category features in the LF we can give LF syntax features too.
2. Make the syntax features a list (stack) instead of an unordered set
   - Who says sequences are less simple than sets anyway? Certainly sequences (especially stacks) are simpler in computing.

Subcategorisation frames

Word	meaning	Cat			
sleep	$\lambda x.\mathit{sleep}\ x$	V	DP		
like	$\lambda y(\lambda x.\mathit{like}\ x)y$	V	DP	DP	
put	$\lambda z(\lambda y(\lambda x.\mathit{put}\ x)y)z$	V	DP	DP	PP

$\rightarrow$

- $\langle$sleep, $\lambda x.\mathit{sleep}\ x$, =D V$\rangle$
- $\langle$like, $\lambda y(\lambda x.\mathit{like}\ x)y$, =D =D V$\rangle$
- $\langle$put, $\lambda z(\lambda y(\lambda x.\mathit{put}\ x)y)z$, =D =P =D V$\rangle$

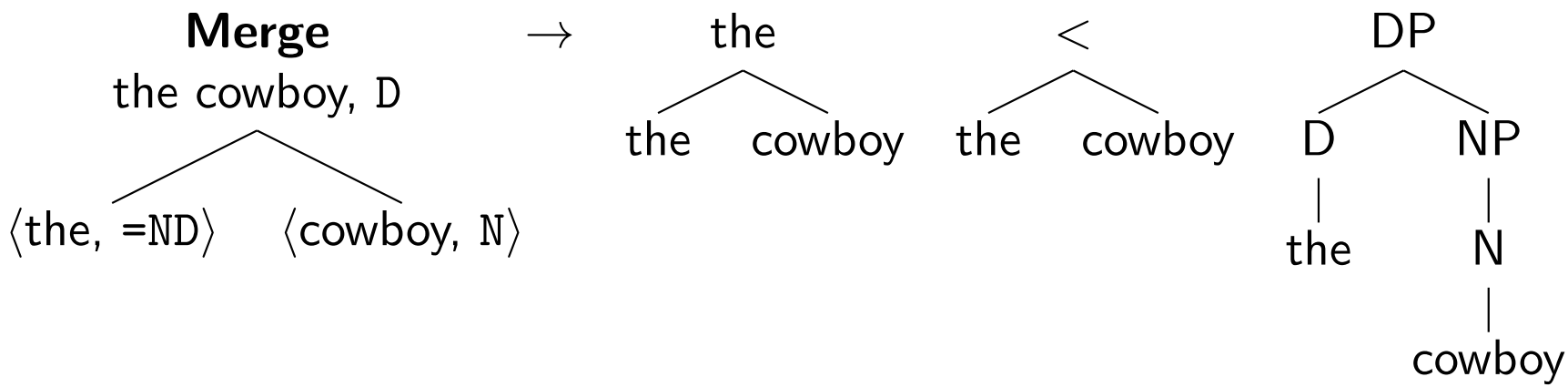
Merge Features

=X *I need an X*

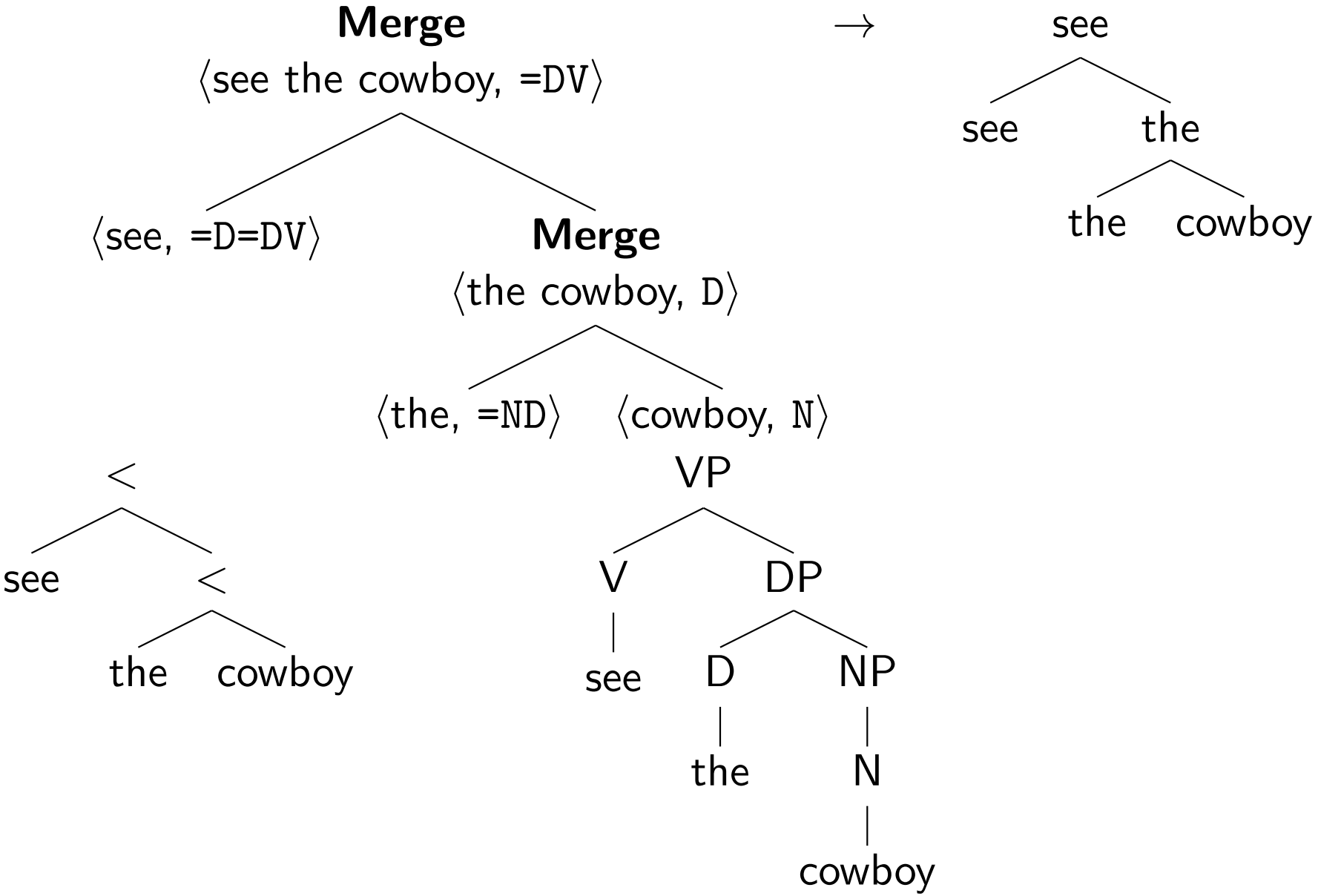
X *My category is X*

Example

Example: Merge



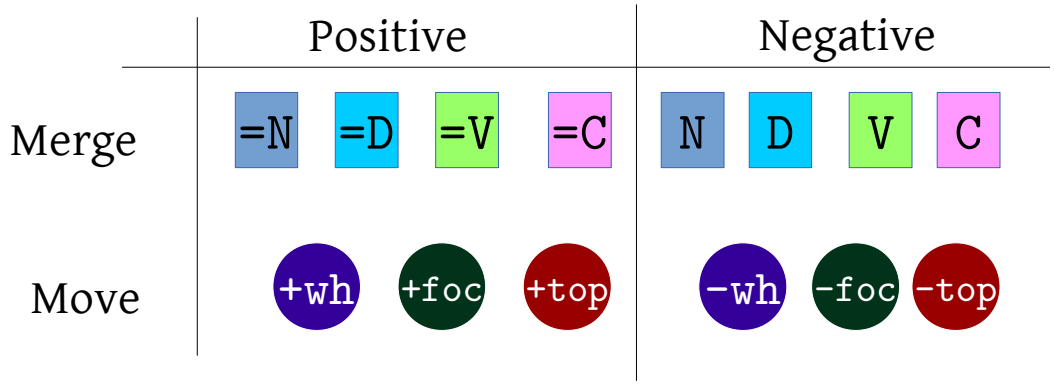
Example: Merge



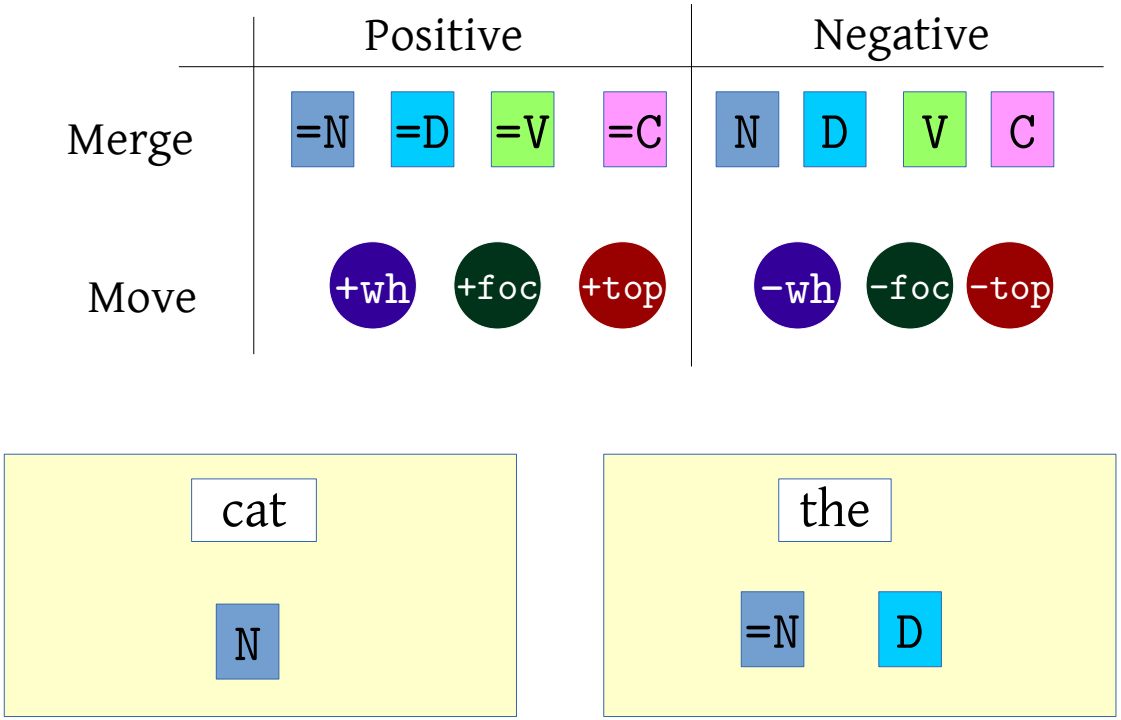
Minimalist Grammars (Stabler, 1997)

- Ed Stabler's (1997 etc.) formalisation of Chomsky (1995) Minimalism
- Two operations: Merge and Move (or External and Internal Merge)
- Merge & Move both say "put 2 things together when their features match"
- Merge: get second element from Lexicon/Numeration
- Move: get second element from Mover List

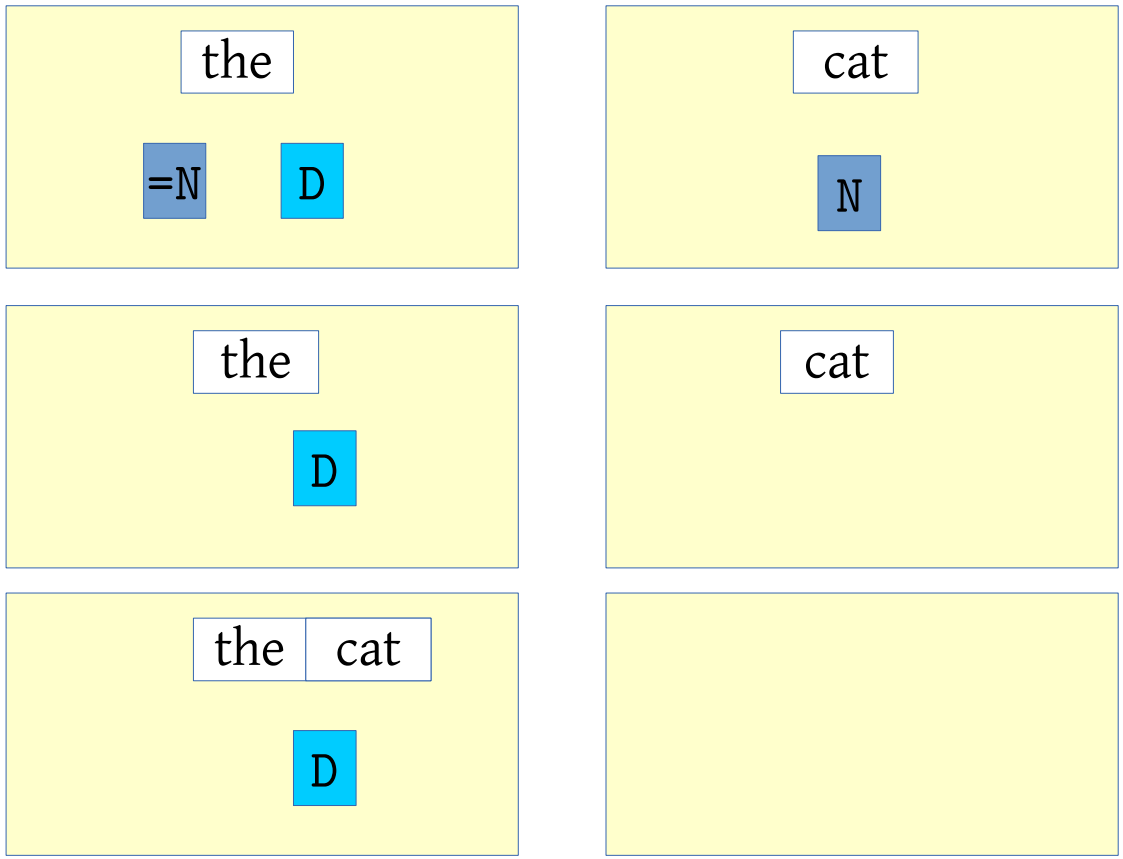
Features



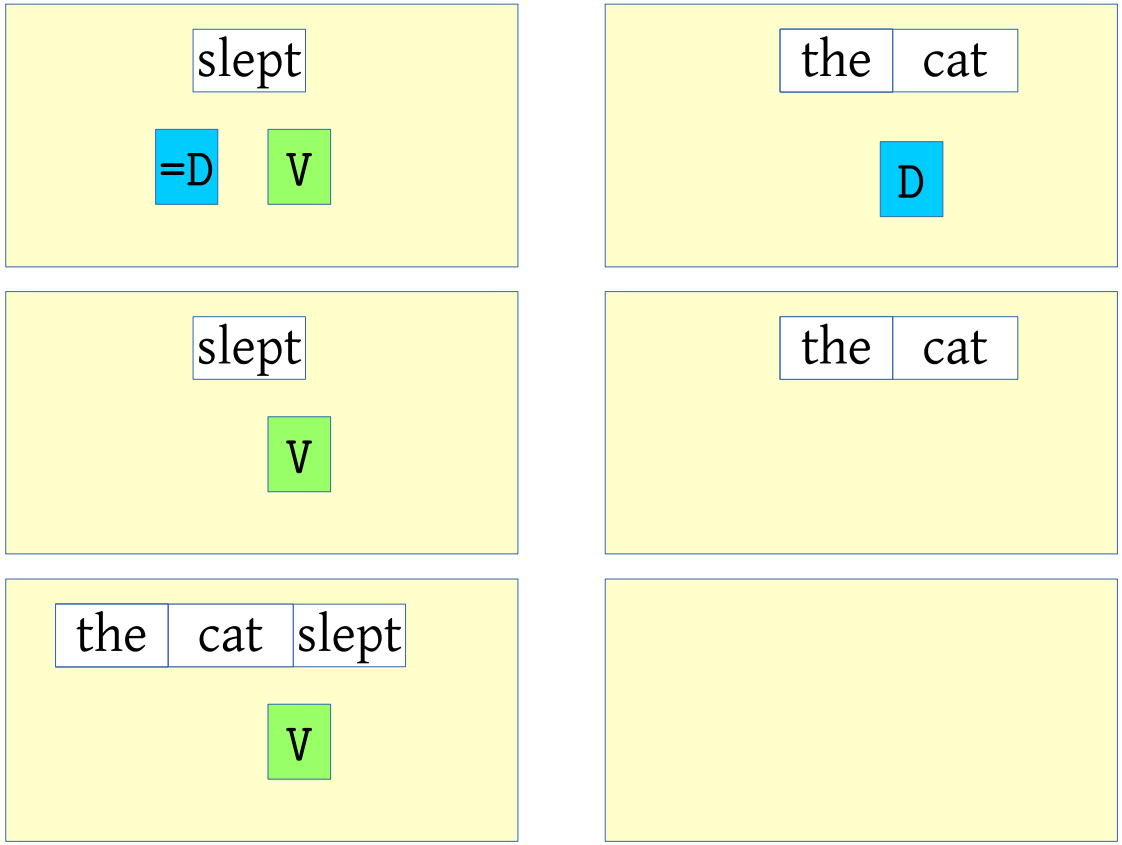
Features


Positive
Negative
Merge
=N
=D
=V
=C
N
D
V
C
Move
+wh
+foc
+top
-wh
-foc
-top
cat
N
the
=N
D

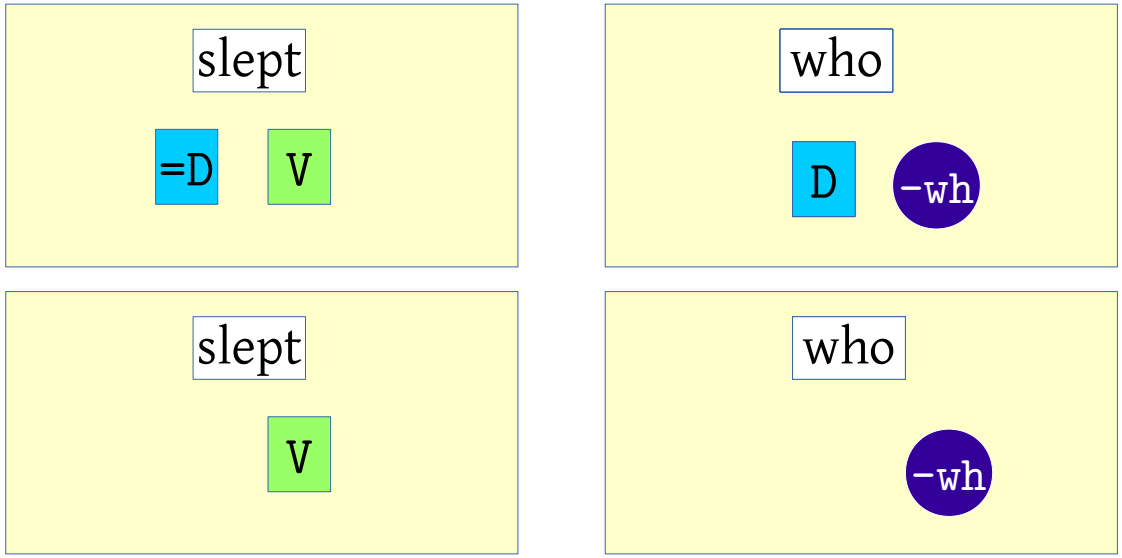
Merge



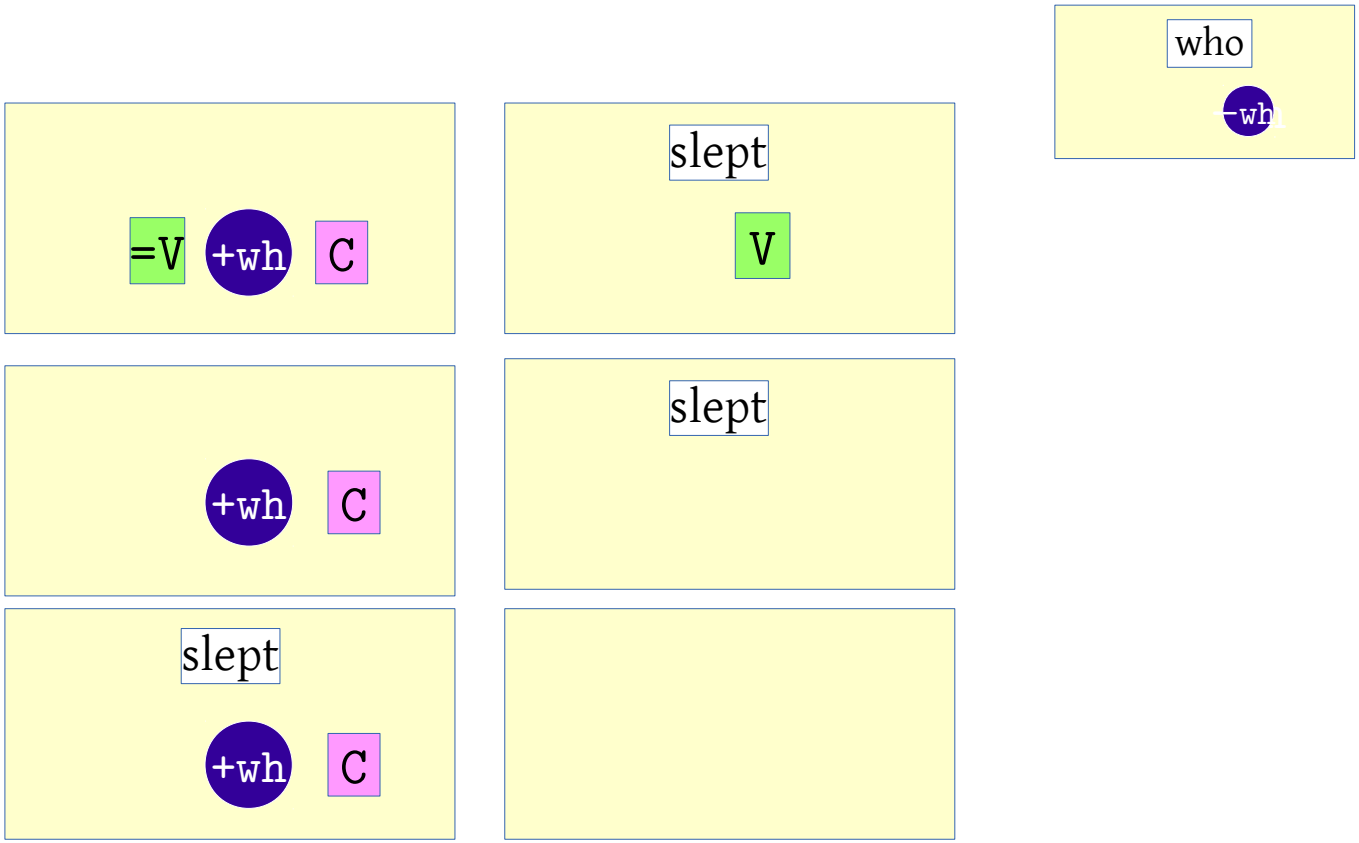
Merge



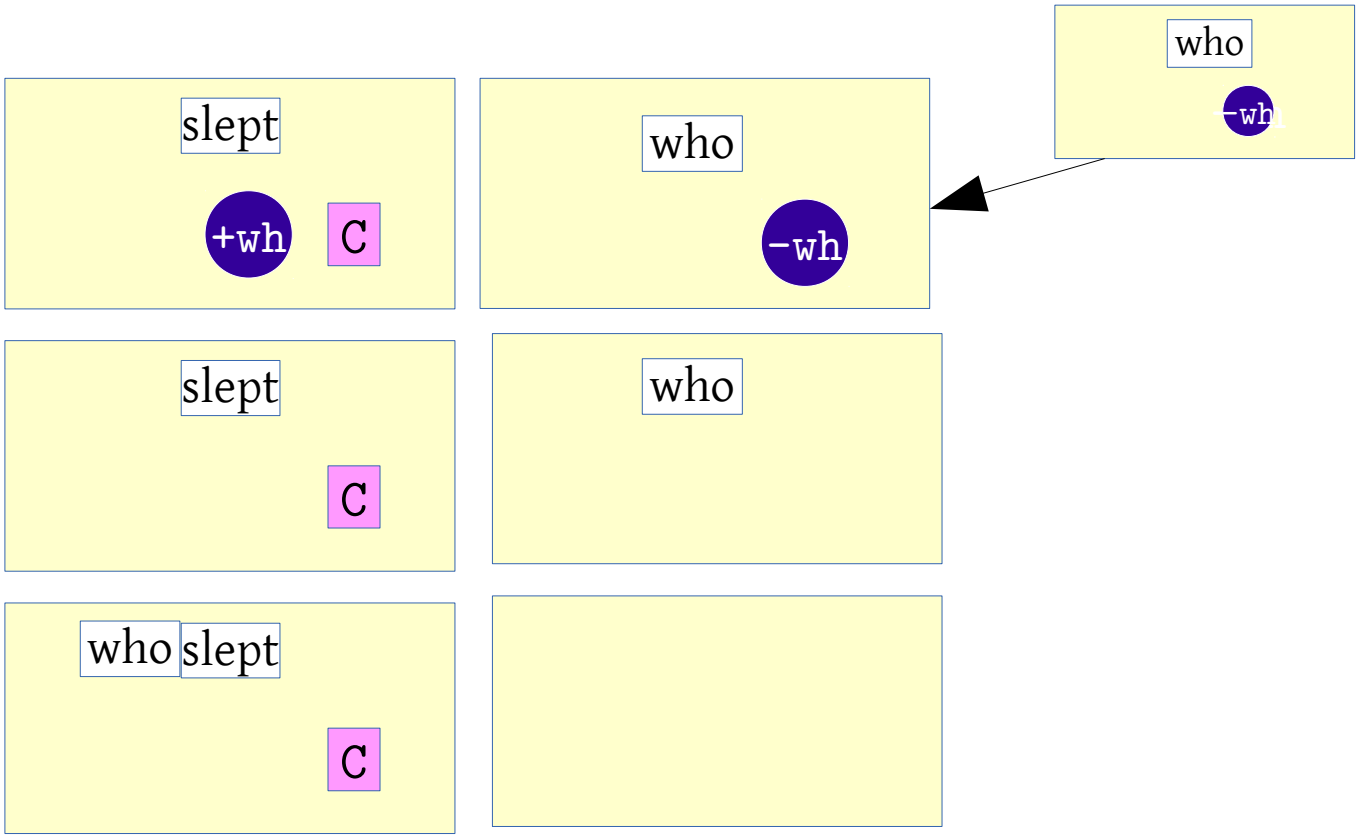
Merge a Mover



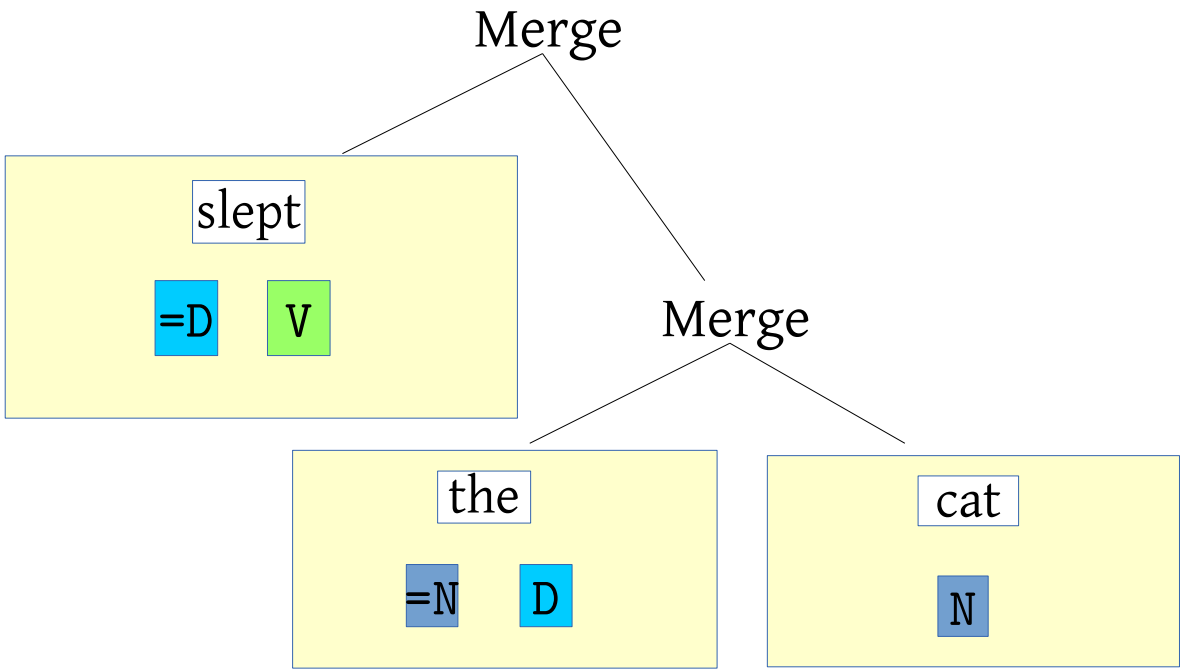
Merge (with waiting Mover)



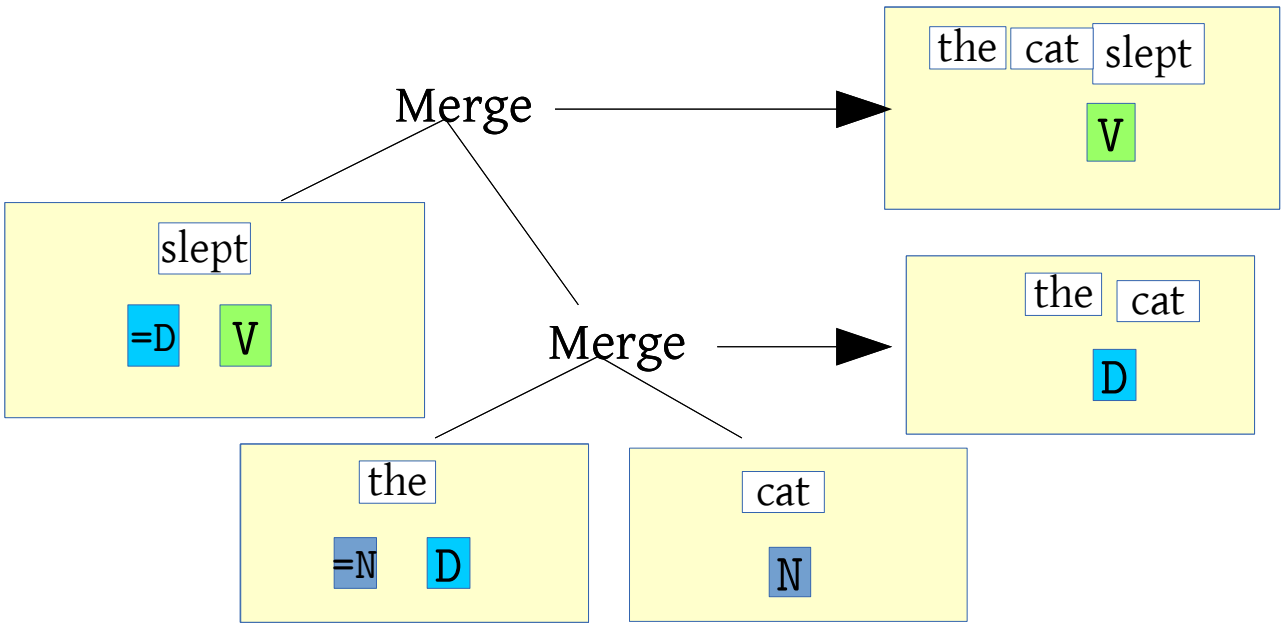
Move



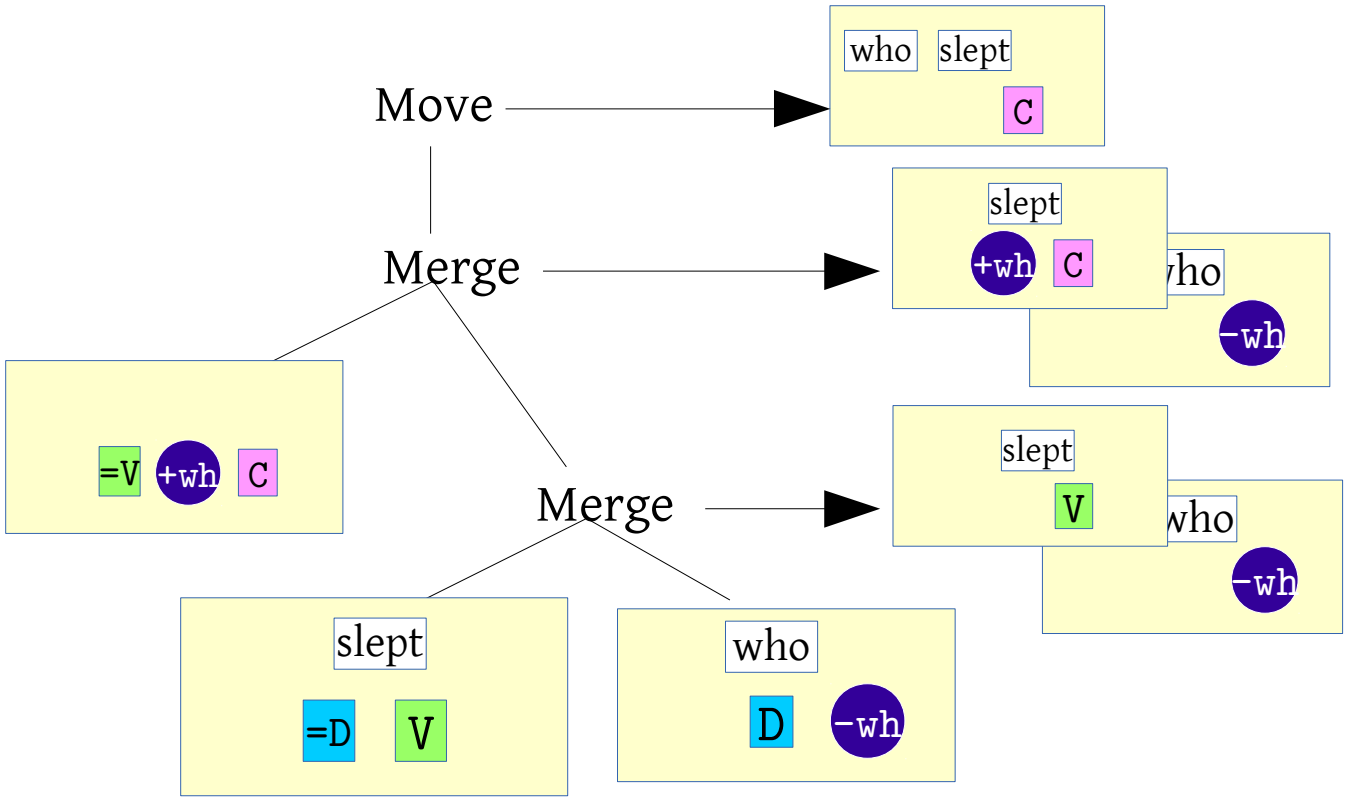
Derivation Trees



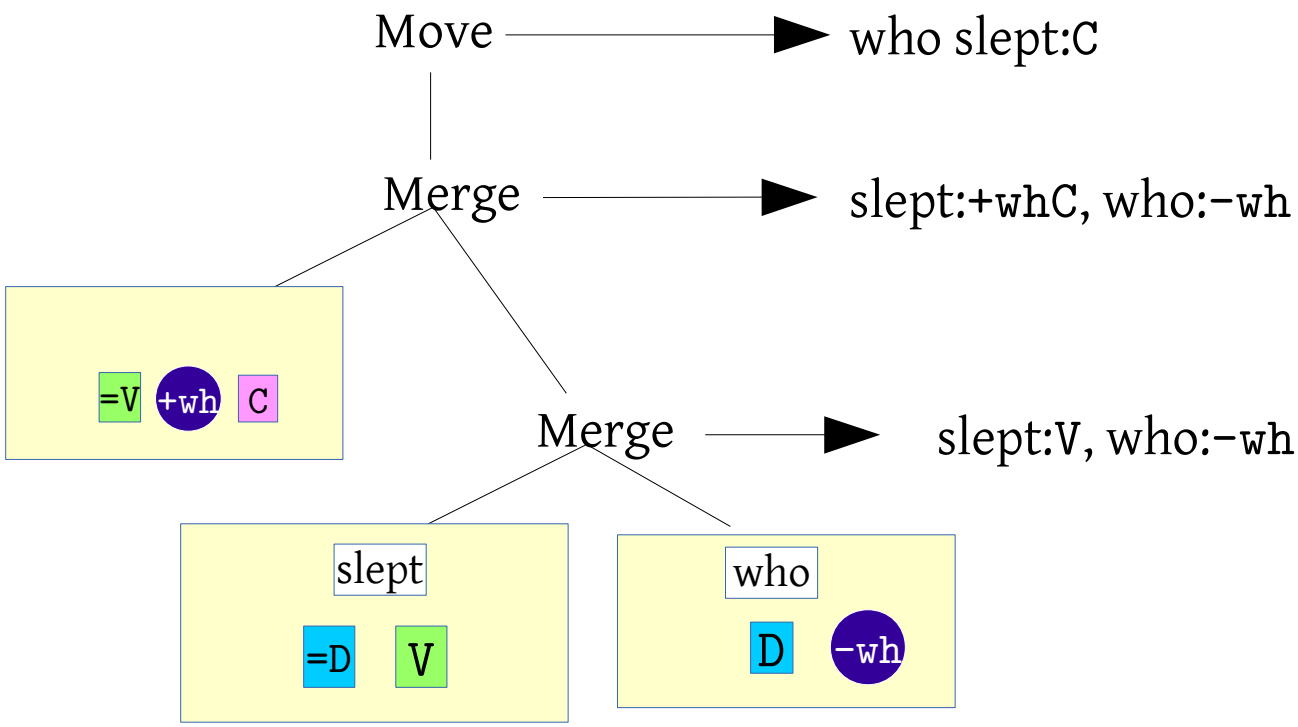
Derivation Trees



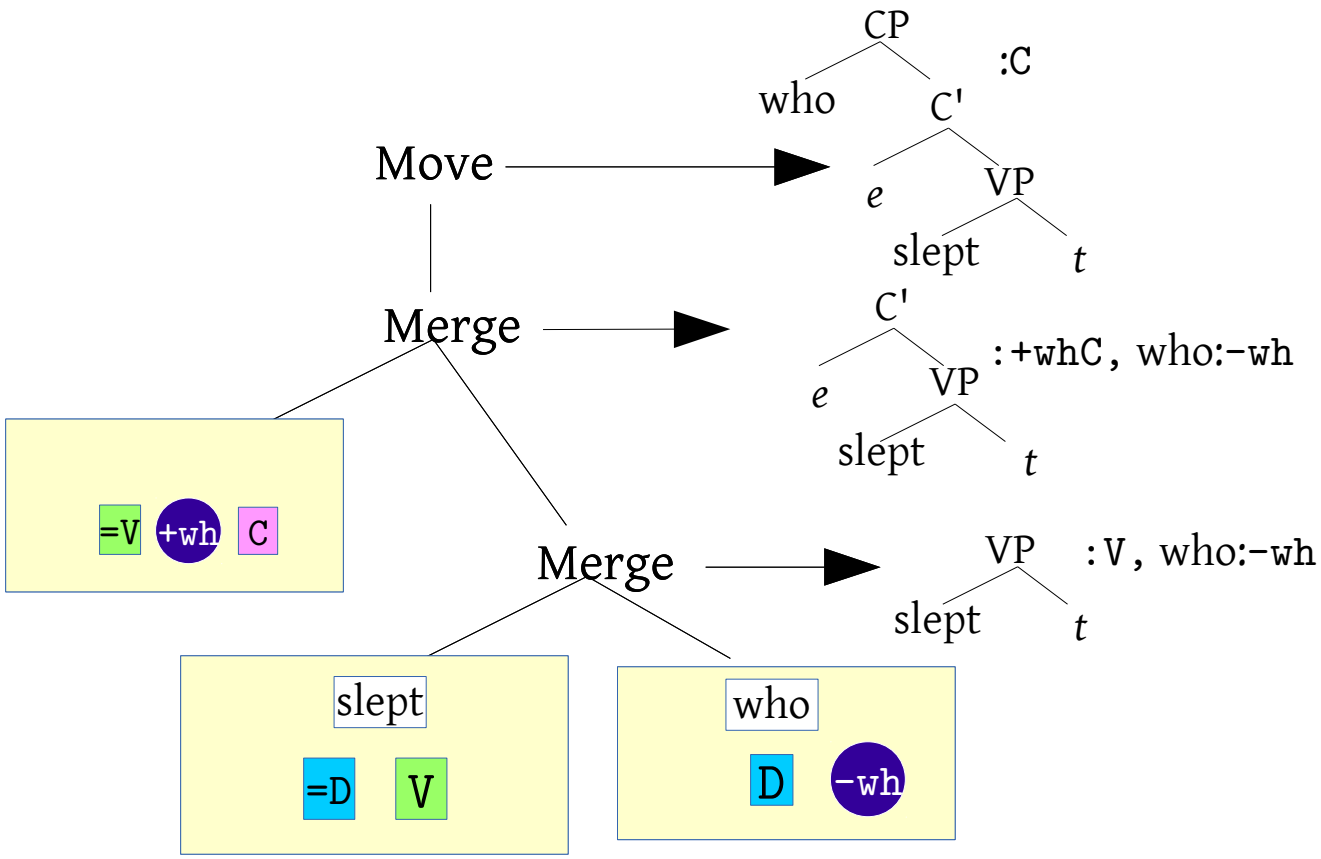
Derivation Trees



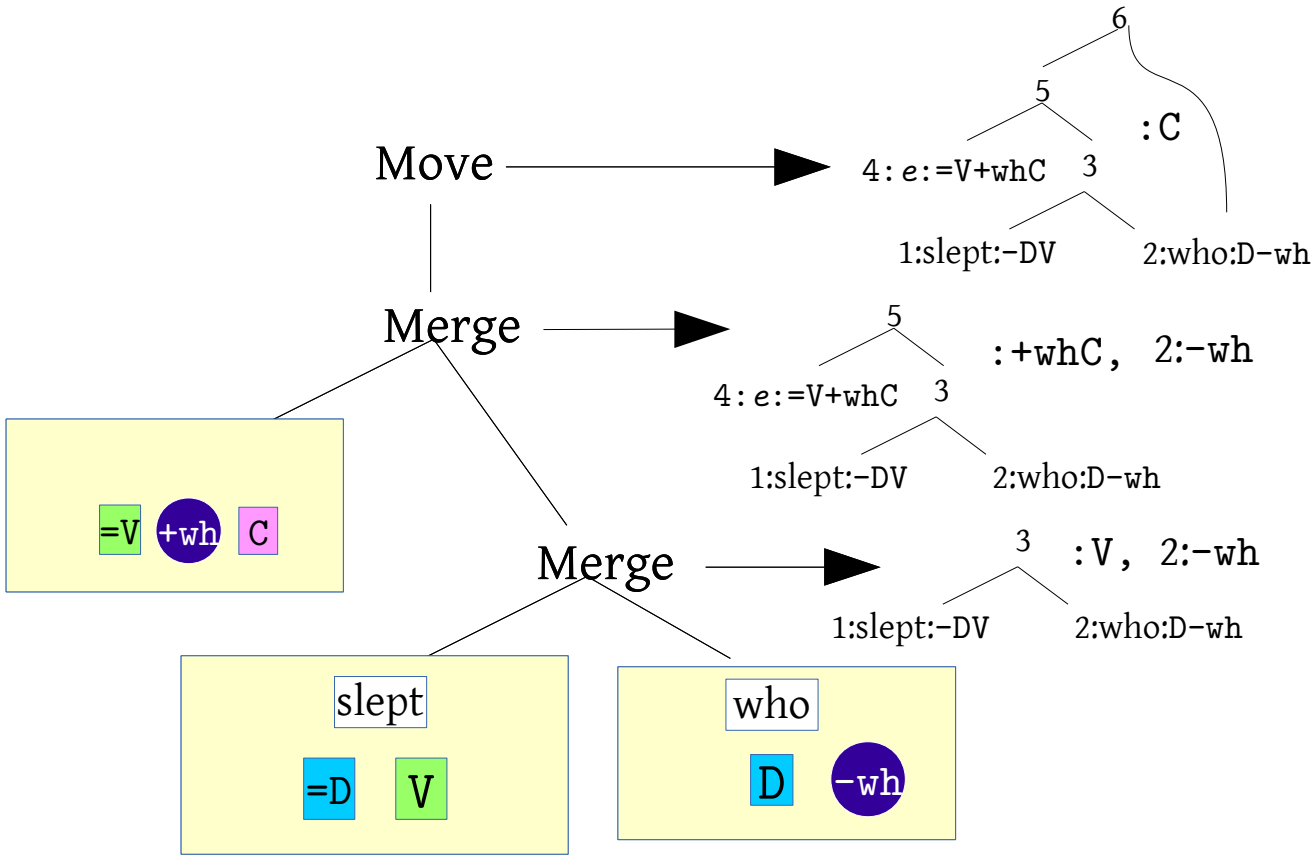
Derivation Trees



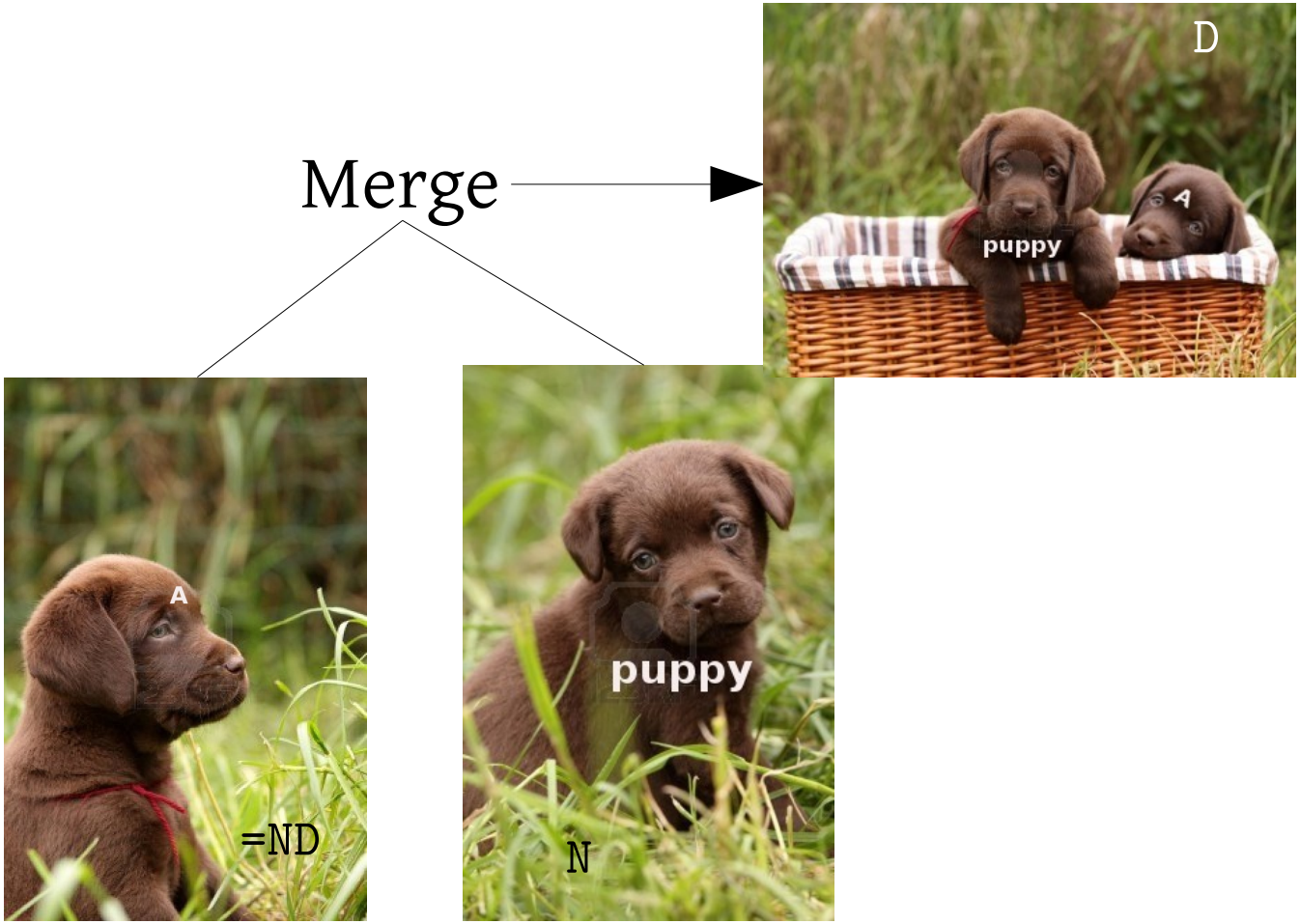
Derivation Trees



Derivation Trees



Derivation Trees



Minimalist Grammars as a formalisation of Minimalism

- Cancel one or both features? →**both**
- Is Merge feature-driven? →**Yes**
- Which features get cancelled when? →**the ones on the tops of the stacks**
- How do you know which features drive the syntax? →**features are divided into three sets: PF, LF, syntactic**
- When does the derivation crash? →
  - Derivational view:
    - If you can't put together what you want to because the features don't match
    - If you finish the derivation and you don't have a "final" category feature (like `T` for TP or `C` for CP)
  - Transderivational constraint view:
    - If your final derived structure has any syntactic features other than a single "final" category

Adjuncts

Generally adjectives, adverbs, prepositional phrases

(1) a. My love is like a rose.
b. My love is like a red red rose.

(2) a. I'm tired!
b. I'm really really really really tired!

(3) He suddenly (*suddenly suddenly) smiled.

Properties of adjuncts to be captured by a grammar

(4)

a. The (bad) wolf — *optional*

b. The bad wolf — *transparent to selection*

c. The big bad wolf

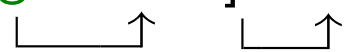
d. *The bad big wolf — *strictly ordered*

e. The Alliance officer shot Kaylee in the cargo hold with a gun

f. The Alliance officer shot Kaylee with a gun in the cargo hold

Unordered

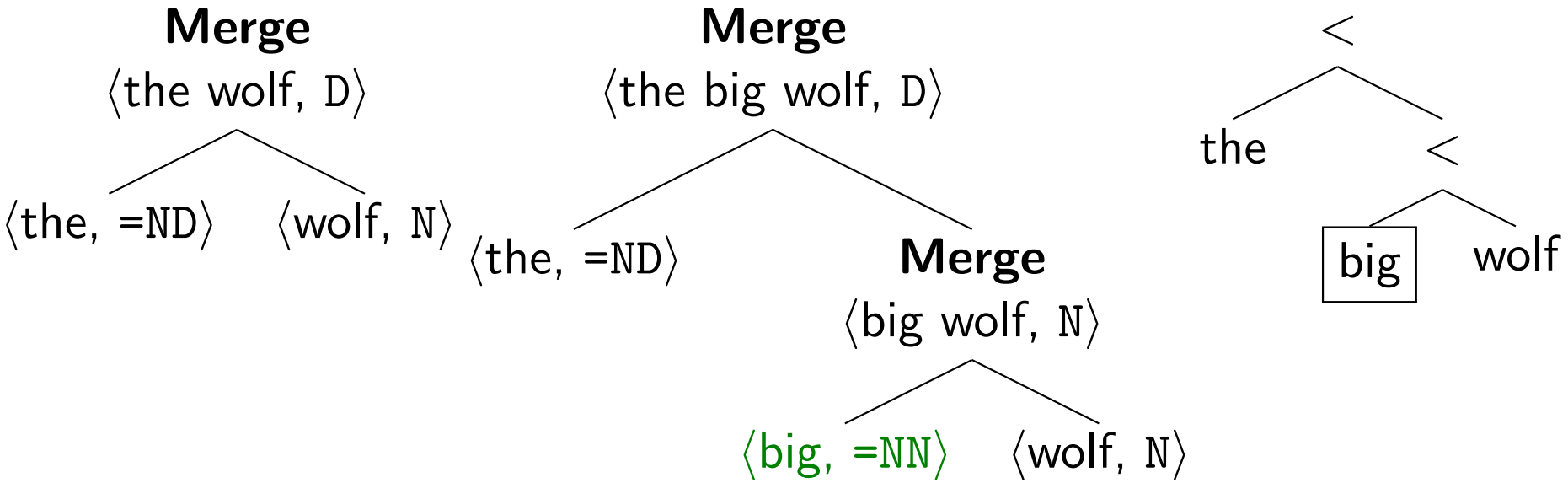
g. [bright blue] balloon — *Adjuncts of adjuncts*



h. Kaylee is clever. — *Selectable category*

Traditional MG/Categorial Grammar approach

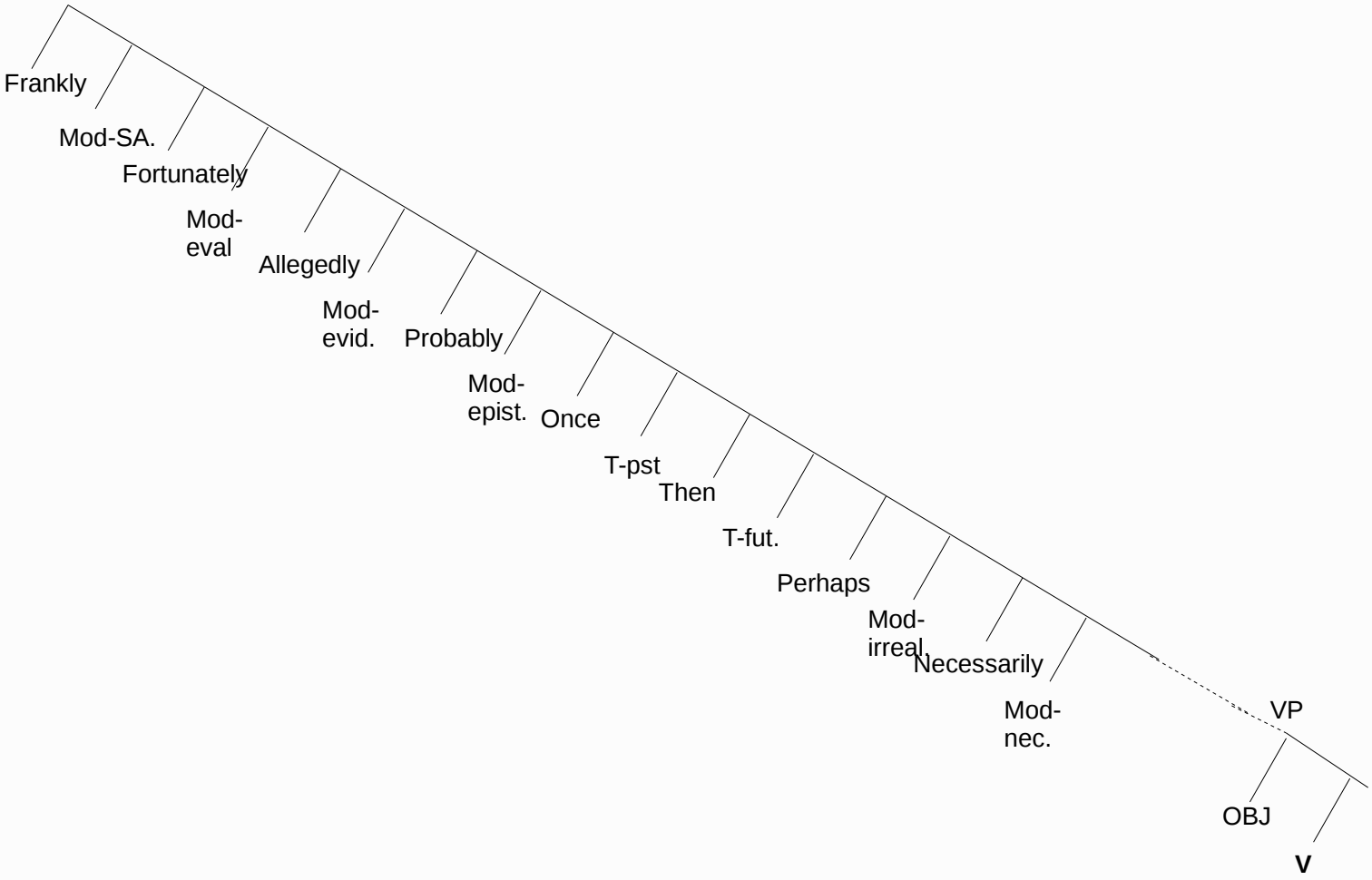
- X-Modifier features: (Categorial Grammar: X/X or X\ X) =XX; Verbal modifier: =VV; Nominal modifier: =NN etc.
- ✓Optionality
- ✗ Ordering
- ✗ Transparent to selection



Traditional MG/Categorial Grammar approach

	Trad. (=XX)	Cart. (=A_5A_6)	MGAs ([X,i,j])
Optionality	✓		
Selector selects expected category	✓		
Adjunct does not become head	✗		
Unordered adjuncts possible	✓		
Ordered adjuncts possible	✗		
Adjuncts of adjuncts	✗		
Selectability	✗		

Cartography: adverbs

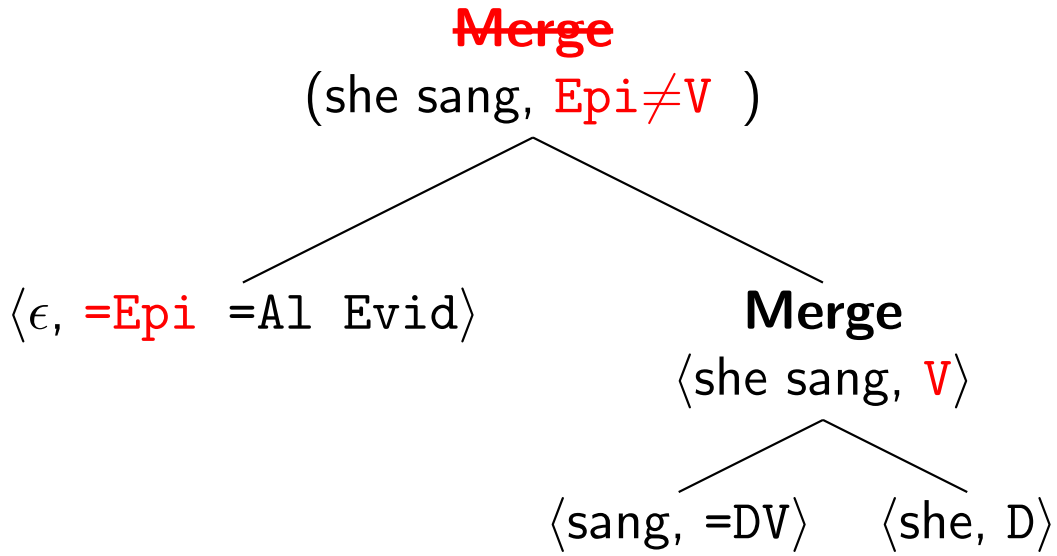


Problem

Allegedly, she sang

Lexicon:

- (Allegedly, Al)
- (ϵ, =Epi =Al Evid)
- (probably, Pr)
- (ϵ, $=T_{pst}$ =Pr T_{fut})
- ...
- (ϵ, =V =Compl Asp_{compl})
- (she, D)
- (sang, =D V)



Solution: silent, meaningless LIs

Lexicon:

- (allegedly, ⟦allegedly⟧, Al)
- (ϵ, ⟦evid⟧, =Epi =Al Evid)
- (ϵ, **id**, =Epi Evid)
- (ϵ, ⟦prob⟧, $=T_{pst}$=Prob Epi)
- (ϵ, **id**, $=T_{pst}$Epi)
- (ϵ, ⟦past⟧, $=T_{fut}T_{pst}$)
- (ϵ, **id**, $=T_{fut}T_{pst}$)
- ...
- (ϵ, ⟦compl⟧, =V=Compl Asp_{Compl})
- (ϵ, **id**, =V Asp_{Compl})
- (she, D)
- (sang, =D V)

Cartography – Properties

	Trad. (=XX)	Cart. (=A_5A_6)	MGAs ([X,i,j])
Optionality	✓	✗	
Selector selects expected category	✓	✗	
Adjunct does not become head	✗	✗	
Unordered adjuncts possible	✓	✗	
Ordered adjuncts possible	✗	✓	
Adjuncts of adjuncts	✗	✗	
Selectability	✗	✗	

Cartography: summary

If we want to keep:

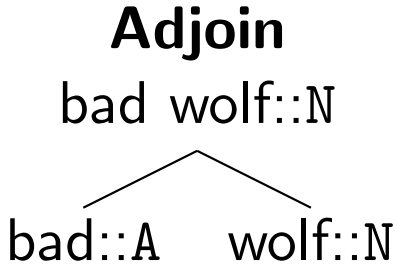
- Merge driven by features
- Adjunct ordering is syntactic
- Just Merge and Move

Then we need:
Silent, meaningless, non-specifier-selecting versions of each functional head, yielding a full Cinque hierarchy in every sentence

Minimalist Grammars with Adjunction (MGAs)

Proposal

Adjoin is optional → Add an optional operation **Adjoin** that applies to full phrases. Resulting phrase has the category of the adjoined-to phrase.



Fowlie (2014)

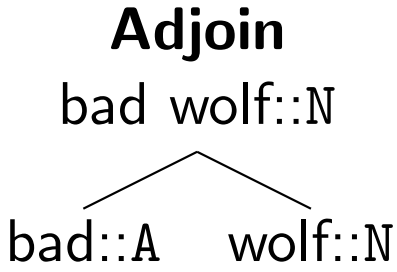
MGAs

Wait! Adjunction doesn't occur between just any to phrases $\rightarrow$ add to the grammar a set of adjuncts for each category

$\mathbf{Ad} : \mathbf{sel} \rightarrow \mathcal{P}(\mathbf{sel})$

eg $\mathbf{Ad}(\mathtt{N}) = \{\mathtt{A}, \mathtt{P}, \mathtt{C}\}$

MGAs



- Optional
- Category-preserving
- Applies to complete phrases
- Specifies which phrases can adjoin to which phrases
- Adjuncts have their own categories ($\rightarrow$ selectable, adjoin-able, intuitive)

MGAs

Wait! What about adjunct ordering?

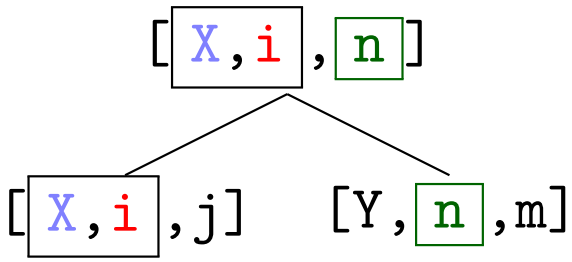
Add indicies to track hierarchy level of most recent adjunct.

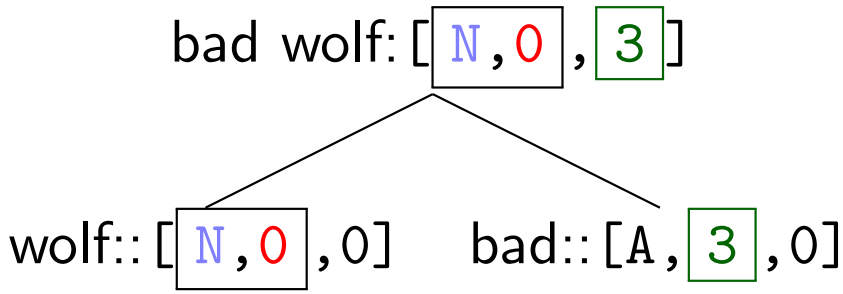
big::A $\rightarrow$big::[A,5,0]

bad::A $\rightarrow$bad::[A,3,0]]

Indices stand for ordered semantic classes of adjuncts

Example





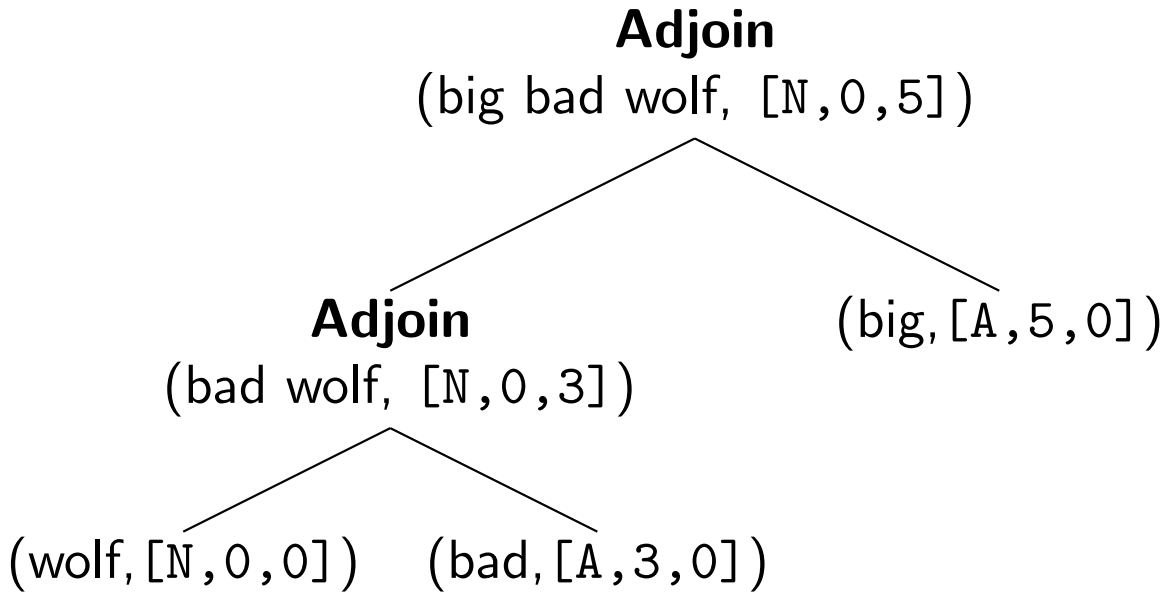
Adjoin example

big bad wolf

$\mathbf{ad}(\mathtt{N}) = \{\mathtt{A}, \mathtt{P}, \mathtt{C}\}$

Lexicon:

- $\langle$bad, [A,3,0]$\rangle$
- $\langle$big, [A,5,0]$\rangle$
- $\langle$the, =N[D,0,0]$\rangle$
- $\langle$wolf, [N,0,0]$\rangle$



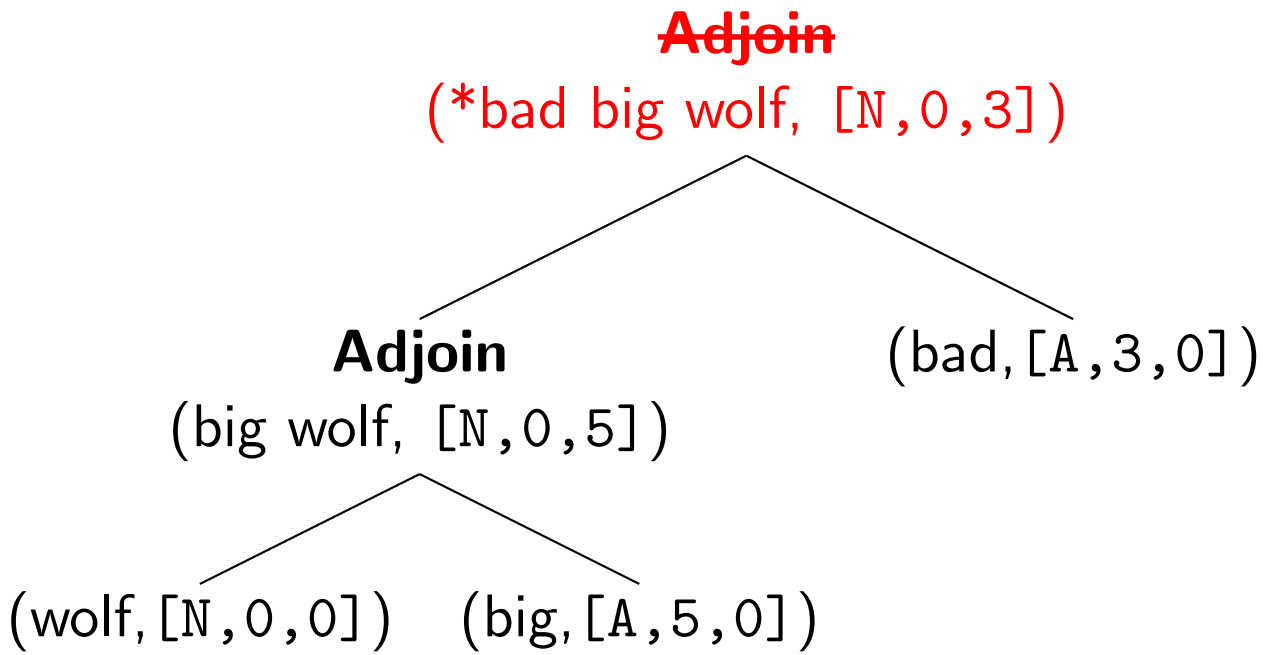
Failed example: bad adjunct order

*The bad big wolf

$\mathbf{ad}(\mathtt{N}) = \{\mathtt{A}, \mathtt{P}, \mathtt{C}\}$

Lexicon:

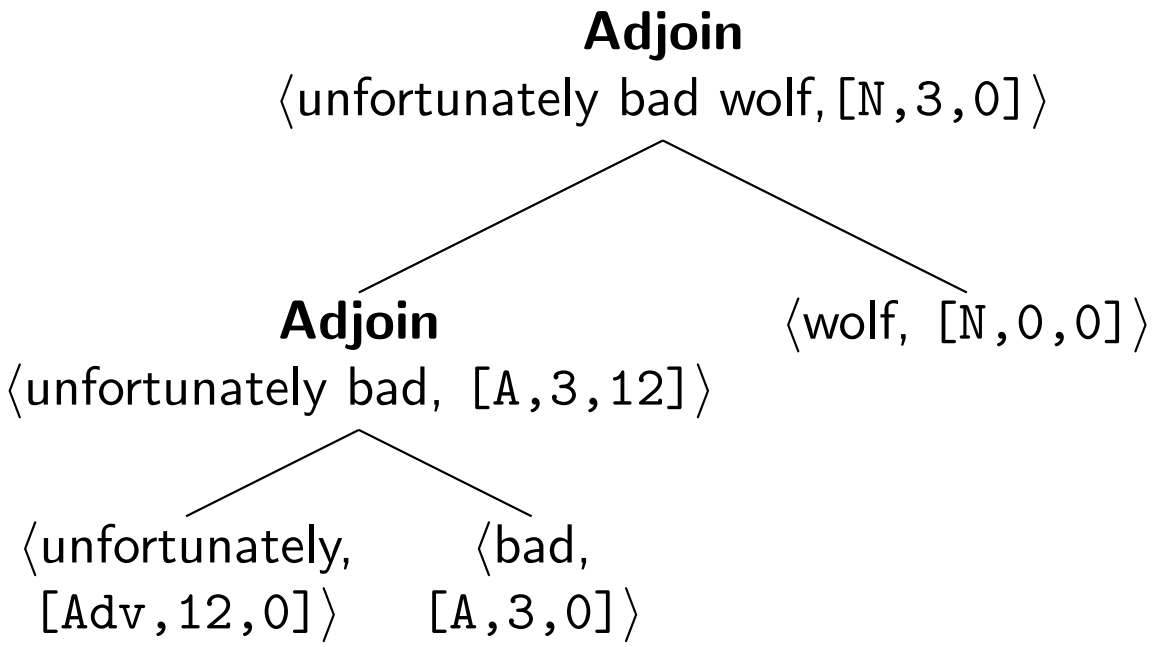
- ⟨bad, [A,3,0]⟩,
- ⟨big, [A,5,0]⟩,
- ⟨the, =N[D,0,0]⟩,
- ⟨wolf, [N,0,0]⟩,



Adjuncts of adjuncts

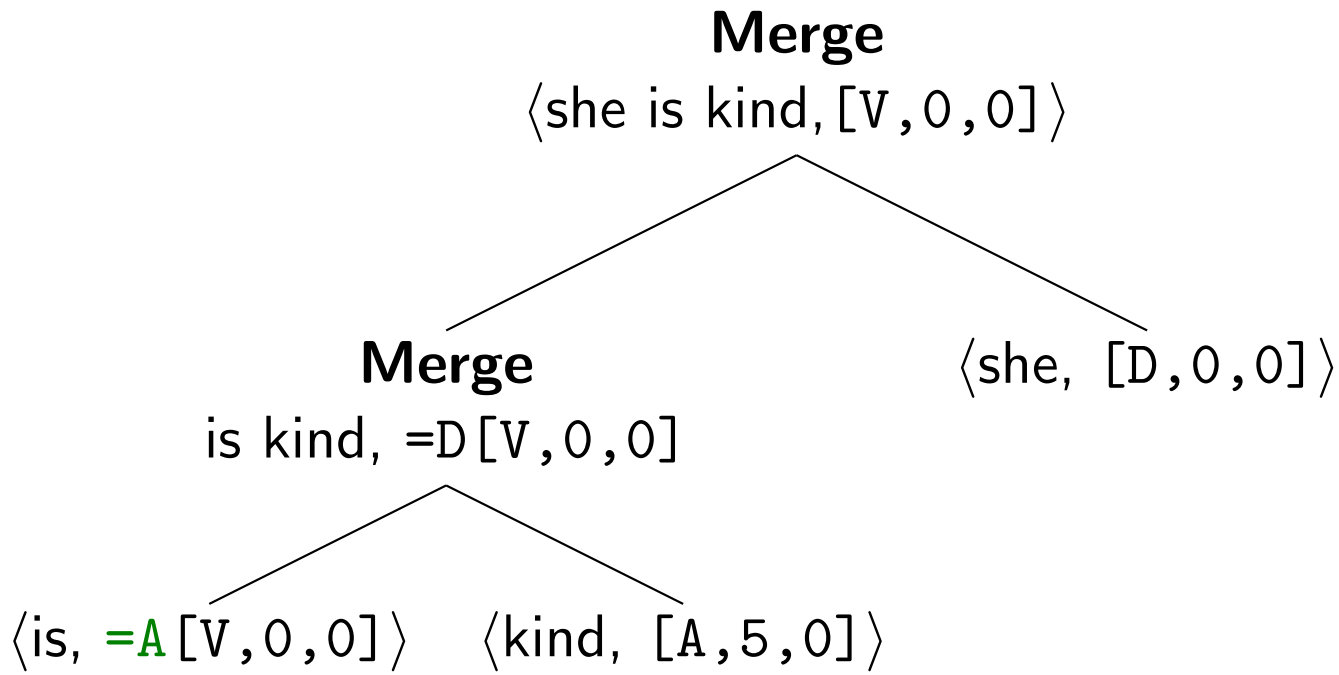
ad(N) = {A, P, C}
ad(V) = {Adv, P, C}
ad(A) = {Adv}

- ⟨frankly, [Adv, 12, 0]⟩
- ⟨unfortunately, [Adv, 11, 0]⟩
- ⟨allegedly, [Adv, 10, 0]⟩
- ⟨bad, [A, 3, 0]⟩
- ⟨wolf, [N, 0, 0]⟩



Selecting adjuncts

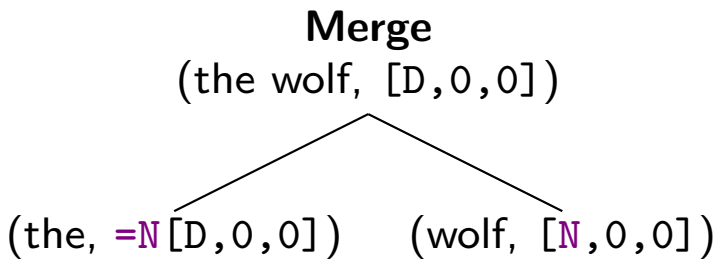
She is kind

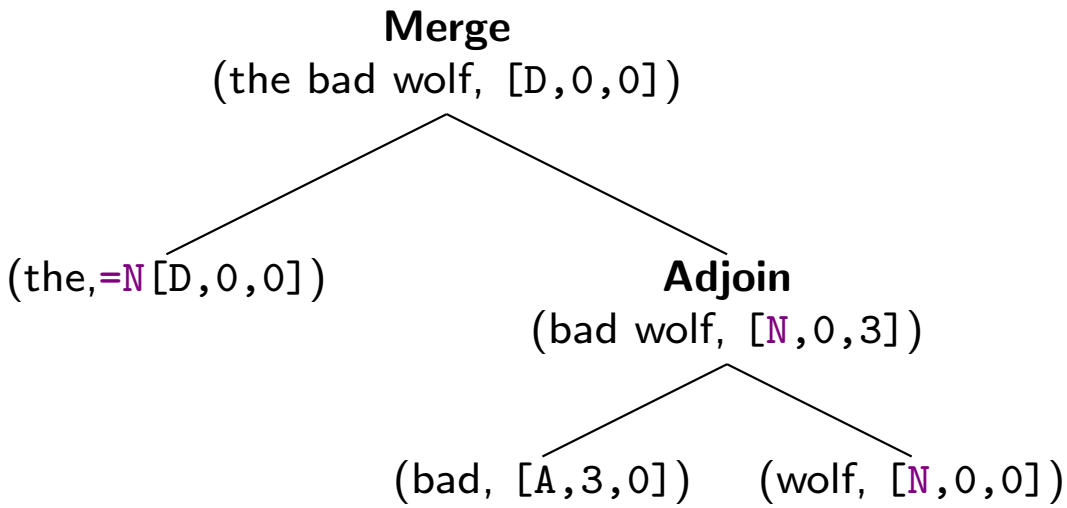


MGA – properties

Optional

Transparent to selection





Comparison

	Trad. (=XX)	Cart. (=A_5A_6)	MGAs ([X,i,j])
Optionality	✓	✗	✓
Selector selects expected category	✓	✗	✓
Adjunct does not become head	✗	✗	✓
Unordered adjuncts possible	✓	✗	(✓)
Ordered adjuncts possible	✗	✓	✓
Adjuncts of adjuncts	✗	✗	✓
Selectability	✗	✗	✓

MGAs and complexity

Graf (2014):
MGAs are measurably more complex than traditional MGs, but this complexity is necessary if we want to capture the real human language properties of adjunction

Conclusions

- Imagining you want to write a program – or better, actually writing one – helps clarify theories
- Minimalist Grammars are an intuitive formalism for minimalism
- Adding adjunction into minimalist grammars requires new machinery, but is relatively straightforward

References

Adger, David. 2008. A minimalist theory of feature structure. MSS, May 2008.

Chomsky, Noam. 1995. *The minimalist program*. Cambridge, MA: MIT Press.

Cinque, Gugliemo. 1999. *Adverbs and functional heads: a cross-linguistic perspective*. Oxford studies in comparative syntax. Oxford: Oxford University Press.

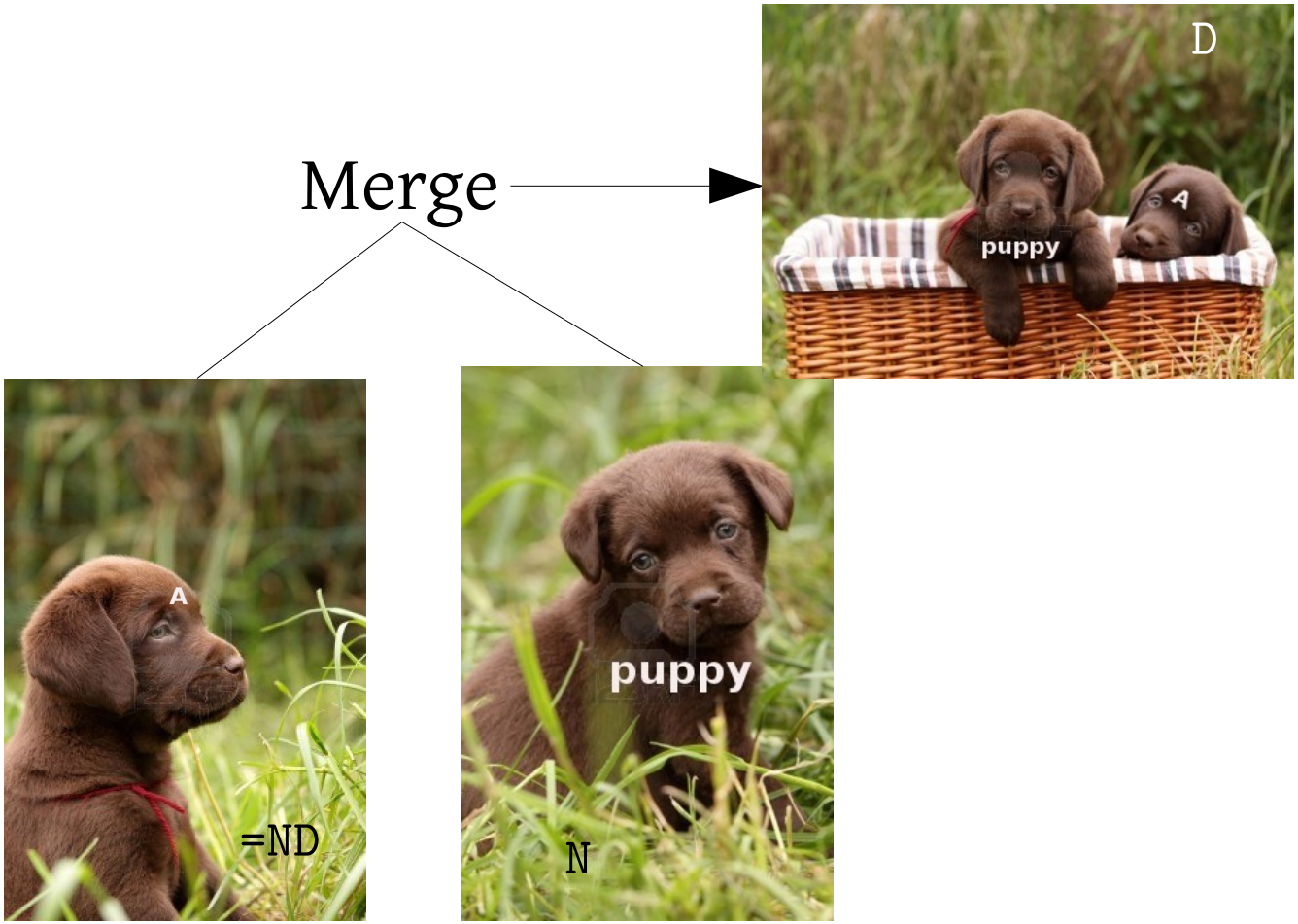
Fowlie, Meaghan. 2014. Adjuncts and minimalist grammars. In *Proceedings of Formal Grammar 2014*, ed. Glyn Morrill, Reinhard Muskens, Rainer Osswald, and Frank Richter, volume 8612 of *Lecture Notes in Computer Science*.

Graf, Thomas. 2014. Models of adjunction in minimalist grammars. In *Formal Grammar*, 52–68. Springer.

Sportiche, Dominique. 2005. Division of labor between merge and move: Strict locality of selection and apparent reconstruction paradoxes. *LingBuzz* .

Stabler, Edward. 1997. Derivational minimalism. *Logical Aspects of Computational Linguistics* 68–95.

Thank you!



Cartography: default adjunct orders

- Using the same architecture, how can we capture ordering?

(5) a. Wear the enormous ugly green hat
Wear the hat that is enormous, ugly, and green
b. #Wear the ugly enormous green hat
Of your enormous green hats, wear the ugly one.

Cartography (Cinque, 1999)

(6)

a. The small ancient triangular green Irish pagan metal artifact was lost.

b. *The metal green small artifact was lost. **Adjectives**

c. Frankly, John probably once usually arrived early.

d. *Usually, John early frankly once arrived probably. **Adverbs**

e. [Il premio Nobel]$_{\text{top}}$, [a chi]$_{\text{wh}}$ lo daranno?
[the prize Nobel]$_{\text{top}}$, [to whom]$_{\text{wh}}$ it give.fut
'The Nobel Prize, to whom will they give it?' **Left periphery**

f. DP zhe [NumP yi [ClP zhi [NP bi]]]
DP this [NumP one [ClP Cl [NP pen]]]
'this pen' **Functional DP projections**

Adjoin: formal definition

Definition (Adjoin)

Let $s, t \in \Sigma$ be strings, $\mathtt{Y}, \mathtt{X} \in \mathbf{sel}$ be categories, $i, j, n, m \in \mathbb{N}$, $mvrs \in (\Sigma^* \times F)^*$ be a mover list, and $\alpha, \beta \in F^*$.

$$\mathbf{Adjoin}(\langle s, [\mathtt{X}, \mathtt{i}, \mathtt{j}]\alpha :: mvrs\rangle, \langle t, [\mathtt{Y}, \mathtt{n}, \mathtt{m}]\beta\rangle) =$$
$$\begin{cases} \langle ts, [\mathtt{X}, \mathtt{i}, \mathtt{n}]\alpha\rangle :: mvrs & \text{if } \mathtt{n} \geq \mathtt{j} \ \&\ \mathtt{Y} \in \mathbf{Ad}(\mathtt{X}) \ \&\ \beta = \epsilon \\ \langle s, [\mathtt{X}, \mathtt{i}, \mathtt{n}]\alpha\rangle :: \langle t, \beta\rangle :: mvrs & \text{if } \mathtt{n} \geq \mathtt{j} \ \&\ \mathtt{Y} \in \mathbf{Ad}(\mathtt{X}) \ \&\ \beta \neq \epsilon \end{cases}$$

Merge: new formal definition

Definition (Merge)

For $\alpha, \beta \in F^*$; s, t strings:

Merge$(\langle s, =\mathtt{X}\alpha\rangle ::\text{mvrs}_s, \langle t, [\mathtt{X}, \mathtt{i}, \mathtt{j}]\beta\rangle ::\text{mvrs}_t) =$

$$\begin{cases} \langle st, \alpha\rangle :: \text{mvrs}_s \cdot \text{mvrs}_t & \text{if } \beta = \epsilon \\ \langle s, \alpha\rangle :: \langle t, \beta\rangle :: \text{mvrs}_s \cdot \text{mvrs}_t & \text{if } \beta \neq \epsilon \end{cases}$$

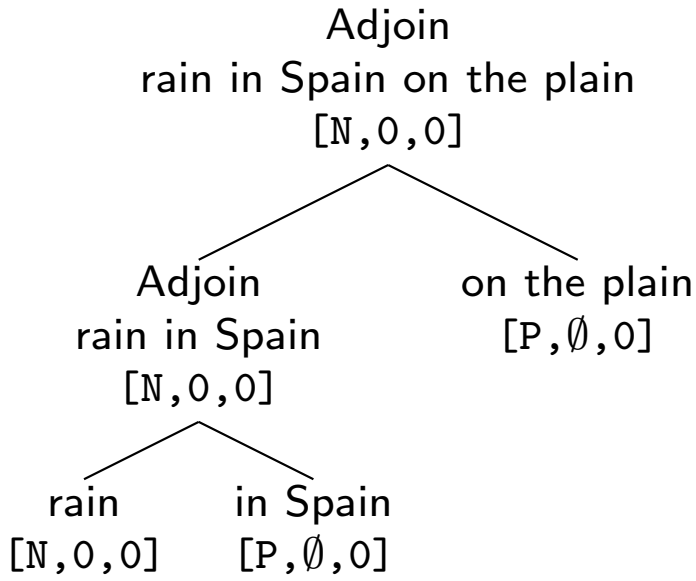
Unordered adjuncts

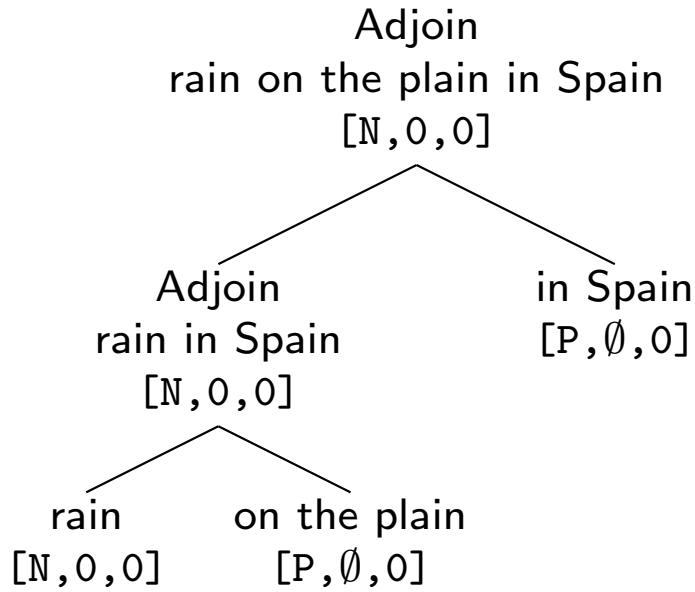
? Unordered

- Could make them all one level
- Or at every level
- Better: expand index set to include non-number, $\emptyset$
  - When **Adjoin** sees $\emptyset$, *asymmetrically* checks features
  - Hierarchy level of phrase doesn't change

Unordered adjuncts

✓**Unordered**





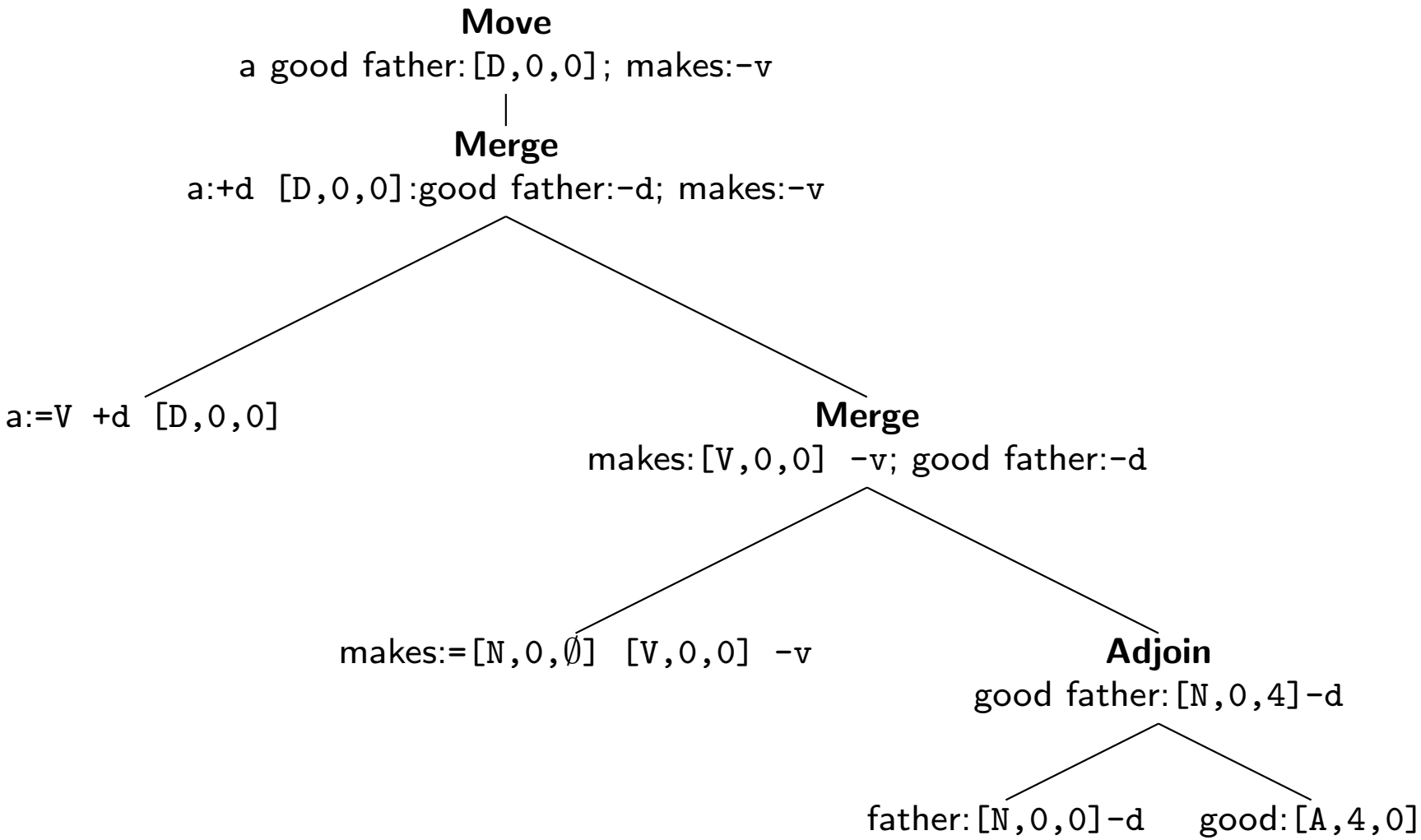
✓**Adjuncts on either side of head**

Obligatory adjuncts

(7) a. He makes a **good** father.
b. *He makes a father.

- Noun with no adjuncts: `[N,0,0]`
- Noun with adjunct: `[N,0,3]`
- →Expand Merge to require last element to be non-zero
- =`[N,0,∅]` can Merge with `[N,i,j]` for `j>0`

Obligatory adjuncts



Note: Sportiche (2005) proposes that verbs select NPs, and the NPs move to their Ds, which are functional heads on the spine.

Minimalist Grammars with Hierarchies

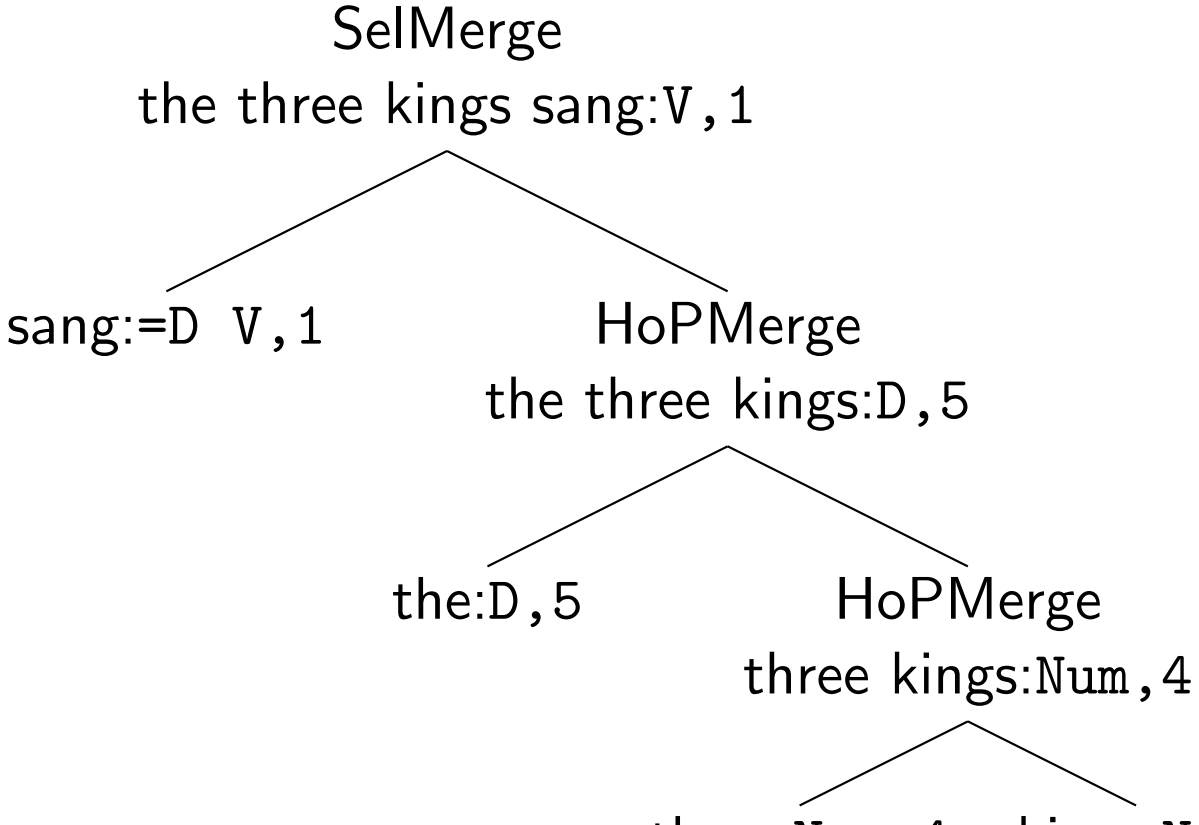
Adger (2008): Hierarchy of Precedence Merge

textbfLex = {the:=D V,0, three:Num,4, kings:N,0, sang:=D V,0}

$H_1 = \texttt{D}, 5 > \texttt{Num}, 4 > \texttt{Poss}, 3 > \texttt{n}, 2 > \texttt{N}, 1$

$H_2 = \texttt{C}, 3 > \texttt{T}, 2 > \texttt{V}, 1$

We can derive *the three kings sang* as in Figure **??**.



Minimalist Grammars with Hierarchies