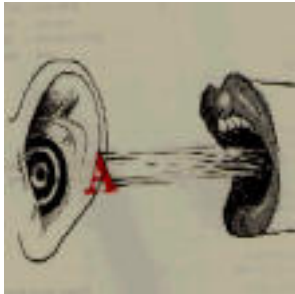


Lecture 9: Grammars & features; lexicalized parsing; treebanks



Professor Robert C. Berwick
berwick@csail.mit.edu

The Menu Bar

- Administrivia: Lab 3 due Friday; Lab 4 out today
- Why context-free grammars are no good: representation and computational issues
- Slash & Burn grammar: mind the gap!
- Featuritis!
- What's the matter with Probabilistic context-free kansas?
- How to fix – a bit – with *lexicalized* context-free grammars; treebank parsing

The Menu Bar

- Why context-free grammars are no good: representation and computational issues
- Slash & Burn grammar: mind the gap
- Featuritis!
- WHY do we choose certain rules and not others?
- What's the matter with Probabilistic context-free kansas?
- How to fix – a bit – with *lexicalized* context-free grammars; treebank parsing

‘Empty’ elements or categories

Very often, a phrase is displaced from its canonical syntactic position & *nothing* shows on the surface:

- Examples:

The ice-cream was eaten vs.

John ate the ice-cream

What did John eat?

What did Bill say that that John thought the cat ate?

(= For What x , did Bill say... the cat ate x)

Quadaffi is too stubborn to talk to =

Quadaffi is too stubborn [x to talk to Quadaffi]

Quadaffi is too stubborn to talk to the Pope =

Quadaffi is too stubborn [Quadaffi to talk to the Pope]

‘Missing’ or “empty” categories

- *Quadaffi promised Obama ___ to leave*
- *Quadaffi promised Obama [Quadaffi to leave]*
- Known as ‘control’

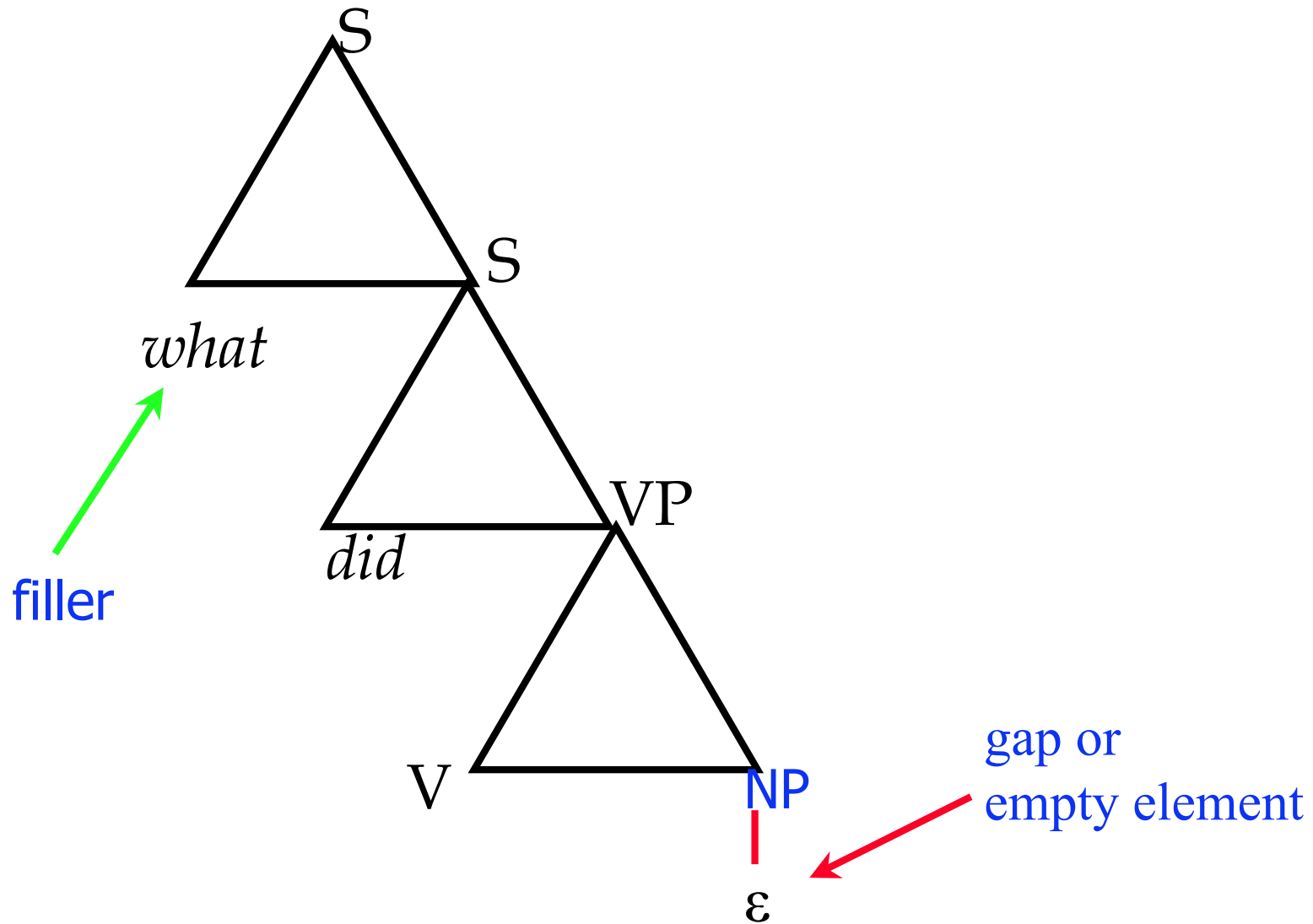
- *Obama persuaded Quadaffi [___ to leave]*
- *Obama persuaded Quadaffi [Quadaffi to leave]*

Representation & computation questions again

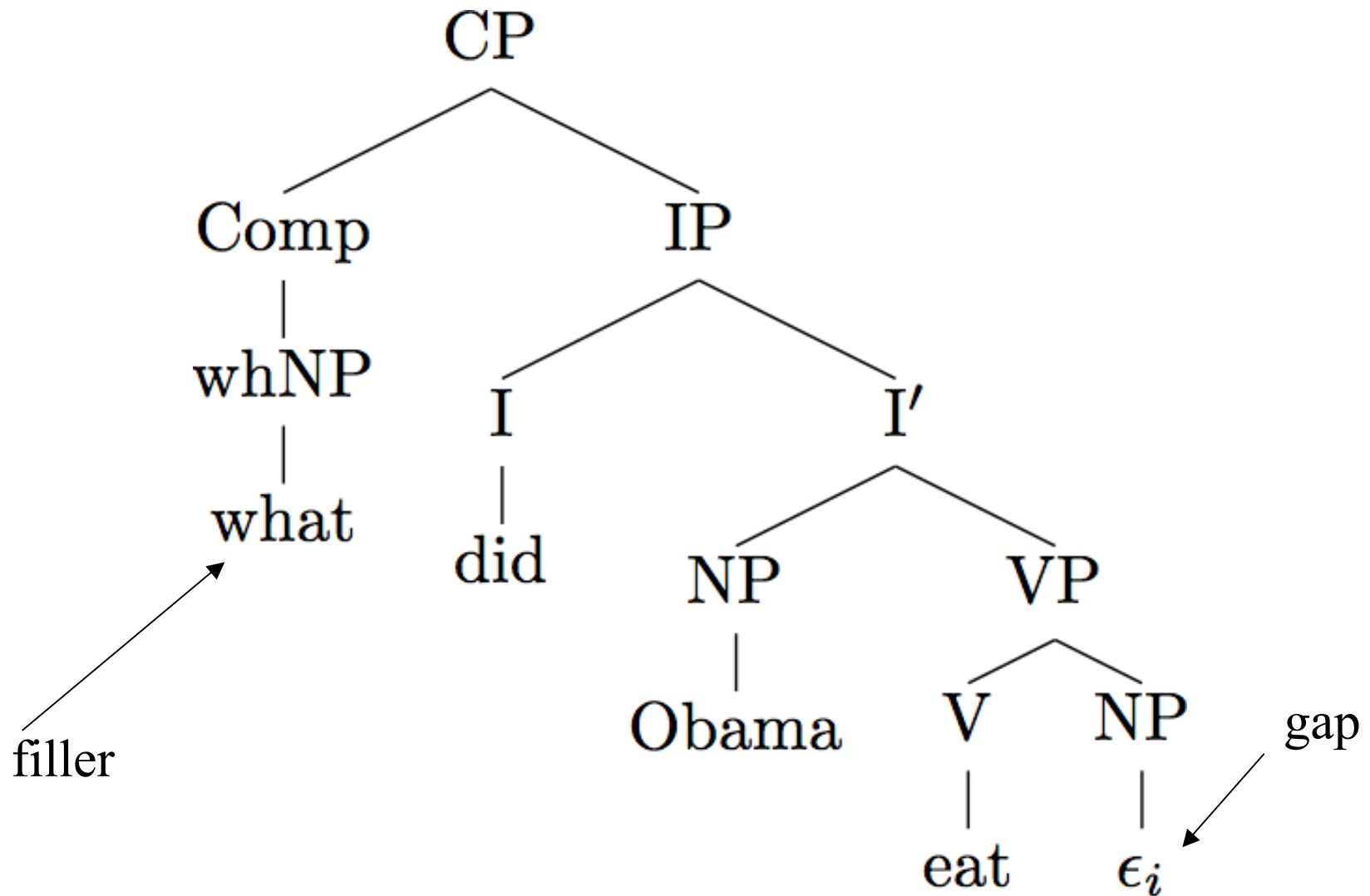
- How do we *represent* this displacement? (difference between underlying & surface forms)
- How do we *compute* it? (I.e., parse sentences that exhibit it)
- We want to recover the *underlying* structural relationship because this tells us what the predicate-argument relations are – *Who did what to whom*
- Example: *What did John eat* → For which x , x a thing, did John eat x ?
- Note how the eat- x predicate-argument is established

Representations with gaps

- Let's first look at a tree with gaps:



Slightly crisper representation



Fillers can be arbitrarily far from gaps
they match with...

- *What did John say that Mary thought that the cat ate _____?*

In short, many types of similar examples...

- *Wh*-questions:

What did you find

Tell me who you talked to

- Topicalization:

This manual, I can't find

Morris, you should talk to

- *Easy* adjectives:

No Stata office is easy to find

This guy is hard to talk to

This problem everyone thought was too easy

But there are constraints on being ‘too far’ away...

- *?John seems it is certain [x to like ice-cream]*
- *Which man did you ask whether I saw at the park?*

Fillers and gaps

- Since ‘gap’ is NP going to empty string, we could just add rule, $NP \rightarrow \epsilon$
- But this will *overgenerate* Why?
- We need a way to distinguish between:
 - *What did John eat*
 - *Did John eat*

So, what do we need?

- A rule to expand NP as the empty symbol; that's easy enough: $NP \rightarrow \epsilon$
- A way to make sure that NP is expanded as empty symbol iff there is a gap (in the right place) before/after it
- A way to link the filler and the gap
- We can do all this by futzing with the nonterminal names: **Generalized Phrase Structure Grammar (GPSG)** (more recently known as: **Head-driven Phrase structure grammar, or HPSG**)

Example: relative clauses

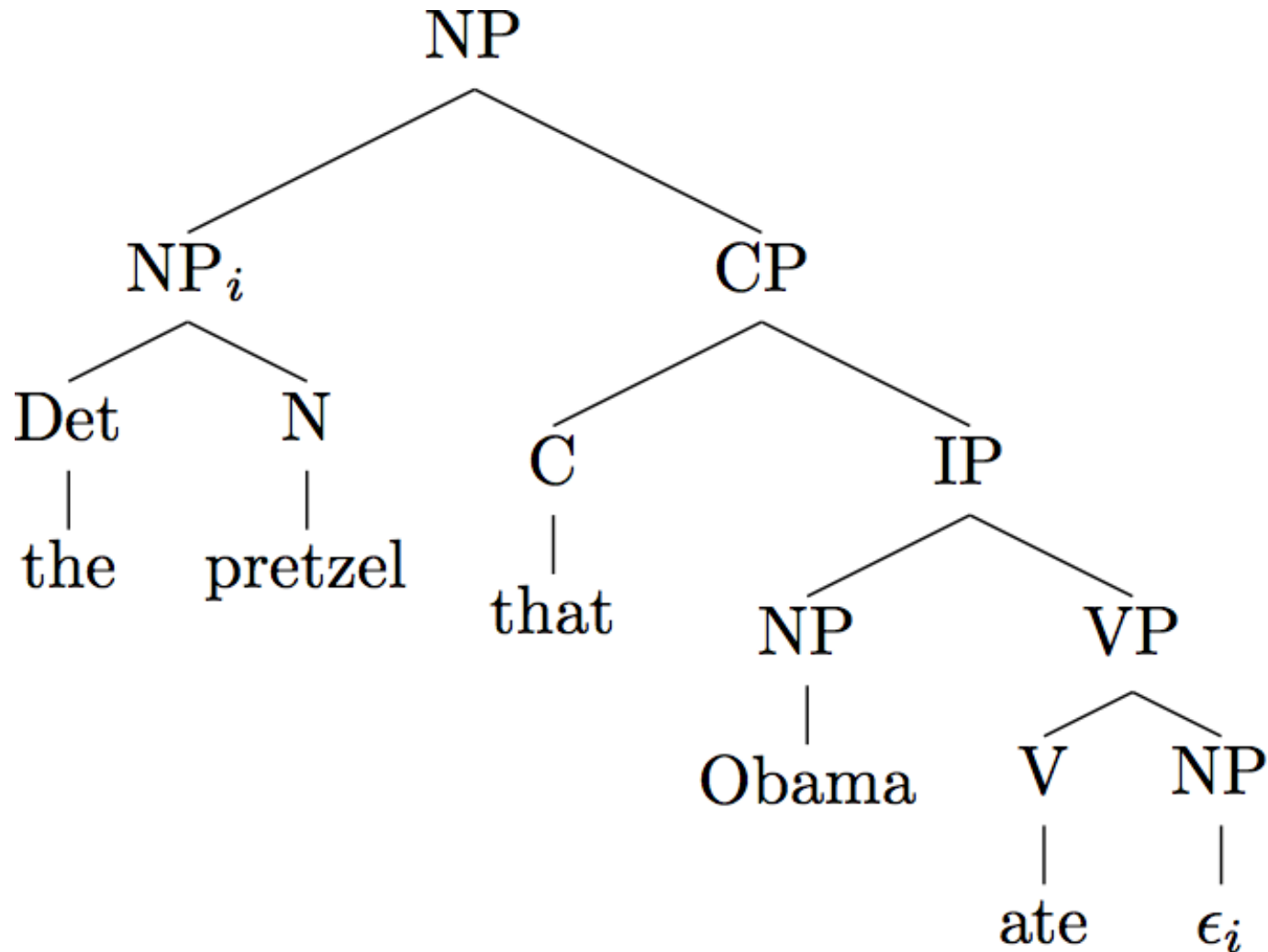
- What are they?
- Noun phrase with a sentence embedded in it:
 - *The pretzel that Obama ate*
- What about it? What's the syntactic representation that will make the semantics *transparent*?

The pretzel_i that Obama ate ϵ_i

OK, that's the output...what are the context-free grammar rules?

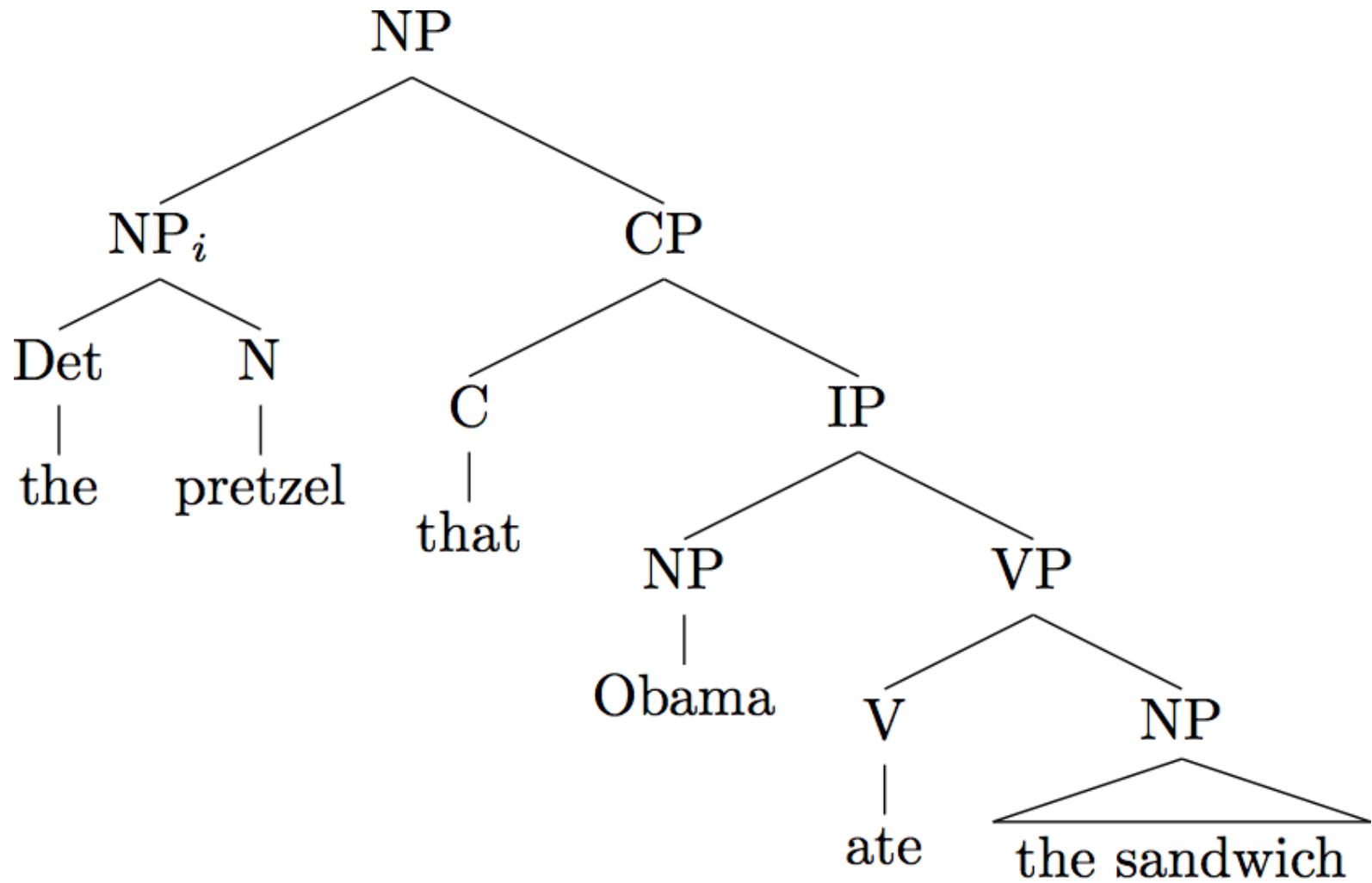
- Need to expand the object of *eat* as an empty string
- So, need rule $NP \rightarrow \epsilon$
- But more, we need to link the head noun “the pretzel” to this position

Parse structure for relative clause



But how to generate this and yet block this:

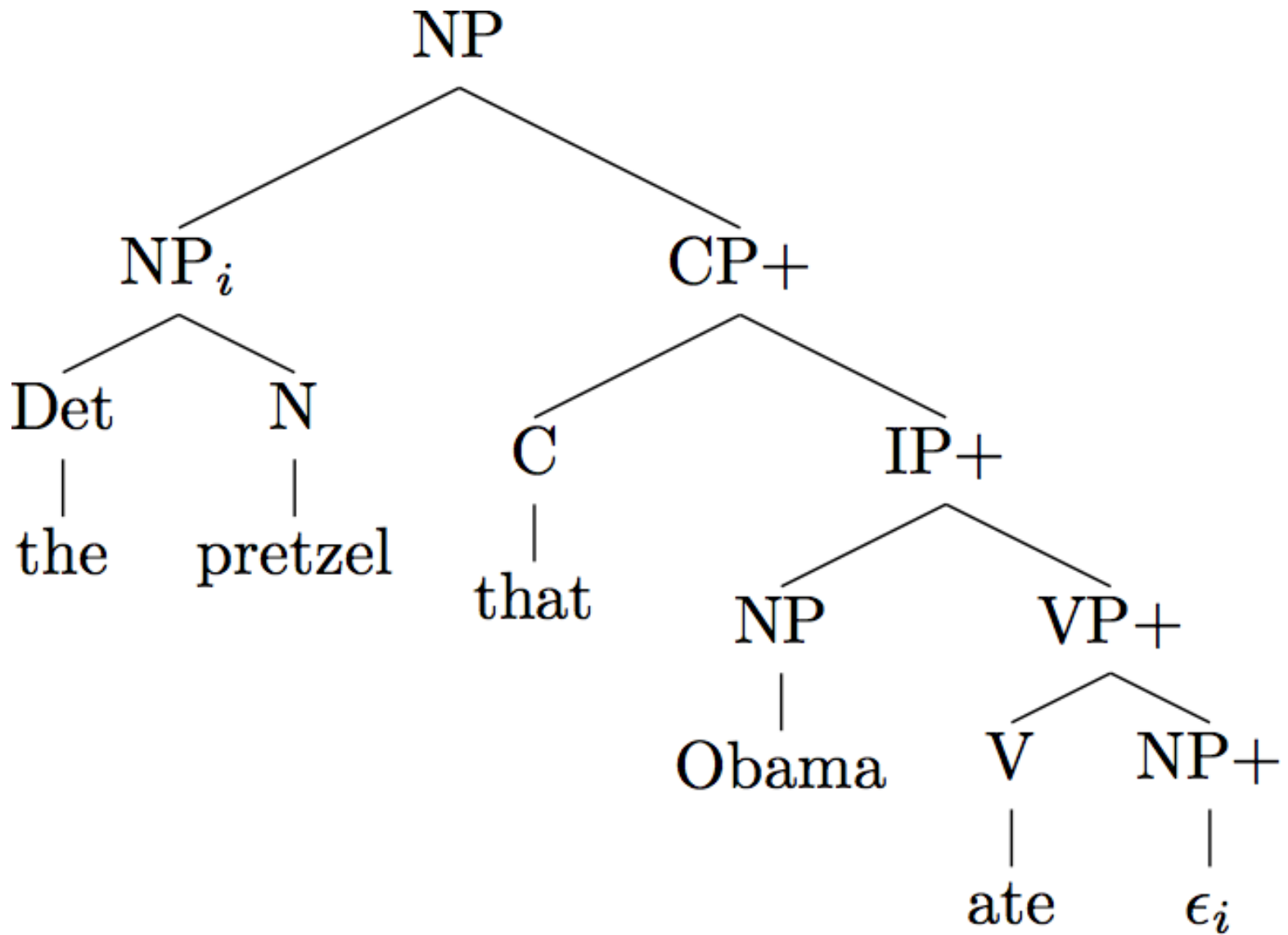
Not OK!



In short..

- We can expand out to ε iff there is a prior NP we want to link to
- So, we need some way of ‘marking’ this in the state – i.e., the nonterminal
- Further, we have to somehow co-index ε and ‘the sandwich’
- Well: let’s use a mark, say, “+”

The mark...



But we can add + except this way:

- Add as part of atomic nonterminal name

- Before: $NP \rightarrow NP\ CP$

$CP \rightarrow \text{Comp } S$

$IP \rightarrow NP\ VP$

$VP \rightarrow VP\ NP$

- After: $NP \rightarrow NP\ CP^+$

$CP^+ \rightarrow \text{Comp } S^+$

$IP^+ \rightarrow NP\ VP^+$

$VP^+ \rightarrow V\ NP^+$

$NP^+ \rightarrow \epsilon$

Why does this work?

- Has desired effect of blocking ‘the pretzel that Bush ate the sandwich’
- Has desired effect of allowing ϵ exactly when there is no other object
- Has desired effect of ‘linking’ pretzel to the object (how?)
- Also: desired configuration between filler and gap (what is this?)

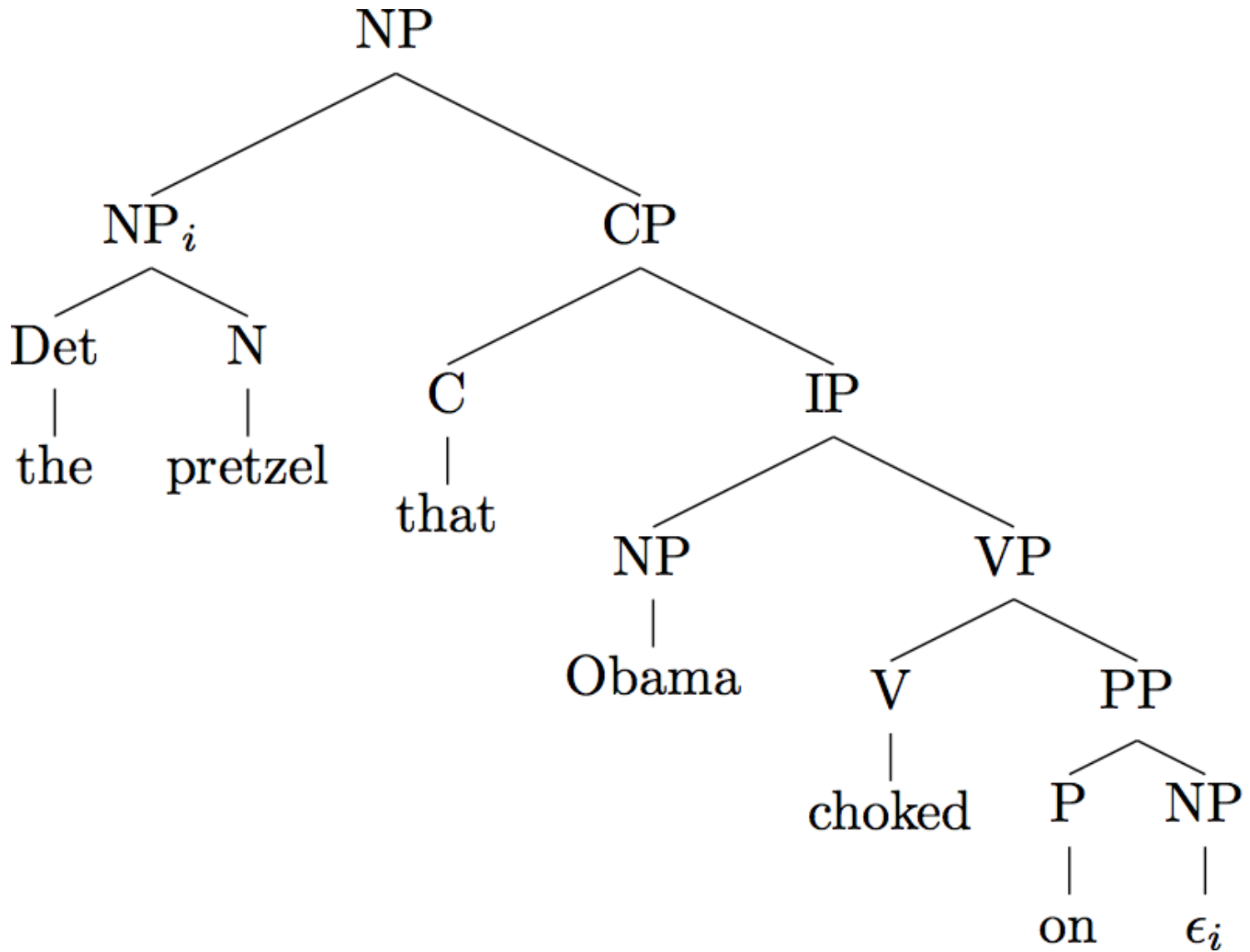
Actual ‘marks’ in the literature

- Called a ‘slash category’
- Ordinary category: CP, VP, NP
- Slash category: CP/NP, VP/NP, NP/NP
- “X/Y” is ONE atomic nonterminal name
- Interpret as: Subtree X is missing a Y (expanded as ϵ) underneath
- Example: CP/NP = CP missing NP underneath (see our example)

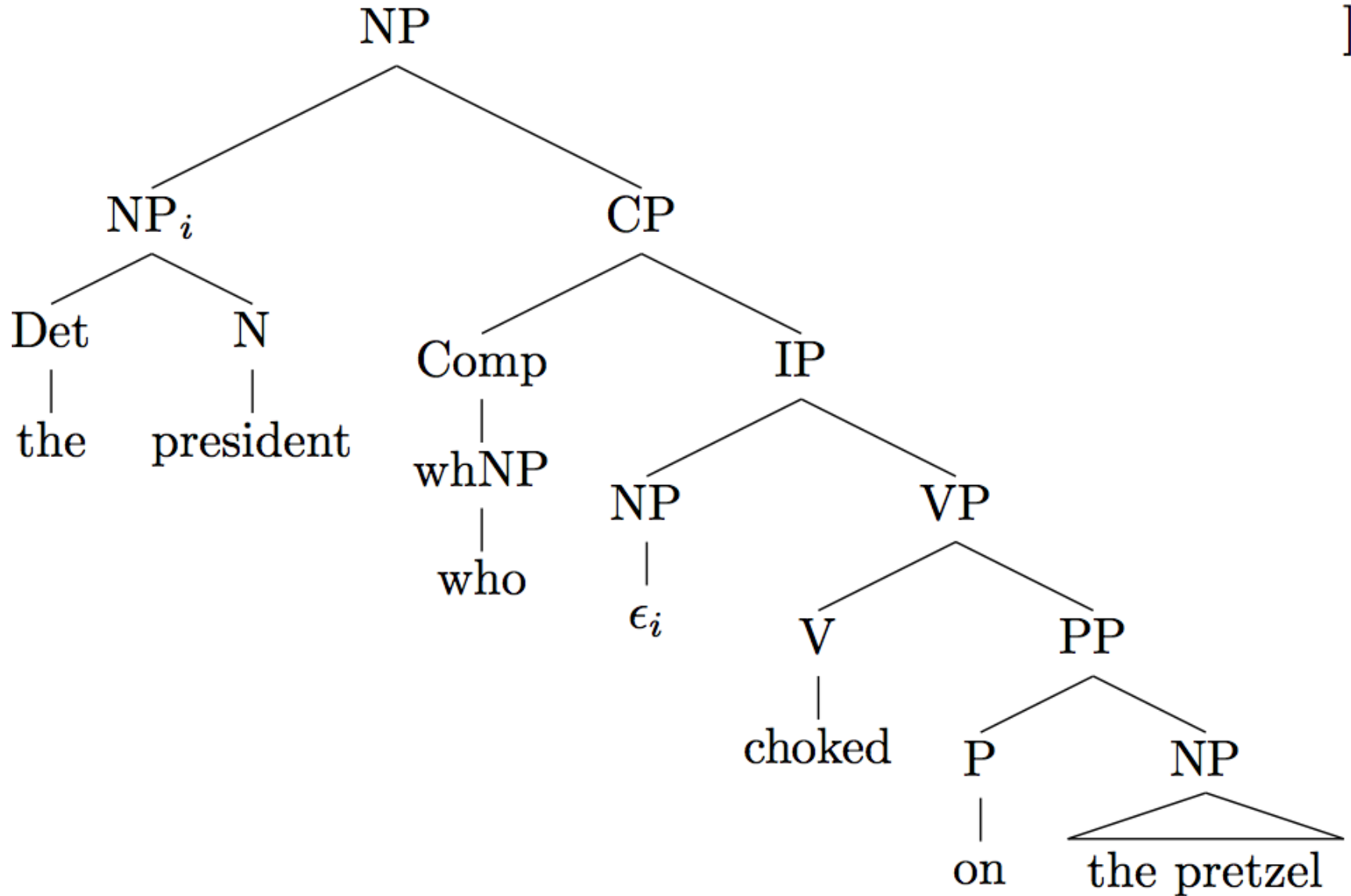
As for slash rules...

- We need slash category introduction rule, e.g.,
 $CP \rightarrow \text{Comp } IP/NP$
- We need ‘elimination’ rule $NP/NP \rightarrow \varepsilon$
- These are paired (why?)
- We’ll need other slash categories, e.g.,

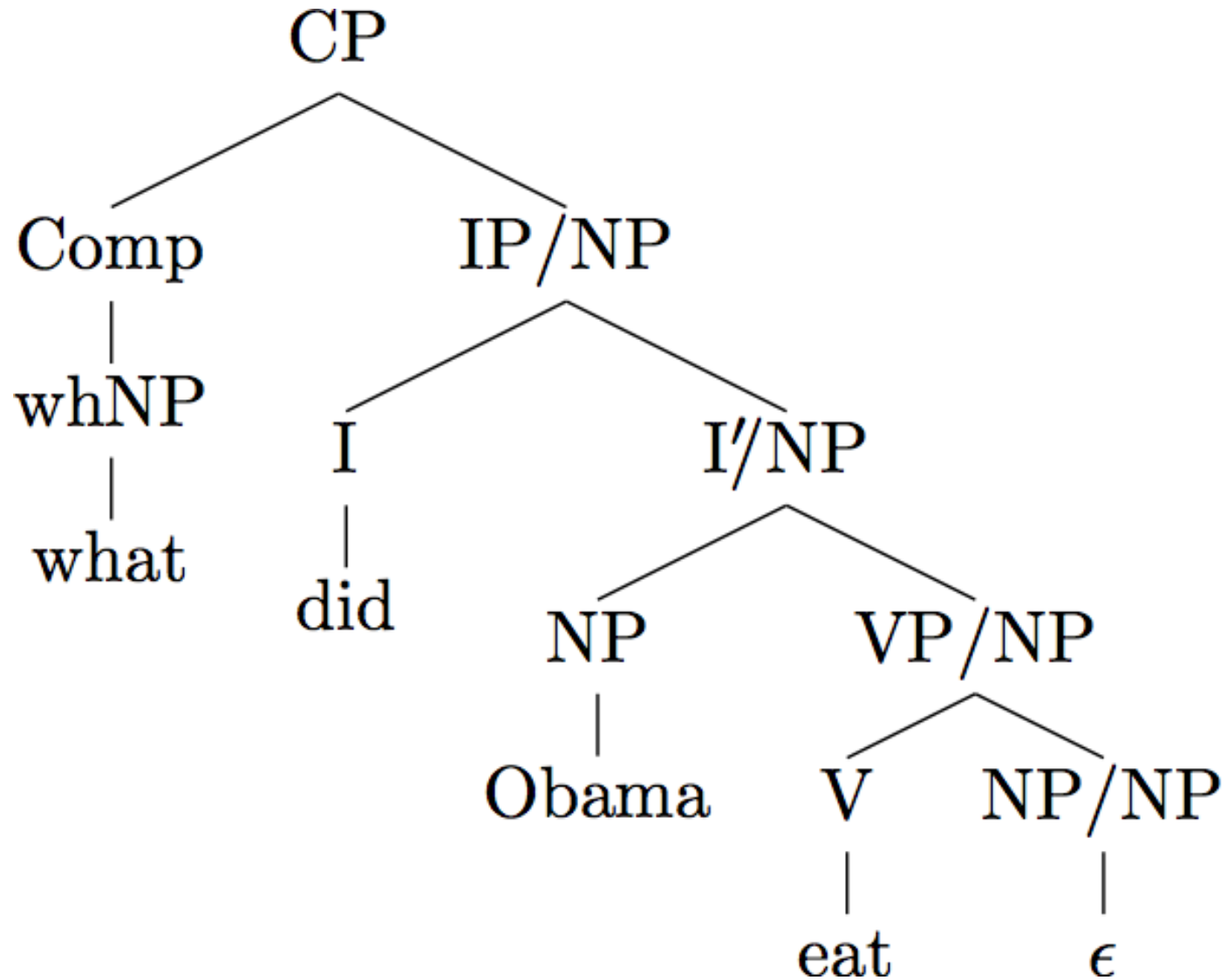
Need PP/NP...



Can also have 'subject' gaps

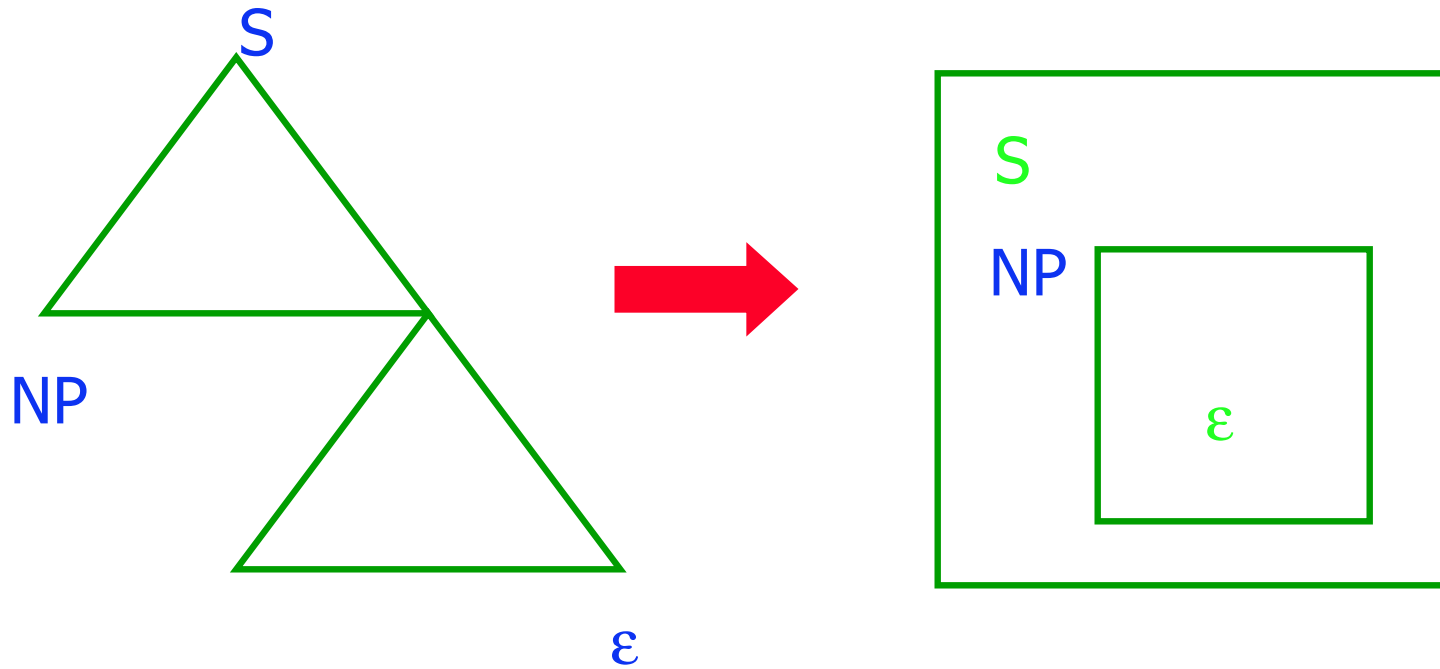


Also needed for questions



How would we write this if we were
Gerry Sussman?

Filler-gap configuration



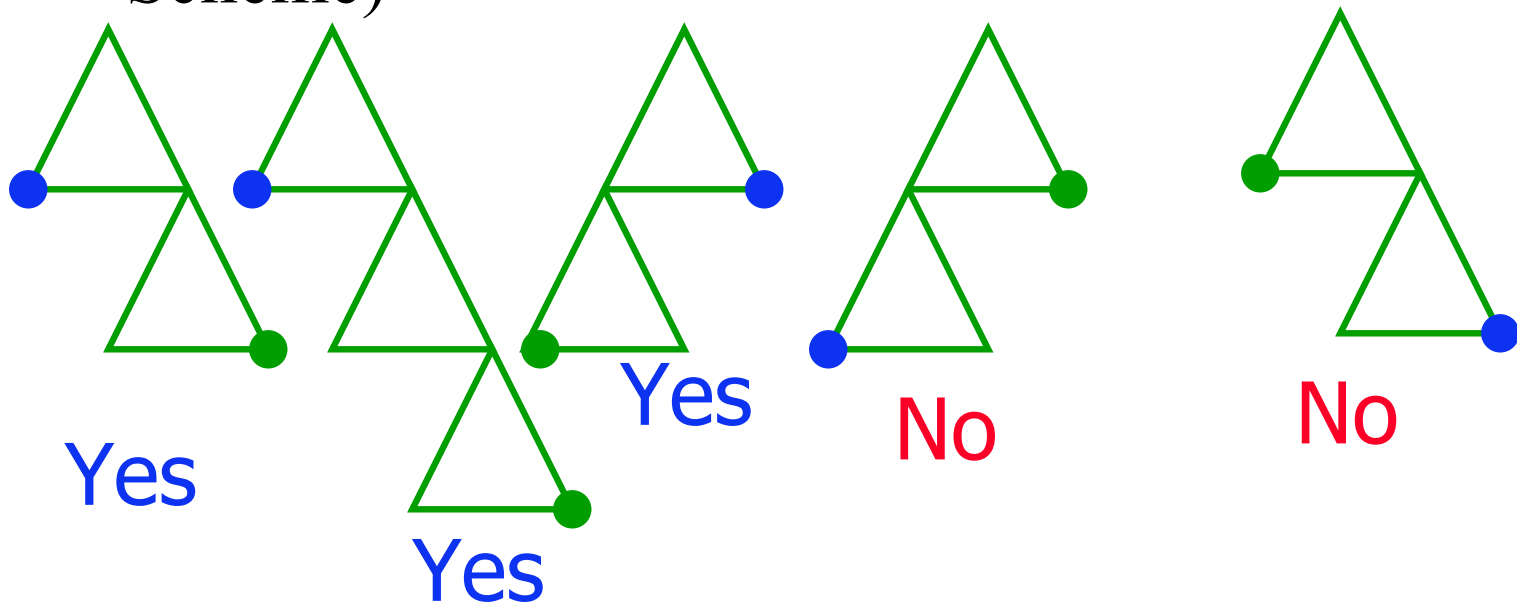
Filler-gap configuration

Is equivalent to the notion of ‘scope’ for natural languages (scope of variables) \approx Environment frame in Scheme/binding environment for ‘variables’ that are empty categories

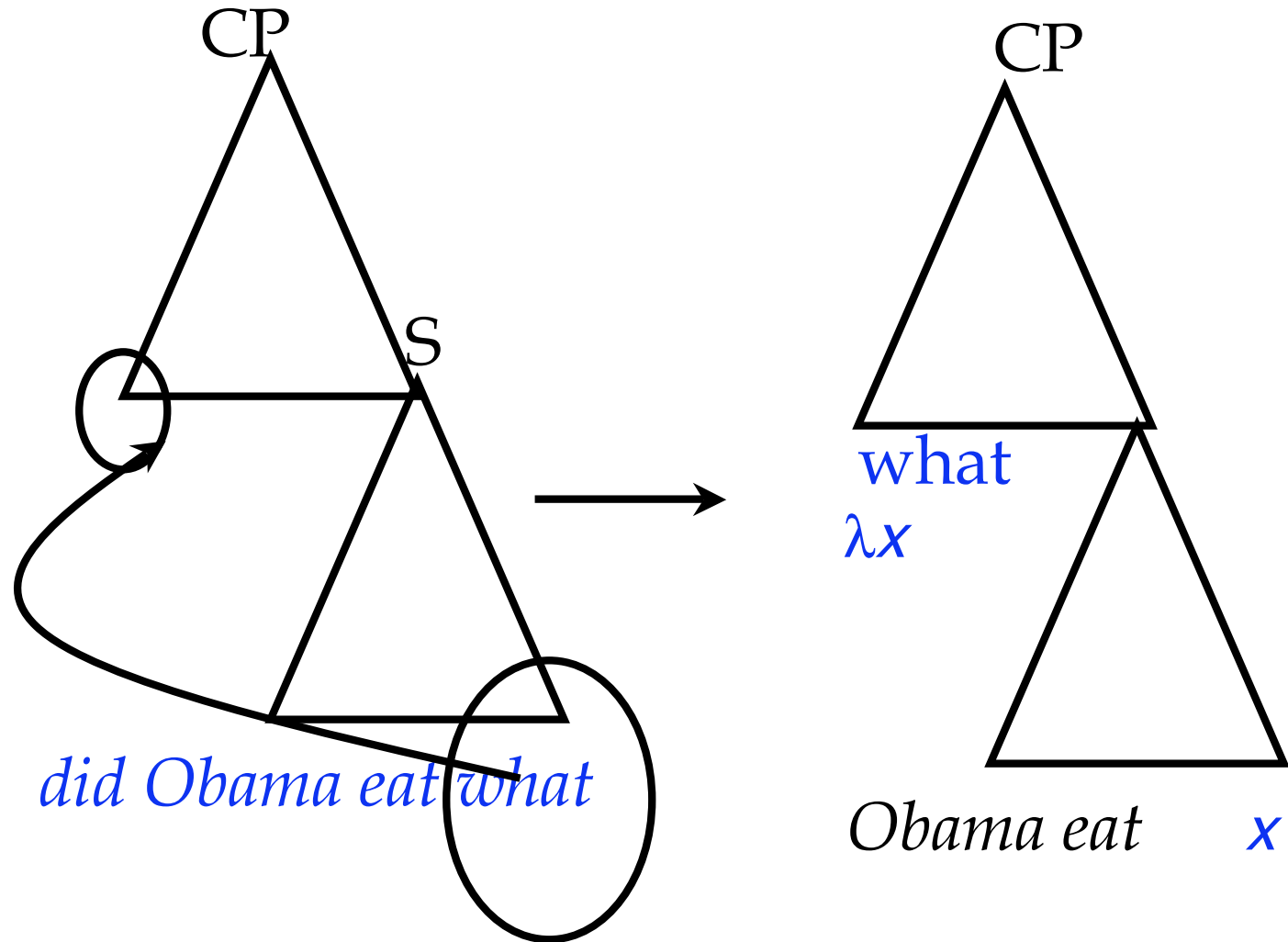
- Formally: Fillers **c-command** gaps (constituent command)
- Definition of c-command:

C-command

- A phrase α **c-commands** a phrase β iff the first branching node that dominates α also dominates β (blue = filler, green = gap)
- We will see this is critical for semantics! (as in Scheme)



Natural for λ abstraction

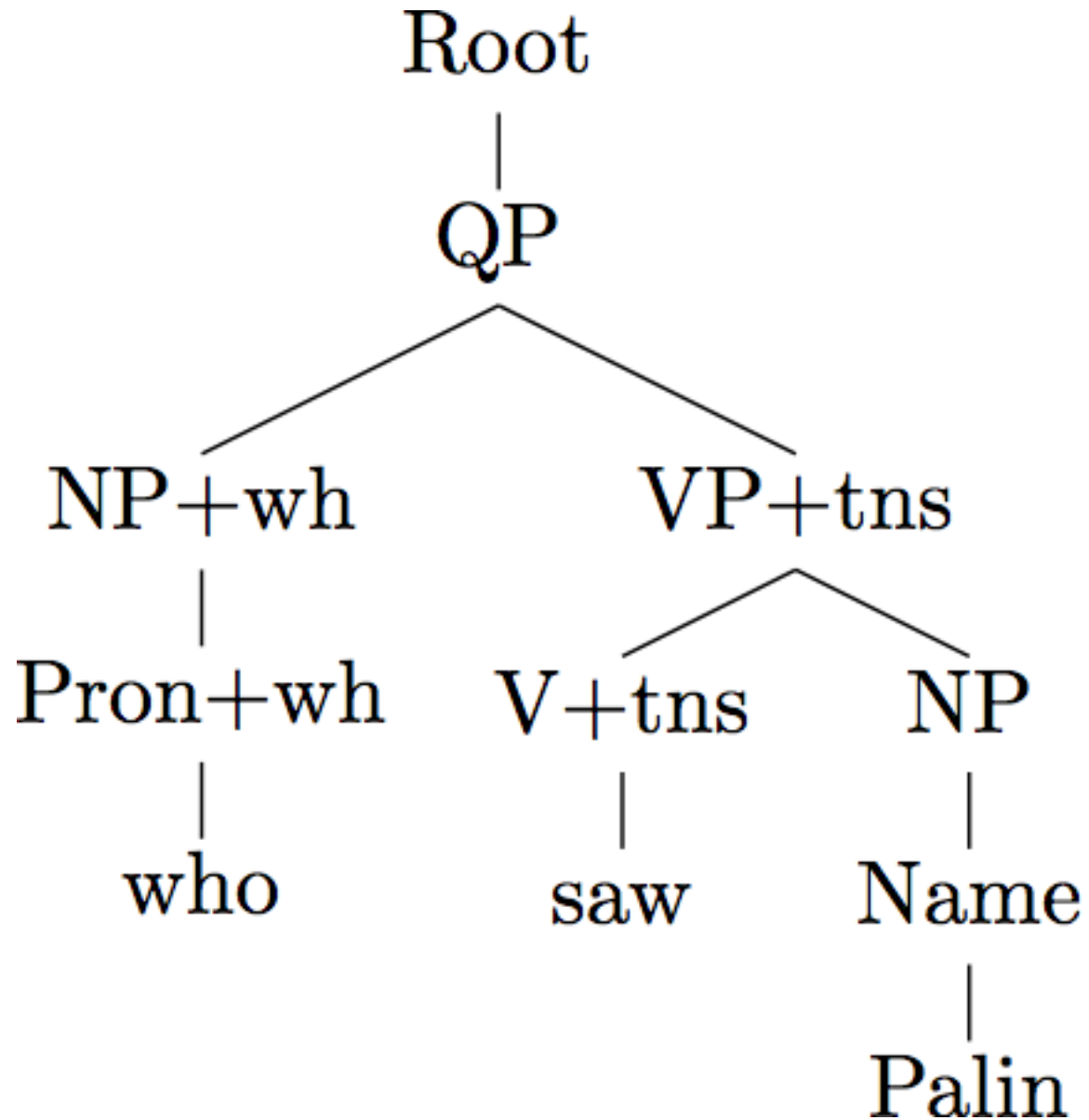


A Puzzle

- *Who saw Palin?*
McCain

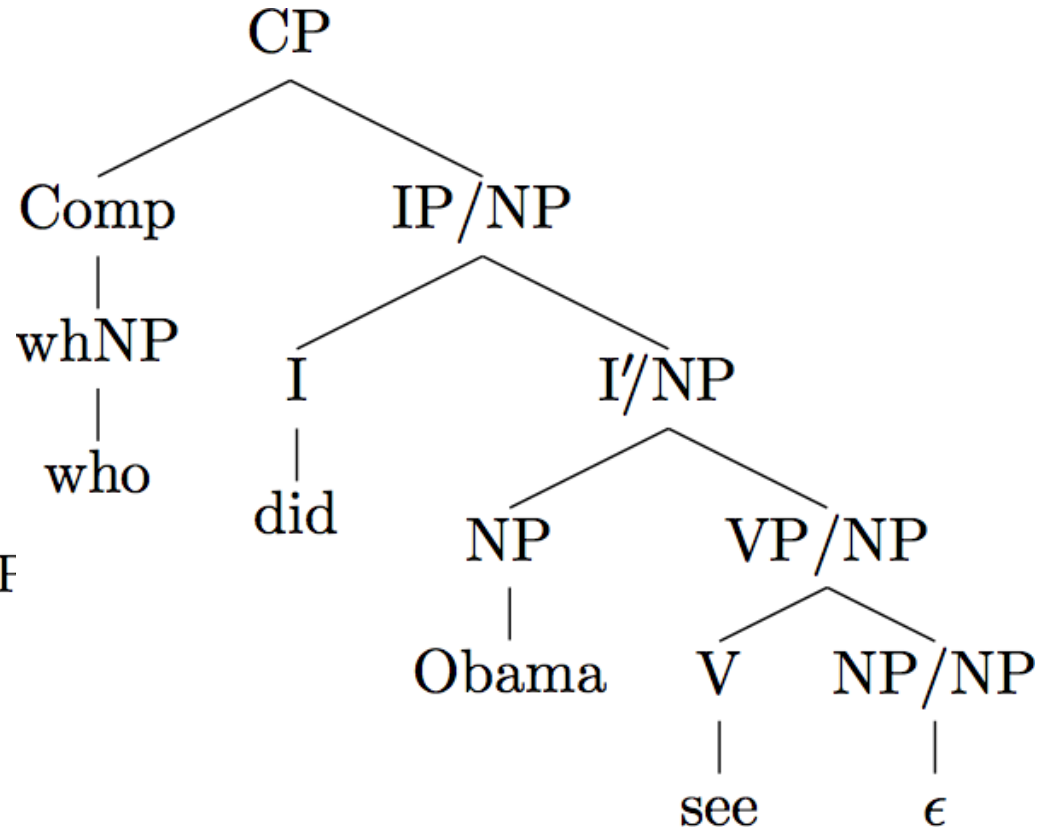
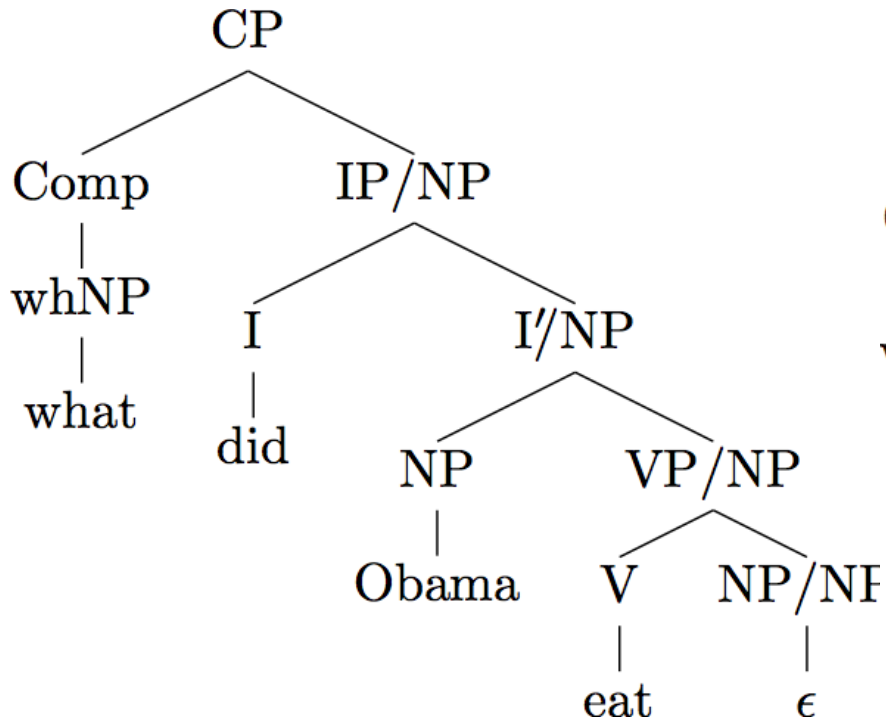
What should the structure of the first sentence be?

Idea 1: WYSIG syntax



Is this right?
Why might it not be correct?

Idea 2: make syntax same as with ‘ordinary’ questions



What are the benefits of this approach?

But language is more complex yet...

TO WHICH CITY AND WHICH CONFERENCE DID BILL GO TO
WHICH CITY AND TO WHICH CONFERENCE DID BILL GO TO
WHICH CITY AND WHICH CONFERENCE DID BILL GO TO TO
WHICH CITY AND WHICH CONFERENCE DID BILL GO TO
TO WHICH CITY AND WHICH CONFERENCE DID BILL GO
TO WHICH CITY AND TO WHICH CONFERENCE DID BILL GO
ON WHICH TABLE AND UNDER WHICH FLOWER POT DID MARY PUT THE KEYS
WHERE AND WHEN DID BILL PUT THE BOOK
WHICH BOOK AND WHICH PENCIL DID MARY BUY
MARY ASKED WHO AND WHAT BOUGHT
MARY ASKED WHO AND WHERE BILL HAD SEEN
I WONDER WHO MARY LIKES AND HOPES WILL WIN
I WONDER WHO BILL SAW AND LIKED MARY
I WONDER WHO SAW BILL AND LIKED MARY
I WONDER WHO BILL SAW AND MARY LIKED

JOHN GAVE MARY AND BILL TO FRED BOOKS THAT LOOKED
REMARKABLY SIMILAR

JOHN HUMMED AND MARY SANG AT EQUAL VOLUMES

JOHN HUMMED AND MARY SANG THE SAME TUNE

I HAVE BEEN WONDERING WHETHER BUT WOULD NOT WANT TO
STATE THAT YOUR THEORY IS CORRECT

I CAN TELL YOU WHEN BUT I CAN NOT TELL YOU WHY BILL LEFT ME

I LIKE BUT BILL DOES NOT LIKE TO VISIT NEW CITIES

JOHN HAS CLAIMED BUT I DO NOT BELIEVE THAT BILL IS A COMMUNIST

MARY USED TO BE AND BILL STILL IS VERY SUSPICIOUS

MARY SAID SHE WOULD AND BILL ACTUALLY DID EAT A RAW EGGPLANT

BILL MAY BE AND JOHN CERTAINLY IS A WEREWOLF

BILL OFFERED AND MARY GAVE JOHN A CADILLAC

BILL CAUGHT AND MARY KILLED THE RABID DOG

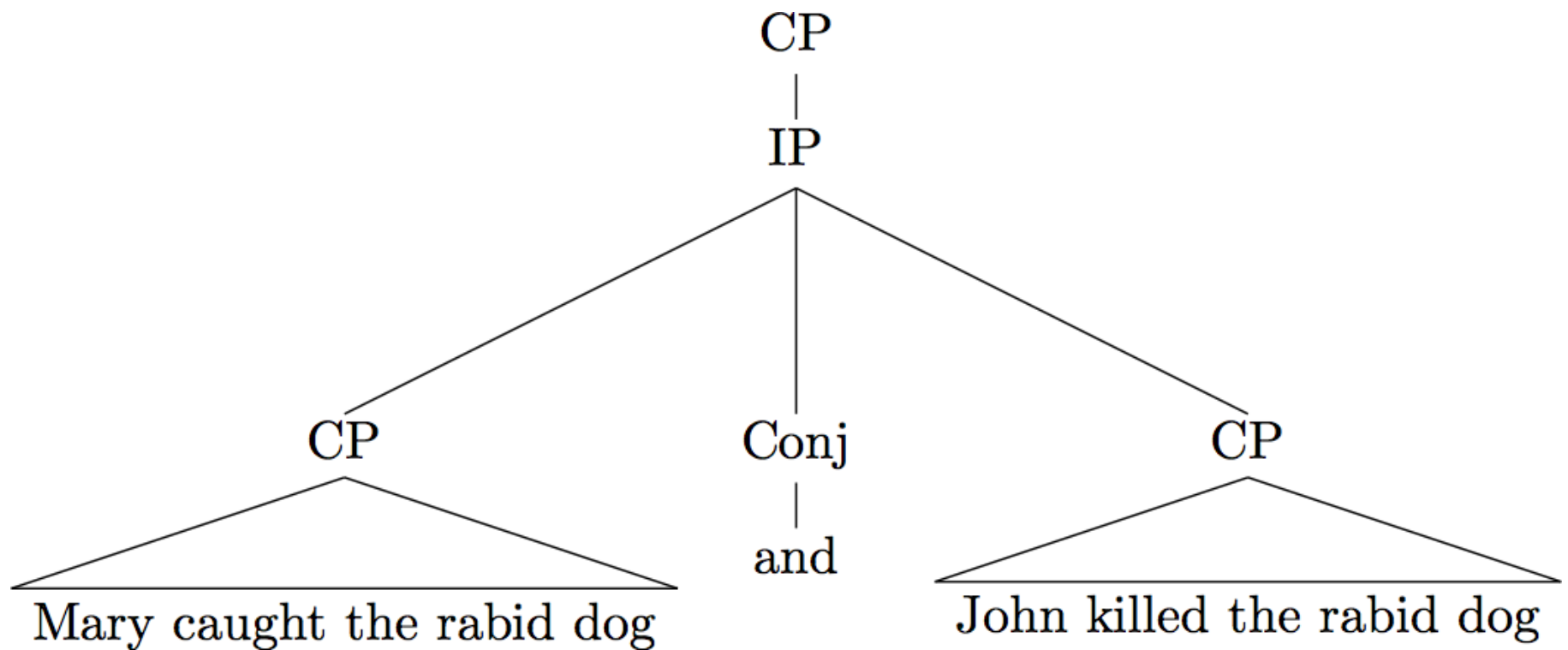
THE WOMAN WHO WAS HERE BELIEVED THAT THE MAN WAS ILL

THE WOMAN BELIEVED THAT THE GUY WHO WAS HERE WAS ILL

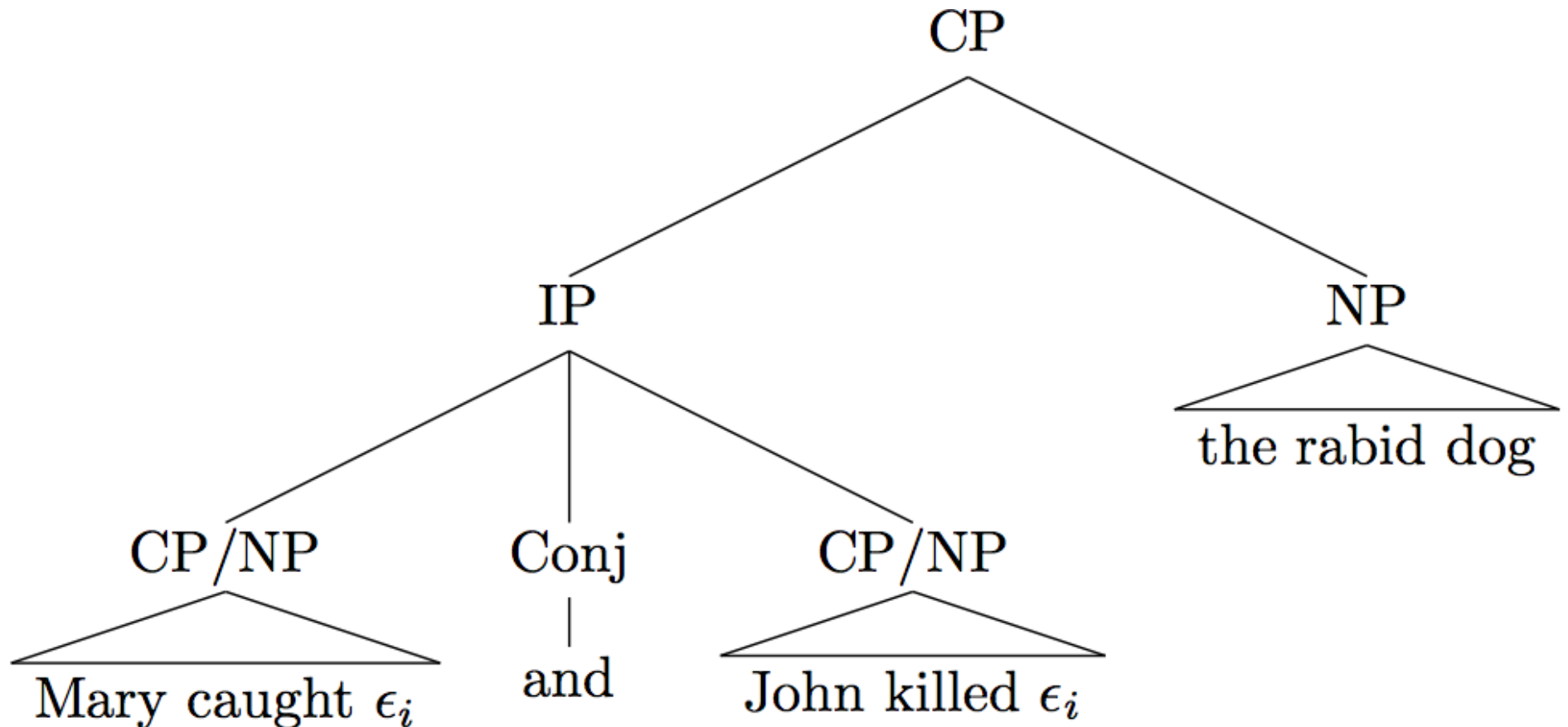
THE WOMAN BELIEVED THAT THE GUY WAS ILL WHO WAS HERE

Can we handle these with the same machinery?

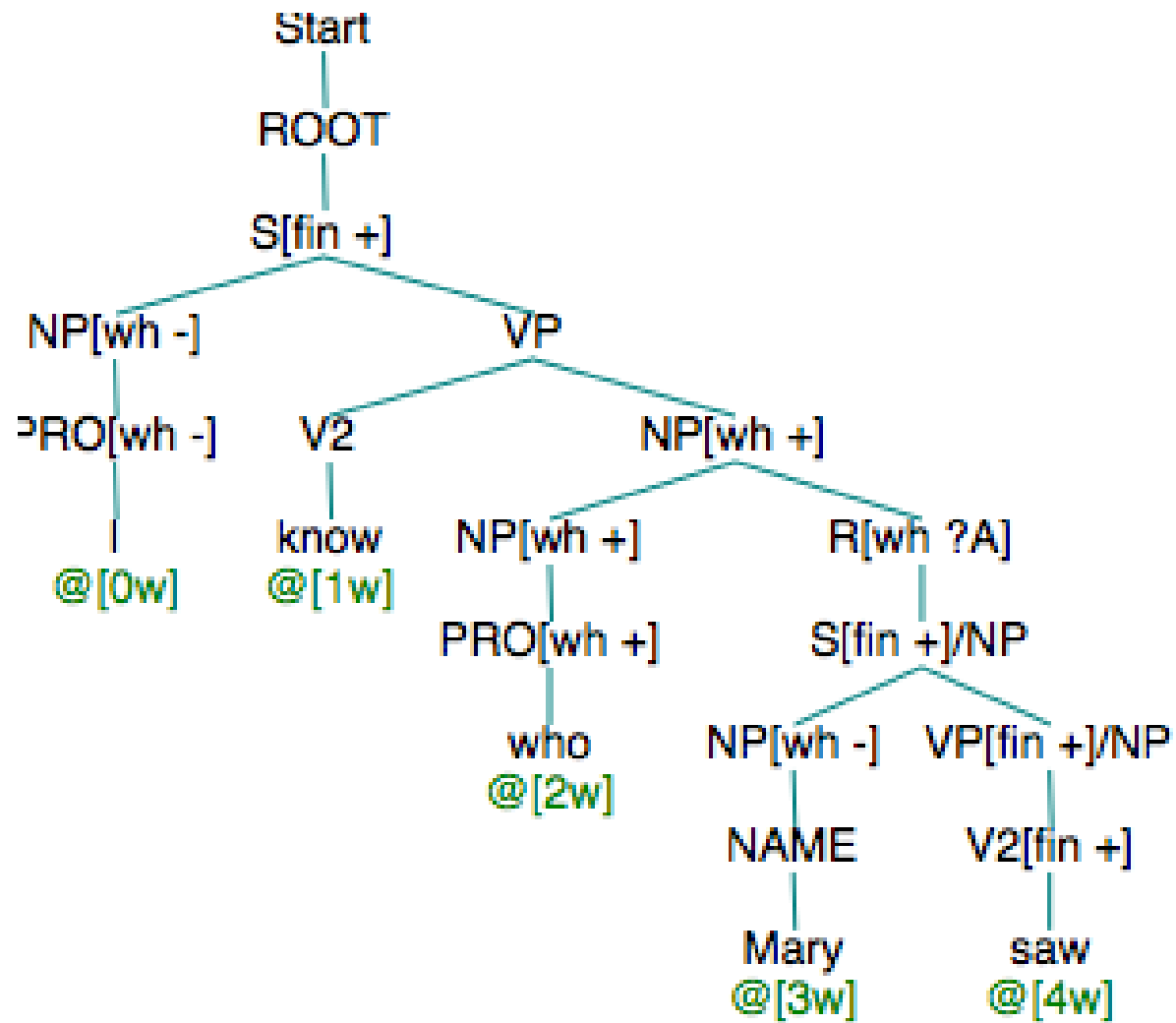
Another example: conjunctions & ‘displacement’

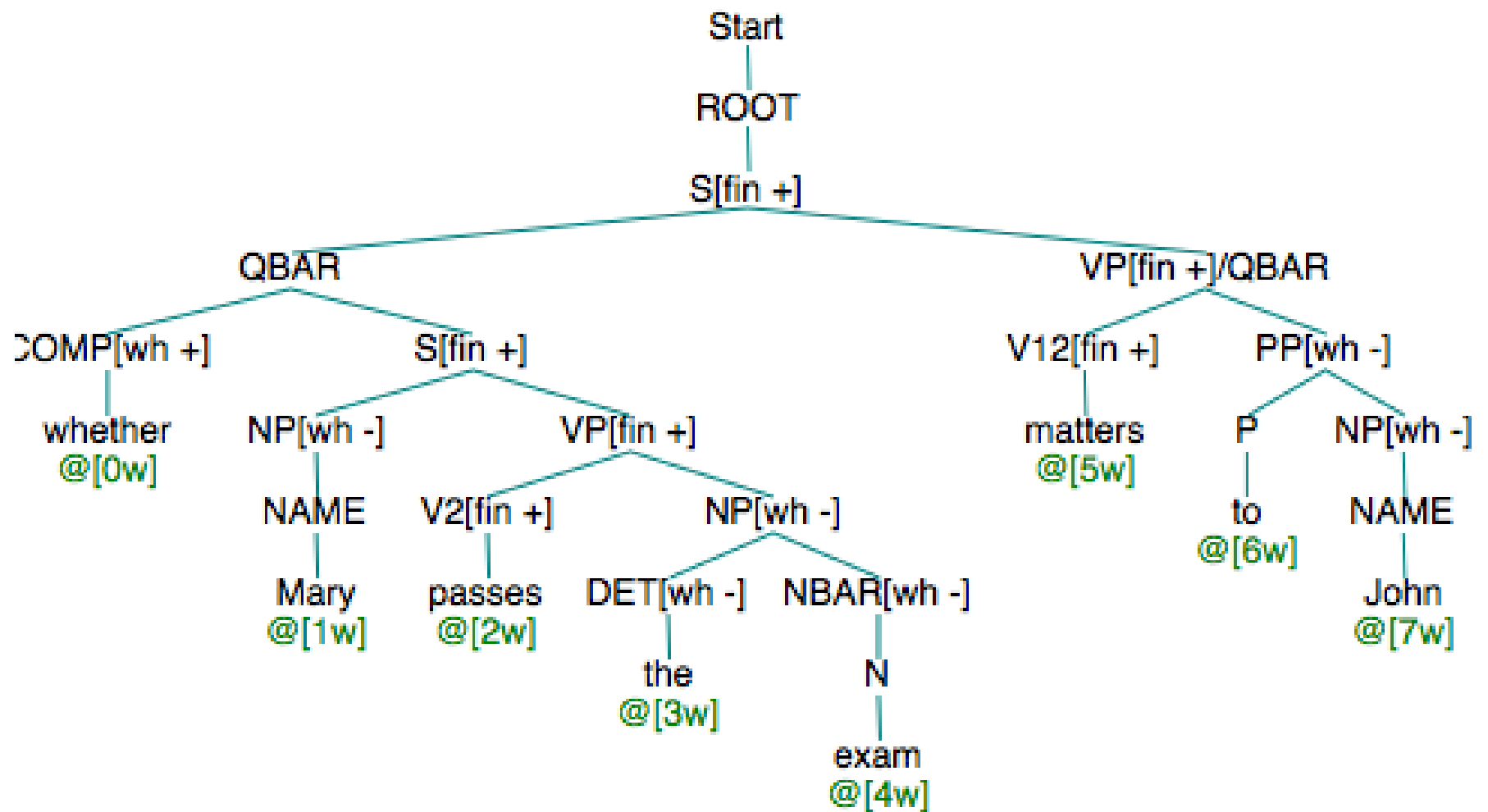


What if we move the object?
“right-node raising”

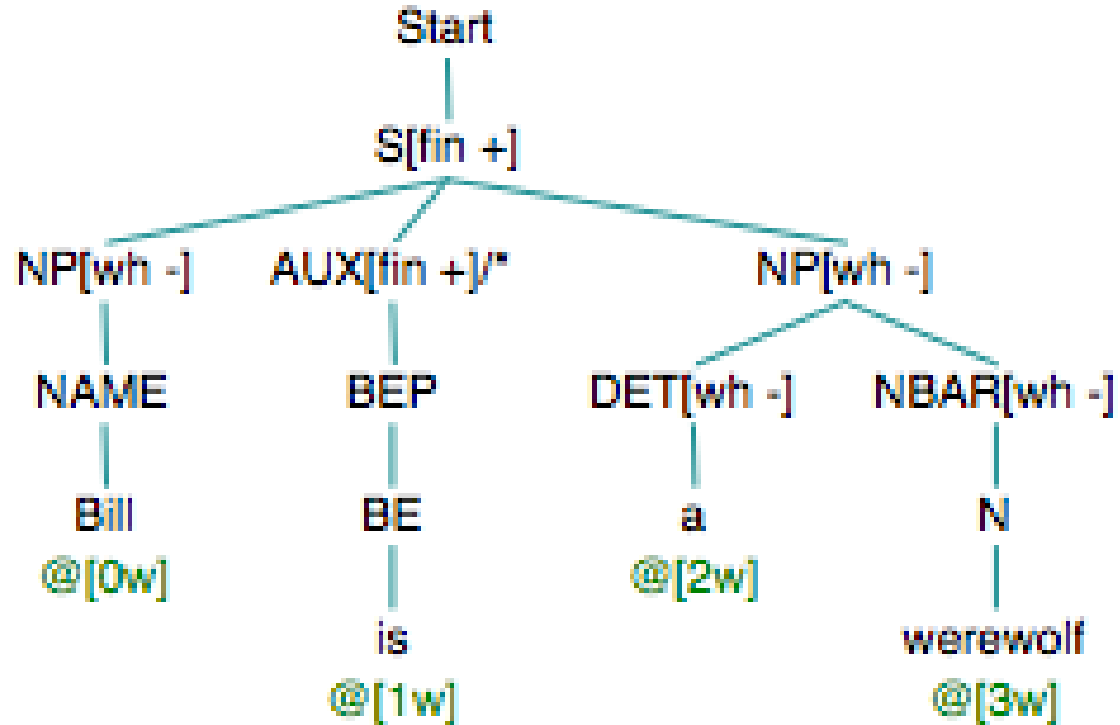


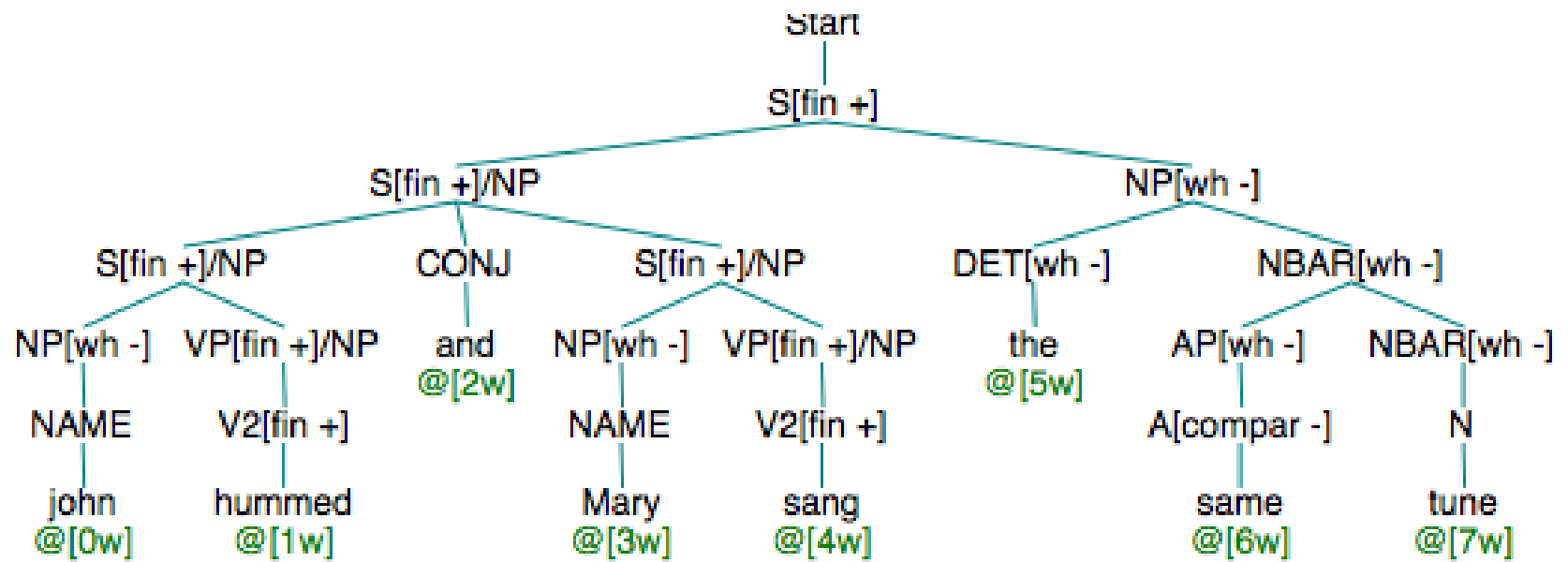
Note conjunction of ‘likes with likes’

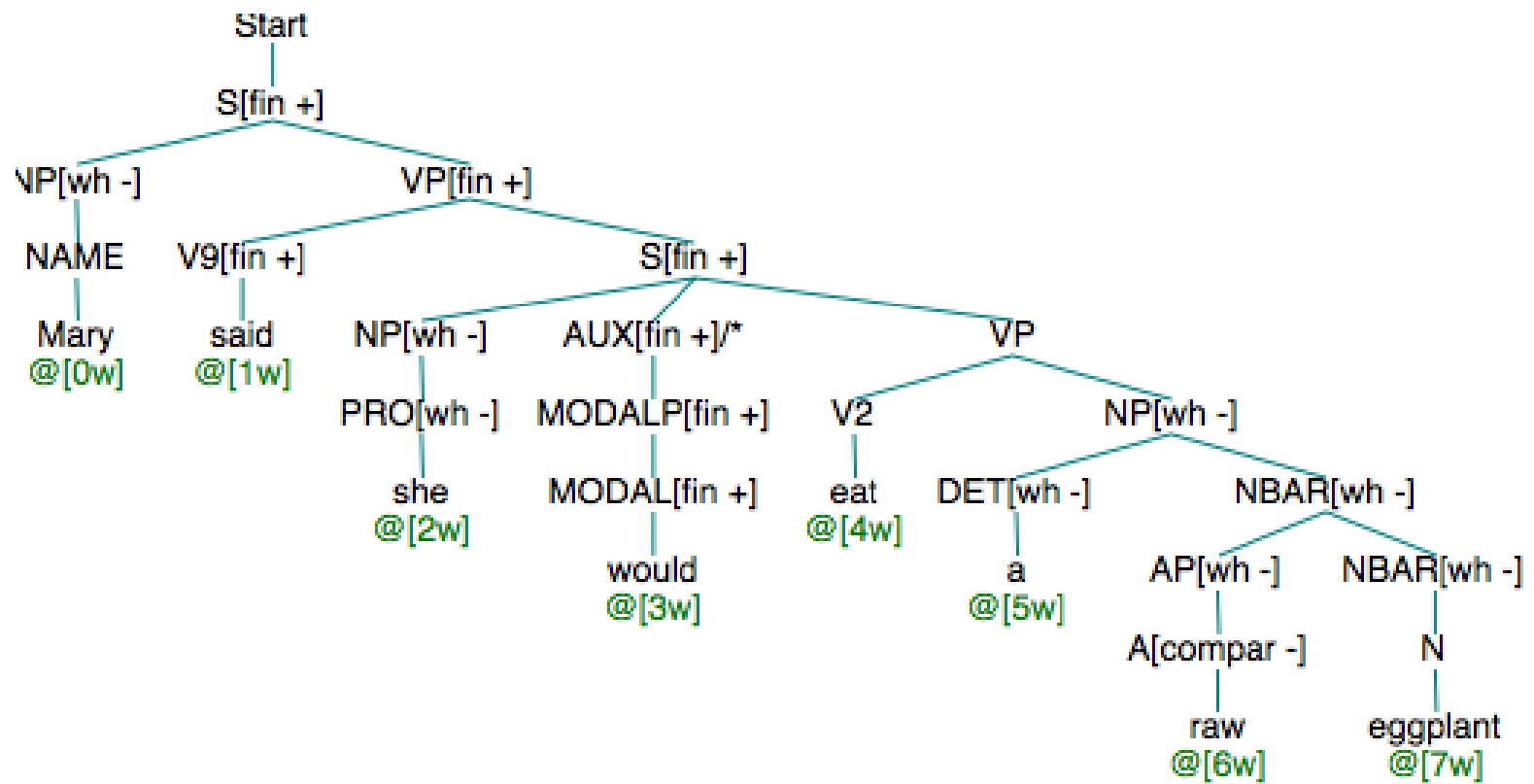




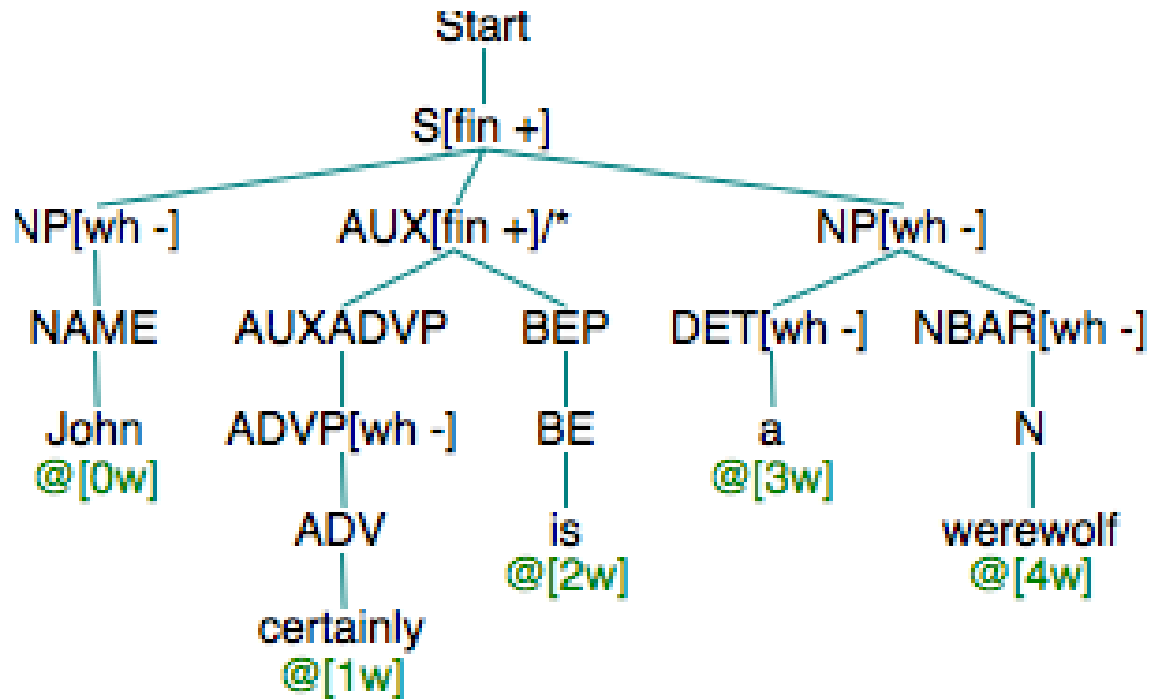
Where wolf?

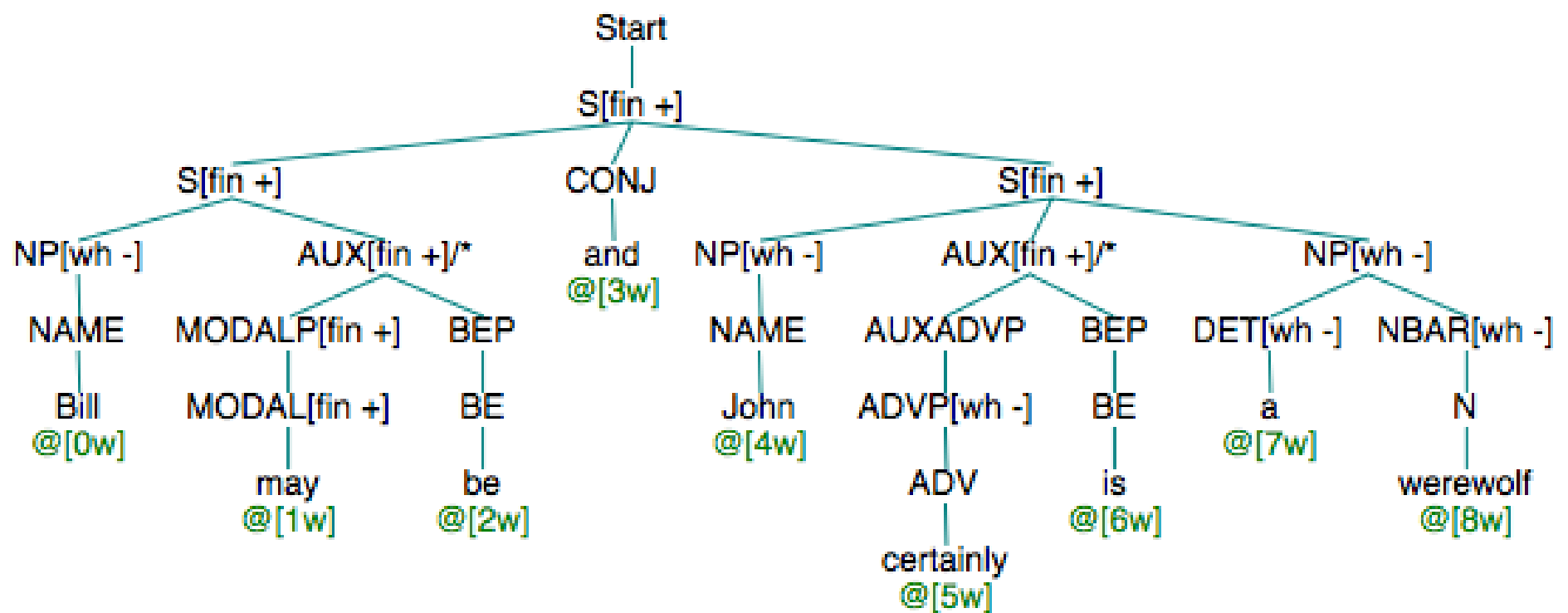


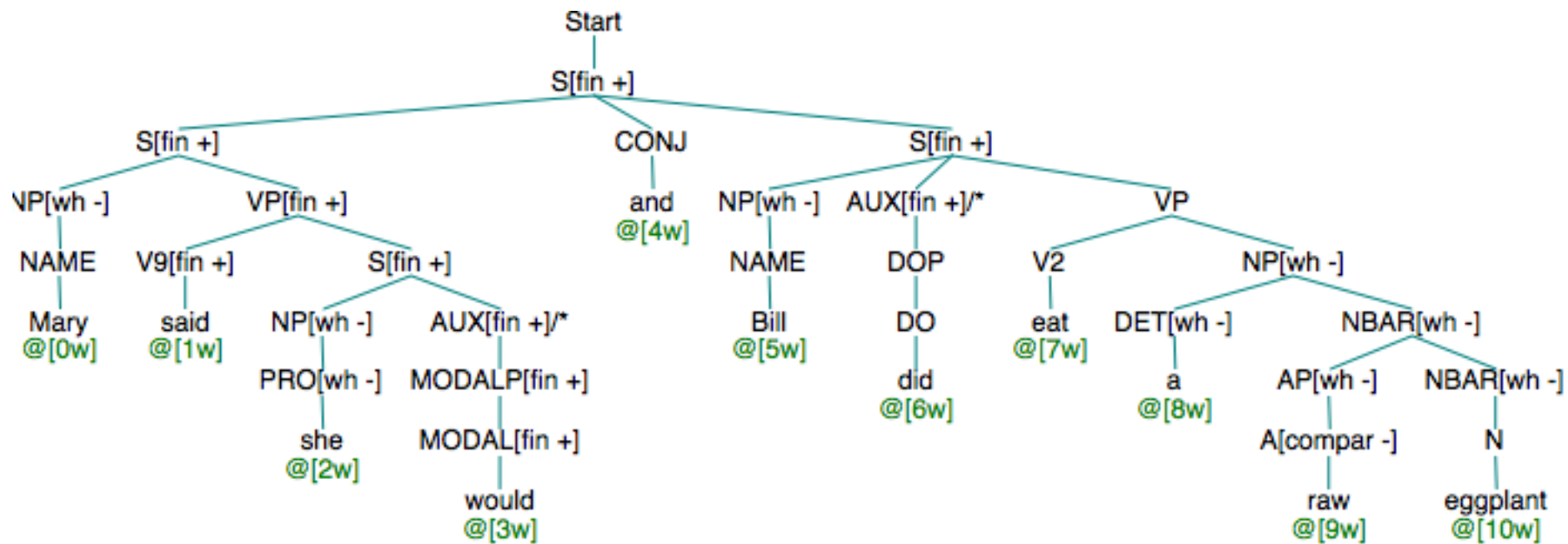


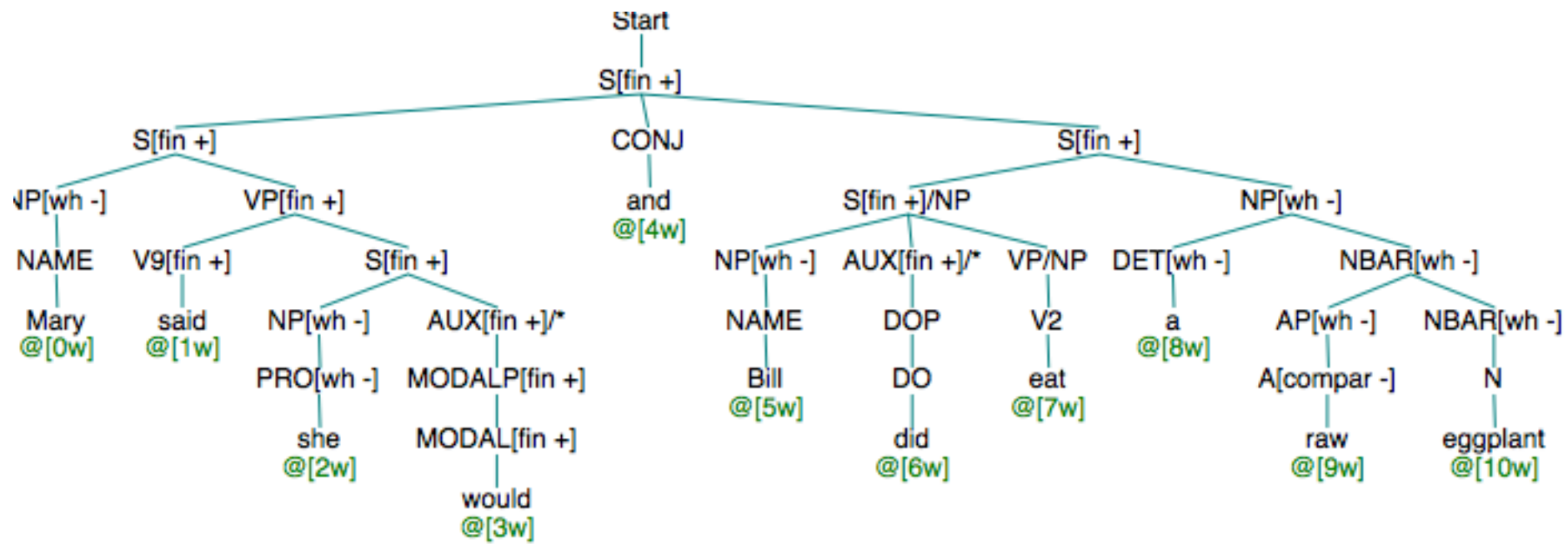


Where wolf?









What about...

Which violins are these sonatas easy to ___ play ___ on ?

Which report did you file ___ without reading ___ ?

The Menu Bar

- Why context-free grammars are no good: representation and computational issues
- Slash & Burn grammar: mind the gap
- Featuritis!
- WHY do we choose certain rules and not others?
- What's the matter with Probabilistic context-free kansas?
- How to fix – a bit – with *lexicalized* context-free grammars; treebank parsing

Feature-based parsing

- Main point is to use [feature value] tuples, in a simple ‘flat’ list, e.g., {[person 1], [number plural]}
- Then feature agreement is checked via unification
- Conflicting feature-value assignments ruled out
- Two questions about computational power:
 1. What is the computational complexity of this system?
 2. Do we need anything *more* powerful than this?

Our feature structures

- NP[plural ?B] -> DET[plural ?B] N[plural ?B]
- VP[fin ?A, person ?B, plural ?C] -> V2[fin ?A, person ?B, plural ?C] NP
- Maria NAME[person 3, plural -]

pykimmo entry for Verb (after analysis):

- **+e Suffix** "[fin +, tense pres, mode ind, person 3, plural -]"

Operations on Feature Structures

- What will we need to do to these structures?
 - Check the **compatibility** of two structures
 - **Merge** the information in two structures
- We can do both using **unification**
- We say that two feature structures **can be unified** if the component features that make them up are **compatible**
 - $[\text{Num SG}] \cup [\text{Num SG}] = [\text{Num SG}]$
 - $[\text{Num SG}] \cup [\text{Num PL}]$ fails!
 - $[\text{Num SG}] \cup [\text{Num } [\]] = [\text{Num SG}]$
 - $[\text{Num SG}] \cup [\text{Pers 3}] = [\text{Num SG, Pers 3}]$

One more trick: instead of slash, use
features!

Rule with ‘slash’ category:
 $IP/NP \rightarrow NP VP/NP$

Convert to ‘feature’ “slash” with “value” NP:
 $IP[Slash ?] \rightarrow NP VP[Slash ?]$

Features and Earley Parsing

- Goal:
 - Use feature structures to provide richer representation
 - Block entry into chart of ill-formed constituents
- Changes needed to Earley parser:
 - Add feature structures to grammar rules, & lexical entries
 - Add field to states containing set representing feature structure corresponding to state of parse, e.g.

$S \rightarrow \bullet NP VP, [0,0], [\quad], \text{Set} = [\text{plural } -]$

- Add new test to Completer operation
 - Recall: Completer adds new states to chart by finding states whose • can be advanced (i.e., category of next constituent matches that of completed constituent)
 - Now: Completer will only advance those states if their feature structures unify
- New test for whether to enter a state in the chart
 - Now feature structures may differ, so check must be more complex
 - Suppose feature structure is more specific than existing one tied to this state? Do we add it?
 - Yes...

How to formalize?

- Let F be a finite set of feature names, let A be a finite set of feature values
- Let p be a function from feature names to permissible feature values, that is,

$$p: F \rightarrow 2^A$$

- Now we can define a specification as a triple,

$$\langle F, A, p \rangle$$

- This is a partial function from feature names to feature values

Example

- $F = \{CAT, PLU, PER\}$

- $p:$

$p(CAT) = \{V, N, ADJ\}$

$p(PER) = \{1, 2, 3\}$

$p(PLU) = \{+, -\}$

$sleep = \{[CAT\ V, PLU\ -, PER\ 1]\}$

$sleep = \{[CAT\ V, PLU\ +, PER\ 1]\}$

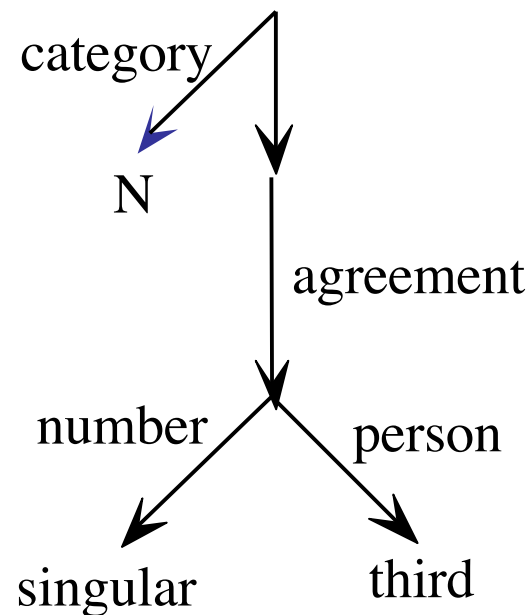
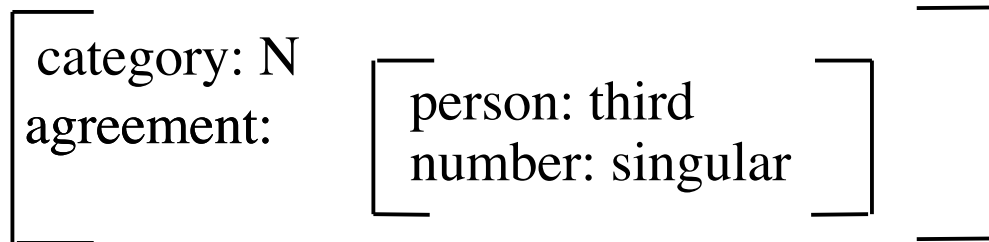
$sleeps = \{[CAT\ V, PLU\ -, PER\ 3]\}$

Checking whether features are compatible is relatively simple here...how bad can it get?

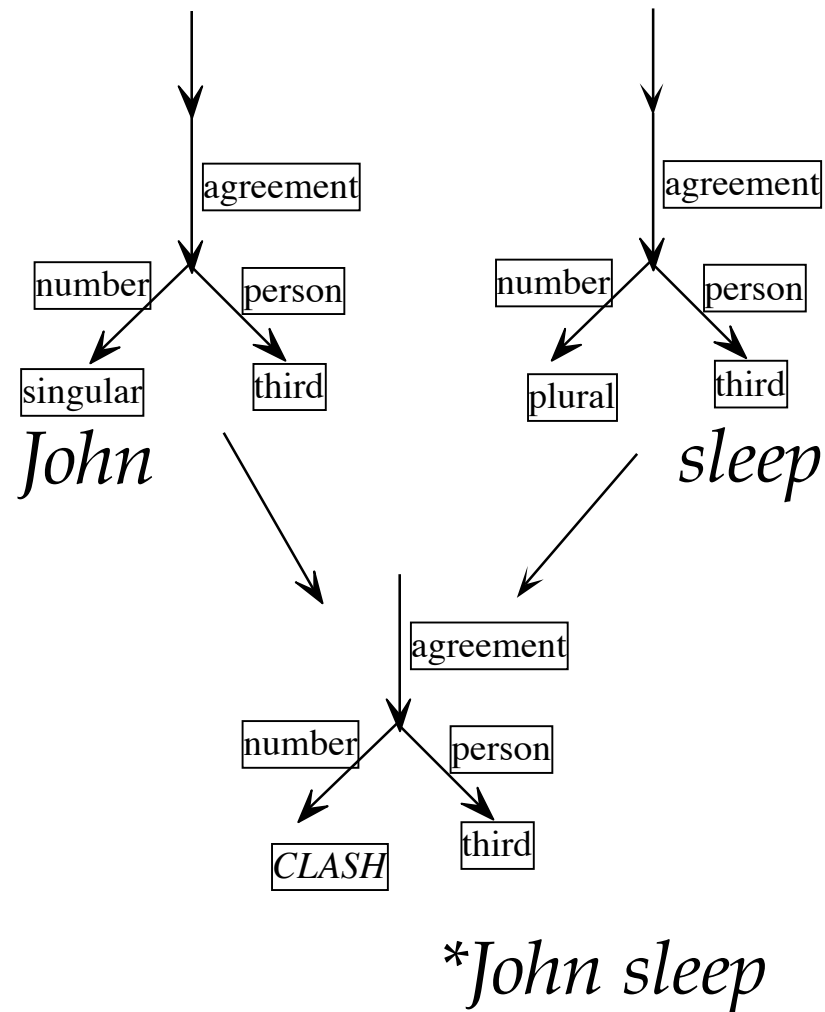
Important question

- Do features have to be more complicated than this?
- More: hierarchically structured (feature structures) (directed acyclic graphs, DAGs, or even beyond)
- Then *checking* for feature compatibility amounts to *unification*
- Example

Features and grammars: general case results in directed acyclic graphs (DAGs)



Feature checking by unification



- How do we define 3plNP?
- How does this improve over the CFG solution?
- Feature values can be feature structures themselves
 - Useful when certain features commonly co-occur, e.g. number and person

$$\left[\begin{array}{l} \textit{Cat} \quad \textit{NP} \\ \textit{Agr} \quad \left[\begin{array}{l} \textit{Num} \quad \textit{SG} \\ \textit{Pers} \quad 3 \end{array} \right] \end{array} \right]$$

- Feature path: path through structures to value, e.g.

Agr → *Num* → *SG*

- Structures are compatible if they contain no features that are incompatible
- Unification of two feature structures:
 - Are the structures compatible?
 - If so, return the union of all feature/value pairs
- A failed unification attempt

$$\left[\begin{array}{l} Agr \ 1 \left[\begin{array}{l} Num \ SG \\ Pers \ 3 \end{array} \right] \\ Subj \ \left[\begin{array}{l} Agr \\ \quad \quad \quad 1 \end{array} \right] \end{array} \right] \cup \left[\begin{array}{l} Agr \ \left[\begin{array}{l} Num \ Pl \\ Pers \ 3 \end{array} \right] \\ Subj \ \left[\begin{array}{l} Agr \ \left[\begin{array}{l} Num \ PL \\ Pers \ 3 \end{array} \right] \end{array} \right] \end{array} \right]$$

Features, Unification and Grammars

- How do we incorporate feature structures into our grammars?
 - Assume that constituents are objects which have feature-structures associated with them
 - Associate sets of unification constraints with grammar rules
 - Constraints must be satisfied for rule to be satisfied
- For a grammar rule $\beta_0 \rightarrow \beta_1 \dots \beta_n$
 - $\langle \beta_i \text{ feature path} \rangle = \text{Atomic value}$
 - $\langle \beta_i \text{ feature path} \rangle = \langle \beta_j \text{ feature path} \rangle$
- NB: if simple feat-val pairs, no arbitrary nesting, then no need for paths

How can we parse with feature structures?

- Unification operator: takes 2 feature structures and returns *either* a merged feature structure or *fail*
- Input structures represented as DAGs
 - Features are labels on edges
 - Values are atomic symbols or DAGs
- Unification algorithm goes through features in one input DAG₁ trying to find corresponding features in DAG₂ – if all match, success, else fail
- WE USE A MUCH SIMPLER kind of feature structure

Evidence that you don't need this much power in human language

- Linguistic evidence: looks like you just check whether features are *nondistinct*, rather than equal or not – variable *matching*, not variable substitution
- Full unification lets you generate unnatural languages:
 $\{a^i \mid s.t. i \text{ is a power of } 2\}$ – e.g., $a, aa, aaaa, aaaaaaaaa, \dots$
why is this ‘unnatural’ – another (seeming) property of natural languages:

Natural languages seem to obey a *constant growth property*

Constant growth property

Claim: \exists Bound \underline{k} on the ‘distance gap’ between any two consecutive sentences in this list, which can be specified in advance (fixed)

- ‘Intervals’ between valid sentences cannot get too big – cannot grow w/o bounds
- We can do this a bit more formally

Constant growth

- Dfn. A language L is semilinear if the number of occurrences of each symbol in any string of L is a linear combination of the occurrences of these symbols in some fixed, finite set of strings of L .
- Dfn. A language L is constant growth if there is a constant c_0 and a finite set of constants C s.t. for all $w \in L$, where $|w| > c_0$, $\exists w' \in L$ s.t. $|w| = |w'| + c$, some $c \in C$
- Fact. (Parikh, 1971). Context-free languages are semilinear, and constant-growth
- Fact. (Berwick, 1983). The power of 2 language is non-constant-growth

General feature grammars – how violate these properties

- Take example from so-called “lexical-functional grammar” but this applies as well to any general unification grammar
- Lexical functional grammar (LFG): add checking rules to CF rules (also variant HPSG)

Example LFG

- Basic CF rule:

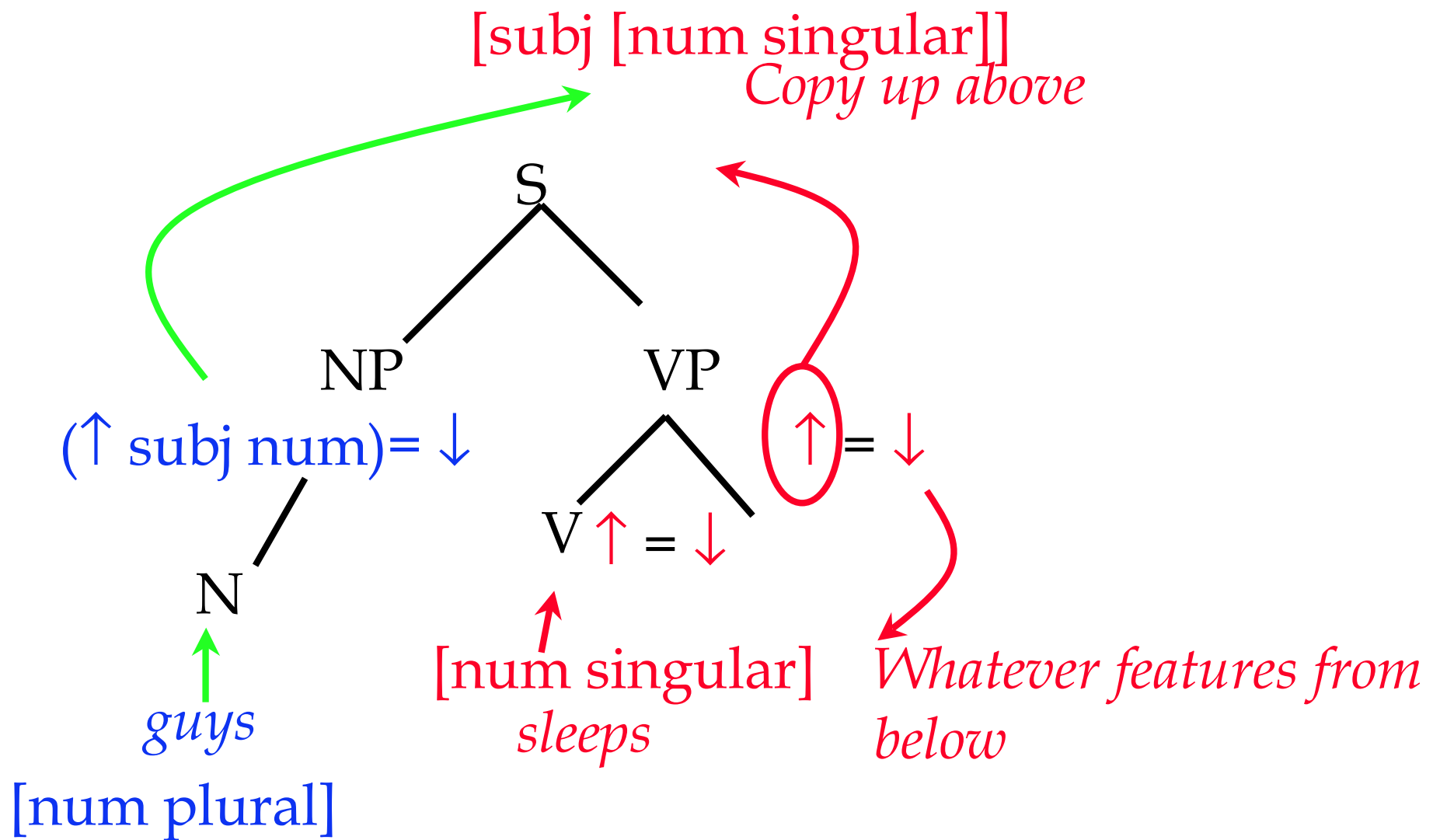
$S \rightarrow NP VP$

- Add corresponding ‘feature checking’

$S \rightarrow NP \quad VP$
 $(\uparrow \text{subj num}) = \downarrow \quad \uparrow = \downarrow$

- What is the interpretation of this?

Applying feature checking in LFG



Alas, this allows non-constant growth, unnatural languages

- Can use LFG to generate power of 2 language

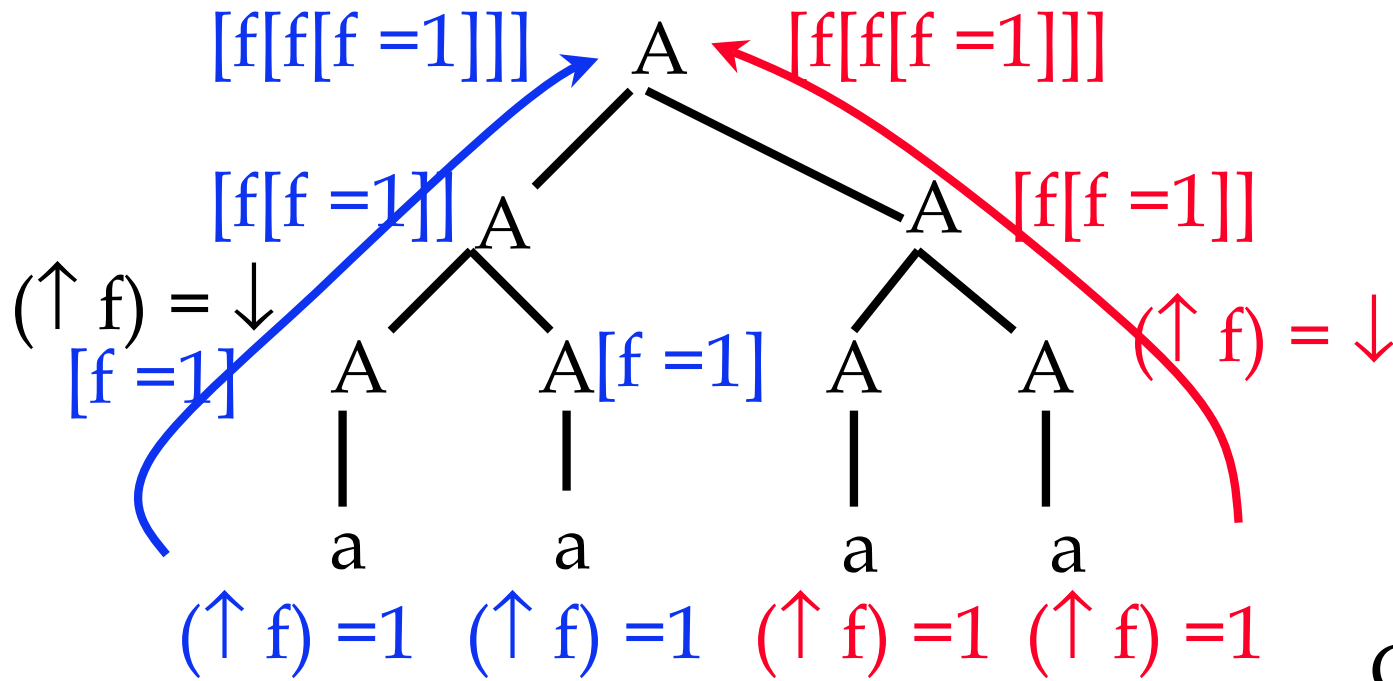
- Very simple to do

- $A \rightarrow A \quad A$
 $(\uparrow f) = \downarrow \quad (\uparrow f) = \downarrow$

$$A \rightarrow a$$
$$(\uparrow f) = 1$$

Lets us `count' the number of embeddings on the right & the left
– make sure a power of 2

Example



Checks ok

If mismatch anywhere, get a feature clash...

