Natural Language Processing

Lecture 9: Otherther Grammar Formalisms
Feature Grammars, Tree Adjoining Grammar, CCG

10/18/2018

COMS W4705
Daniel Bauer

Limitations of Context Free Grammars

- CFGs only model *well-nested, local* structures.
    - Once a constituent has been parsed, the only information we have about it is its category (NP, VP, ...)
    - How would you deal with sucategorization?
      Mary gave the book to John. **Mary gave John.*
    - Ideally, we would like to model the complete grammatical context of each word locally.
- Cannot capture certain languages at all (non-context-free languages, for example long-distance/crossing dependencies).

Problems with CFG

- Agreement
- Subcategorization
- Long-distance dependencies

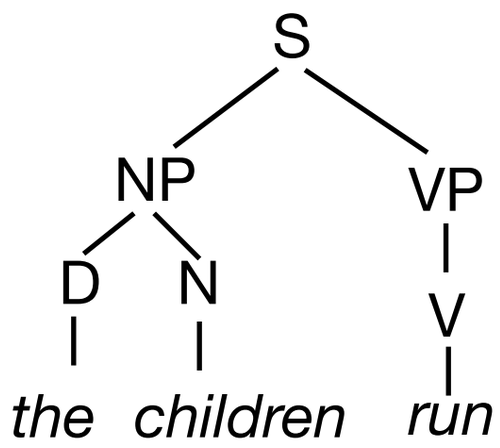
  The book that Kim read.

Agreement

- Many languages have grammatical agreement between words in a sentences (for example, subject-verb agreement in English).

  - *The girl laughs.* ** The girl laugh.*
  - *The children run.* ** The children runs*
    **A children run*

- How would you model this using CFG?



Agreement

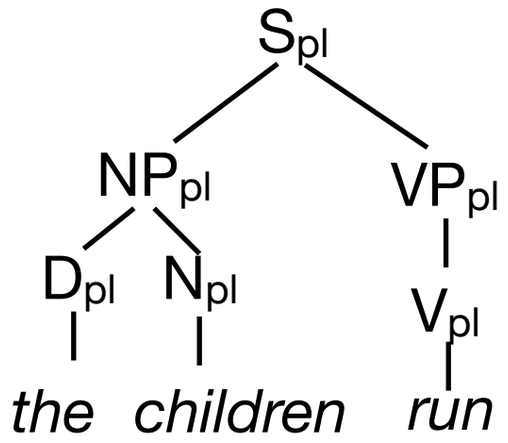
- Many languages have grammatical agreement between words in a sentences (for example, subject-verb agreement in English).

  - *The girl laughs.* ** The girl laugh.*
  - *The children run.* ** The children runs*
    **A children run*

  - How would you model this using CFG?

All information needs to be stored in the nonterminals!

This can quickly blow-up the nonterminal alphabet.



Agreement

- Another example: determiner-adjective-noun agreement in Spanish.

- ***la*** *taz**a*** *roj**a***
  the$_{fem,sg}$ *cup*$_{\text{fem,sg}}$ *red*$_{\text{fem,sg}}$
  (the red cup)

- ***el*** *vas**o*** *roj**o***
  the$_{masc,sg}$ *glass*$_{masc,\text{sg}}$ *red*$_{masc,\text{sg}}$
  (the red glass)

- **los** vas**os** roj**os**
  the$_{masc,pl.}$ *cup*$_{masc,pl.}$ *red*$_{masc,pl}$
  (the red cups)

Feature Grammars

- Based on CFG, but instead of using a fixed set of atomical nonterminals, use a *factored representation* / "*feature structure*" to represent grammatical information.

- Each word is associated with a feature structure:

$$\text{N} \rightarrow \text{taza} \begin{bmatrix} CAT & N \\ PRED & taza \\ AGREEMENT & \begin{bmatrix} NUMBER & fem \\ GENDER & sg \end{bmatrix} \end{bmatrix}$$

  - combining phrases = *unifying* feature structures. Combining phrases is only allowed if *unification* succeeds.

  - The result of the combination is another feature structure, so we are constructing a feature structure for the entire sentence bottom-up.

(Moore 1989), (Knight 1989)
Head Driven Phrase Structure Grammar (Pollard & Sag,1994)
Lexical Functional Grammar (Bresnan, 2001)

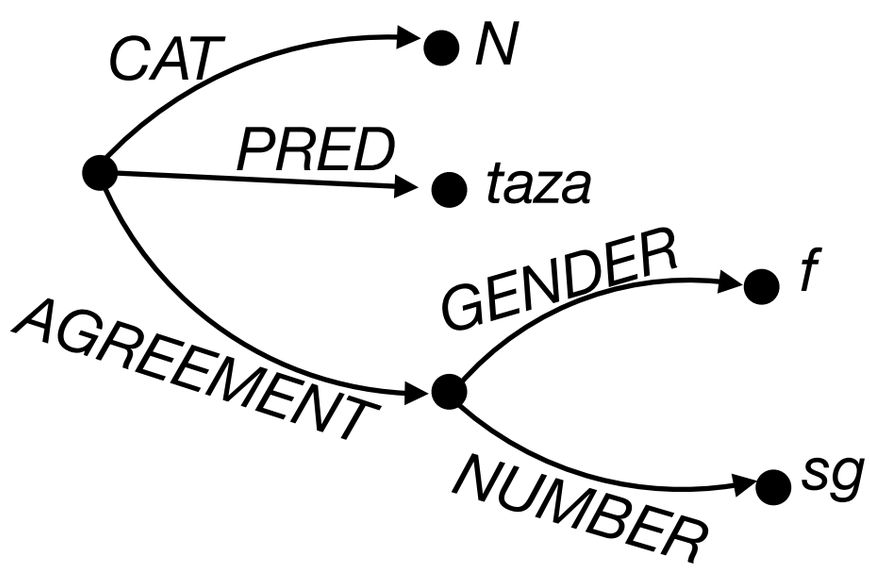
Feature Structures

- Attribute/Value Matrix (AVM):

$$\begin{bmatrix} \textit{CAT} & \textit{N} \\ \textit{PRED} & \textit{taza} \\ \textit{AGREEMENT} & \begin{bmatrix} \textit{gender} & \textit{f} \\ \textit{number} & \textit{sg} \end{bmatrix} \end{bmatrix}$$

recursive definition:
values are primitives or other feature structures.

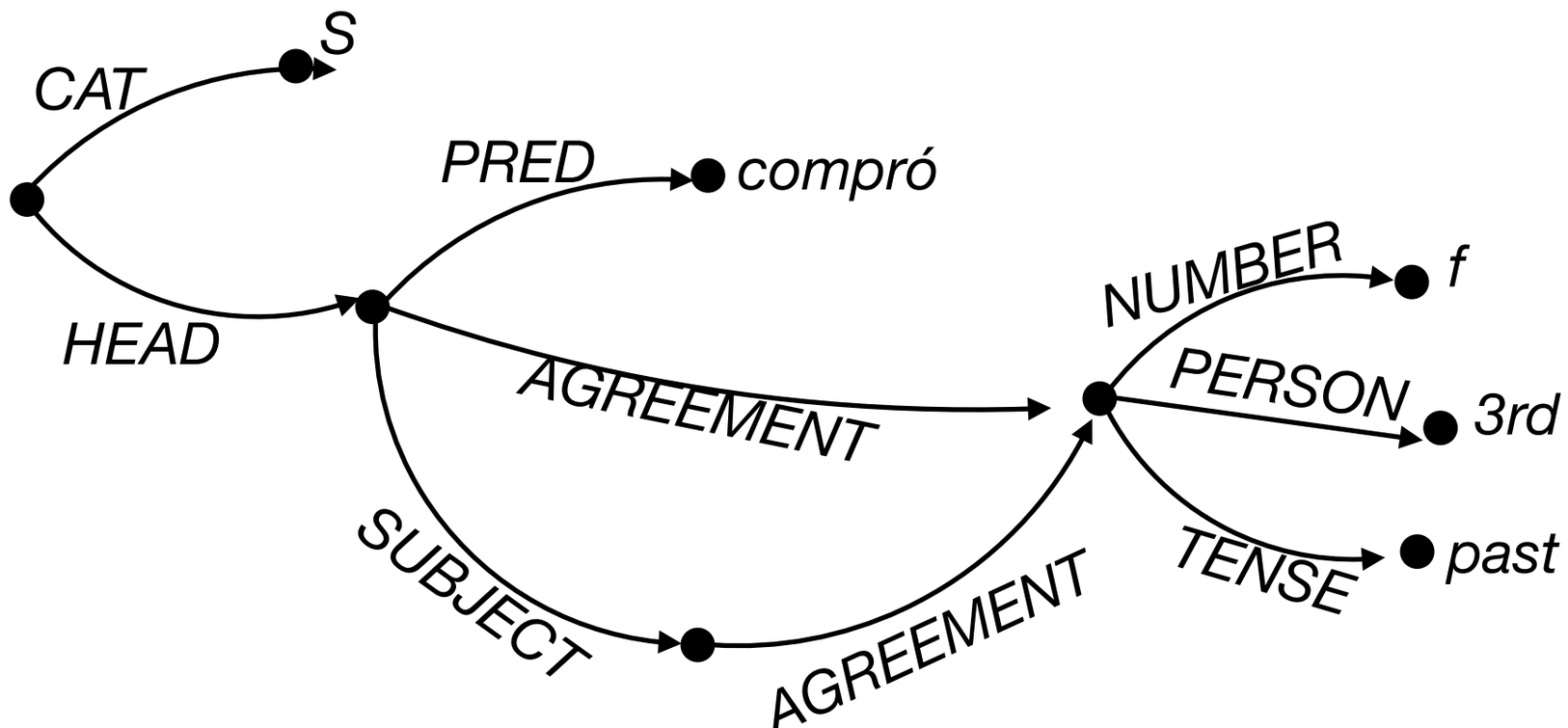
- This is equivalent to a Directed Acyclic Graph (DAG):



Reentrancy

- The same feature structure may be the value for different features.

$$\begin{bmatrix} \textit{CAT} & \textit{S} \\ \textit{HEAD} & \begin{bmatrix} \textit{PRED} & \textit{compró} \\ \textit{AGREEMENT} & \boxed{1}\begin{bmatrix} \textit{number} & \textit{sg} \\ \textit{person} & \textit{3rd} \\ \textit{tense} & \textit{past} \end{bmatrix} \\ \textit{SUBJECT} & \begin{bmatrix} \textit{AGREEMENT} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$



Unification

- During derivation of a sentence, the feature structures of the children are combined using *unification.*

- Unification is binary operation defined on feature structures. It results in a new feature structure or in **Failure.**

$$\begin{bmatrix} gender & f \end{bmatrix} \sqcup \begin{bmatrix} gender & f \end{bmatrix} = \begin{bmatrix} gender & f \end{bmatrix}$$

$$\begin{bmatrix} gender & f \end{bmatrix} \sqcup \begin{bmatrix} gender & m \end{bmatrix} = \textbf{Failure}$$

$$\begin{bmatrix} gender & f \end{bmatrix} \sqcup \begin{bmatrix} number & sg \end{bmatrix} = \begin{bmatrix} gender & f \\ number & sg \end{bmatrix}$$

Unification

$$\begin{bmatrix} \textit{AGREEMENT} & \boxed{1}\begin{bmatrix} \textit{number} & \textit{sg} \\ \textit{person} & \textit{3rd} \end{bmatrix} \\ \textit{SUBJECT} & \begin{bmatrix} \textit{AGREEMENT} & \boxed{1} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \textit{SUBJECT} & \begin{bmatrix} \textit{AGREEMENT} & \begin{bmatrix} \textit{number} & \textit{sg} \\ \textit{gender} & \textit{fem} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \textit{AGREEMENT} & \boxed{1}\begin{bmatrix} \textit{number} & \textit{sg} \\ \textit{person} & \textit{3rd} \\ \textit{gender} & \textit{fem} \end{bmatrix} \\ \textit{SUBJECT} & \begin{bmatrix} \textit{AGREEMENT} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

Unification Constraints

- Grammar rules specify which parts of the feature structures associated with the instance of a phrase need to be unified.

NP → D NP' ⟨D AGREE = NP' HEAD AGREE⟩
⟨NP HEAD = NP' HEAD⟩

NP' → N Adj ⟨N AGREE = Adj AGREE⟩
⟨NP' HEAD AGREE = N AGREE⟩

D → *la* ⟨*D AGREE NUMBER = sg*⟩
⟨*D AGREE GENDER = f*⟩

N → *taza* ⟨N *AGREE NUMBER = sg*⟩
⟨N *AGREE GENDER = f*⟩

Adj → *roja* ⟨Adj *AGREE NUMBER = sg*⟩
⟨Adj *AGREE GENDER = f*⟩

- Important: "=" means "must unify with"

Feature Grammars

- During parsing, the unification constraints exclude impossible parses. If parsing succeeds, a feature structure for the entire sentence has been computed.

D → *la* ⟨*D AGREE NUMBER* = *sg*⟩
⟨*D AGREE GENDER* = *f*⟩

N → *taza* ⟨N *AGREE NUMBER* = *sg*⟩
⟨N *AGREE GENDER* = *f*⟩

Adj → *roja* ⟨Adj *AGREE NUMBER* = *sg*⟩
⟨Adj *AGREE GENDER* = *f*⟩

$$\begin{bmatrix} \text{AGREE} \begin{bmatrix} \textit{gender} & \textit{f} \\ \textit{number} & \textit{sg} \end{bmatrix} \end{bmatrix}$$
D
|
la

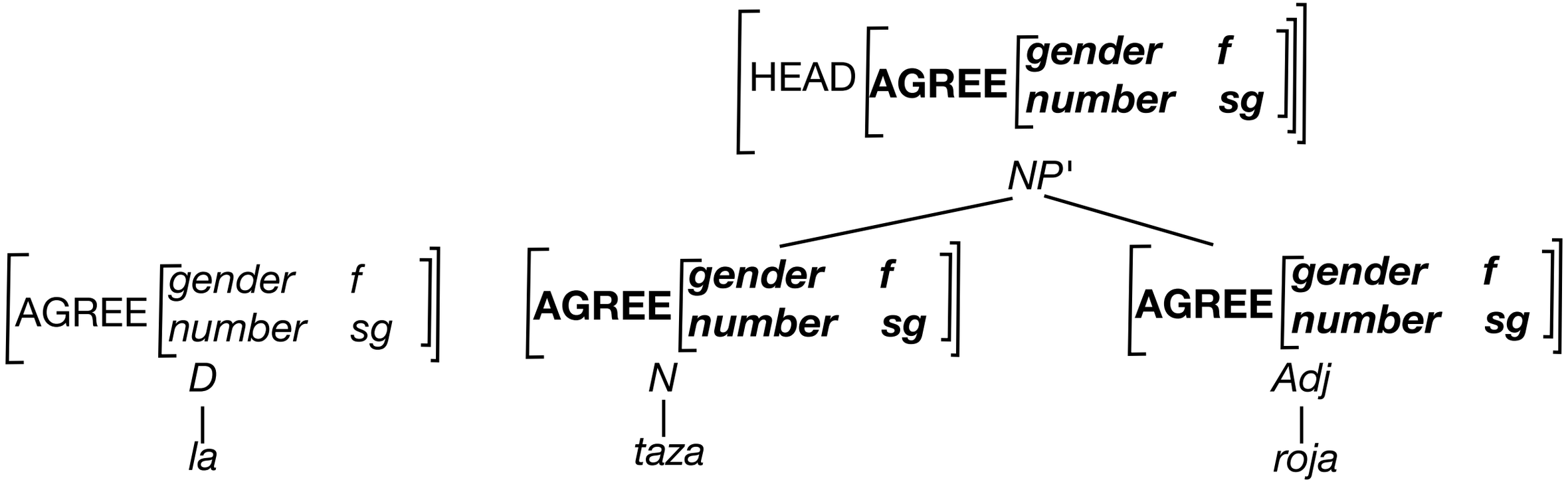
$$\begin{bmatrix} \text{AGREE} \begin{bmatrix} \textit{gender} & \textit{f} \\ \textit{number} & \textit{sg} \end{bmatrix} \end{bmatrix}$$
N
|
taza

$$\begin{bmatrix} \text{AGREE} \begin{bmatrix} \textit{gender} & \textit{f} \\ \textit{number} & \textit{sg} \end{bmatrix} \end{bmatrix}$$
Adj
|
roja

Feature Grammars

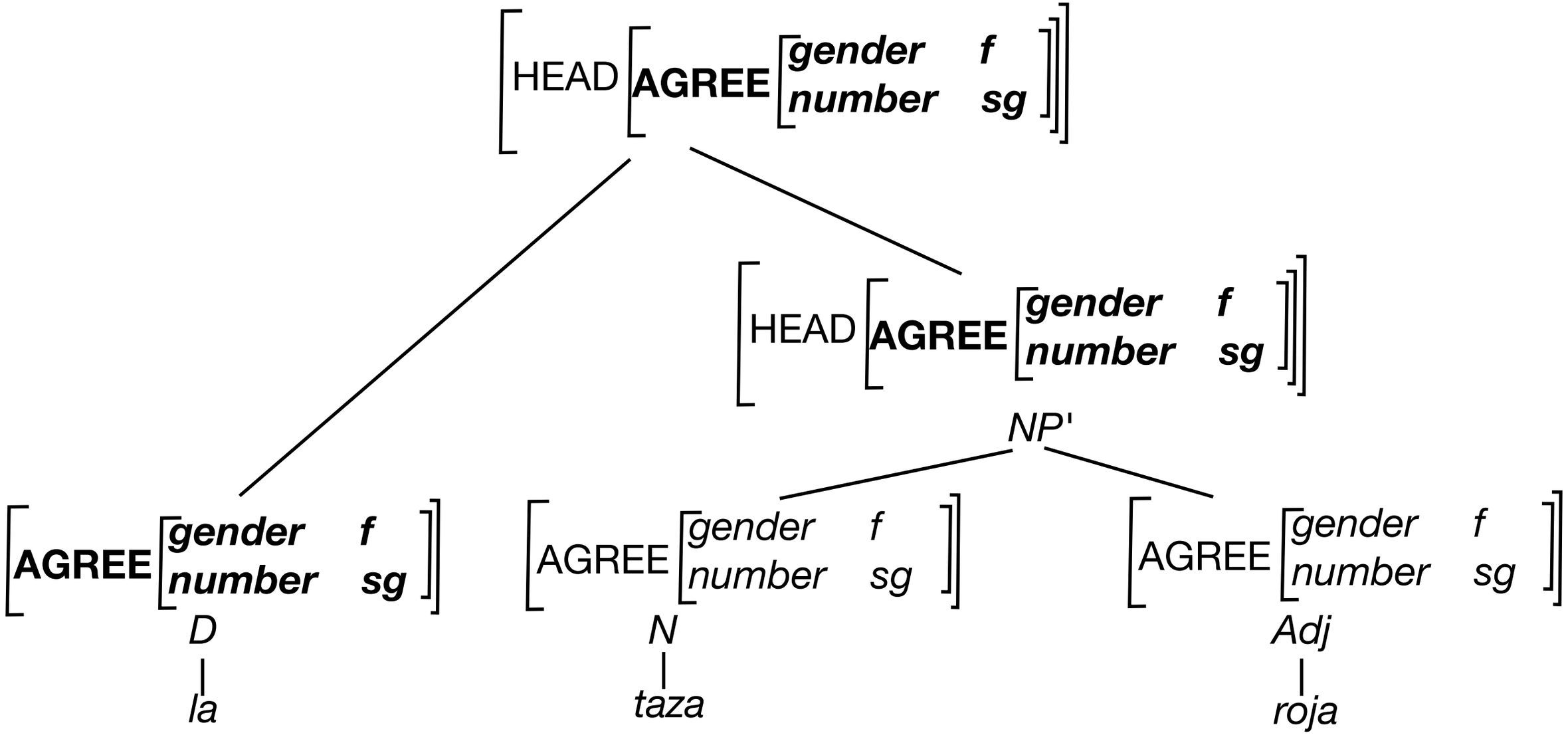
- During parsing, the unification constraints exclude impossible parses. If parsing succeeds, a feature structure for the entire sentence has been computed.

NP' → N Adj ⟨N AGREE = Adj AGREE⟩
⟨NP' HEAD AGREE = N AGREE⟩



Feature Grammars

NP → D NP' ⟨D AGREE = NP' HEAD AGREE⟩
⟨NP HEAD = NP' HEAD⟩



Feature Grammars - Observations

- Extremely flexible framework, can be used to deal with agreement, subcategorization, long-distance dependencies.

- Parsing results in a feature structure / DAG. Parsing with feature grammars can be seen as string-to-DAG translation.

- Works with small hand-written grammars, difficult to scale.
    - Parsing: Easy to extend CKY(but unification is expensive!)
    - Difficult to learn unification constraints.
    - Difficult to make probabilistic.

Categorical Grammar

- An alternative approach to building phrase structure.
    - Phrases are associated with syntactic *categories* rather than non-terminal symbols.
        - Each lexicon entry is associated with a lexical category.
        - There is a small set of rules to guide combination of these categories in context.
- Inspired by lambda-calculus: Categories are higher-order functions applied to other categories.

Ajdukiewicz (1935), Bar-Hillel (1953)

Categorical Grammar

- A category is either:
  - An atomic constituent symbol NP, S, ...
  - A single-argument function mapping a desired category to a category.

(X / Y) - take category Y as an argument (to the right), return X

(X \ Y) - take category Y as an argument (to the left), return Y.

Example Lexicon:

```
Mary      :=  NP
musicals  :=  NP
likes     :=  ((S \ NP) / NP)
gives     := (((S \ NP ) / NP) / NP)
```

Categorical Grammar: Rules and Derivations

- Forward function application: >

$$\frac{(X / Y) \quad Y}{X} >$$

- Backward function application: <

$$\frac{Y \quad (X \setminus Y)}{Y} <$$

Mary	:=	NP
musicals	:=	NP
likes	:=	((S \ NP) / NP)

$$\dfrac{\dfrac{\text{Mary}}{\text{NP}} \quad \dfrac{\dfrac{\text{likes}}{((S \setminus NP) / NP)} \quad \dfrac{\text{musicals}}{\text{NP}}}{(S \setminus NP)} >}{S} <$$

Observation: This is exactly like CFG so far!

Combinatory Categorial Grammar (CCG)

- In addition to forward/backward application, we add a number of function *combinators.*

- Forward composition:

$$\frac{(X / Y) \quad (Y / Z)}{(X / Z)}{>}B$$

- Backward composition:

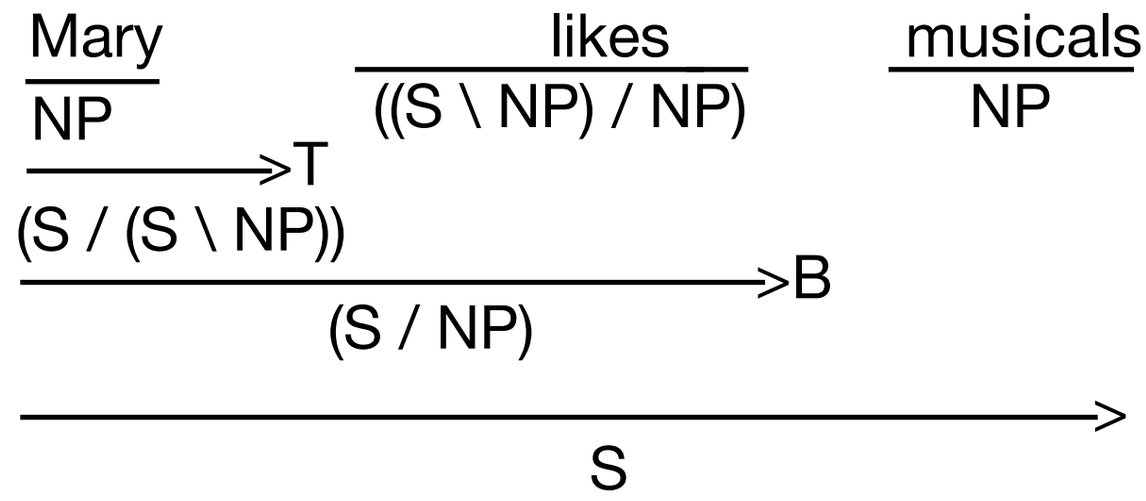
$$\frac{(Y \setminus Z) \quad (X \setminus Y)}{(X \setminus Z)}{<}B$$

- Type raising: $\frac{X}{(T / (T \setminus X))}{>}T$ or $\frac{X}{(T \setminus (T / X))}{<}T$

Steedman (1996)

Type Raising Example

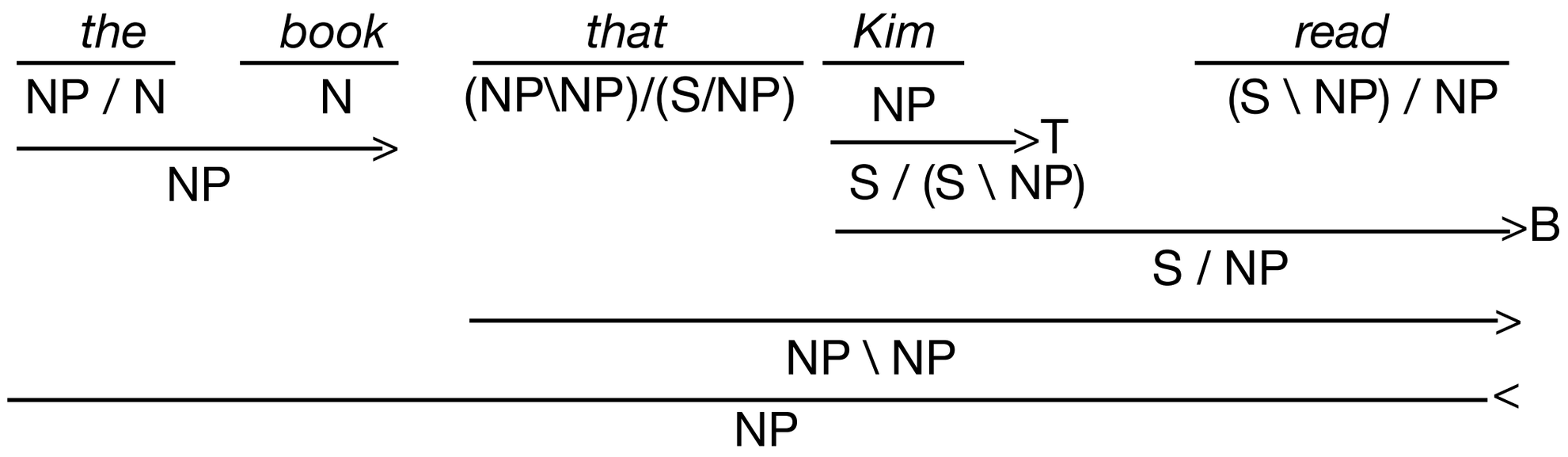
Mary	:=	NP
musicals	:=	NP
likes	:=	((S \ NP) / NP)



Note:

- Can process input tokens left-to-right.
- The (S / NP) cartegory does not correspond to a traditional English constituent.

Long-Distance Dependencies in CCG



Note:

- Grammar needs only one lexical entry for read!
- Type raising allows us to combine *Kim* with *read* before the object NP is attached. The missing NP is represented n the new category S / NP.

CCG Observations

- CCG has become really popular. CCGBank (Hockenmaier & Steedman 2007) is an automatic conversion of the Penn Treebank to CCG.

- CCG is more expressive than CFG ("mildly context sensitive"). Weak expressivity is the same as TAG.

- Parsing is more expensive (can be done in $O(N^6)$).

    - Efficient greedy approximation Algorithms exist (e.g. Lewis 2014).

Tree Substitution Grammars

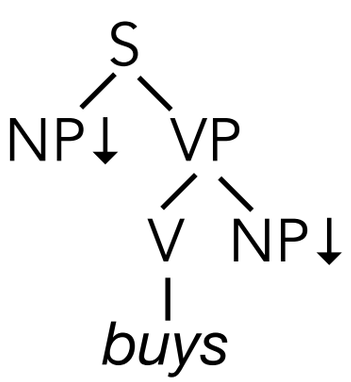
- The grammar formalism discussed so far (CFG, FG, CCG) operate on strings.
  - Recognition problem: determine if a sentence is grammatical or not.
  - Parse tree as a "side-produce"

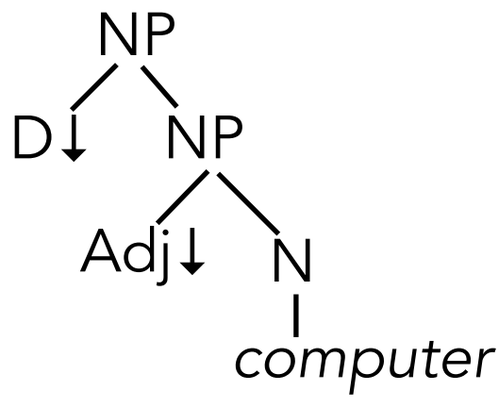
- Tree substitution grammars operate on trees.

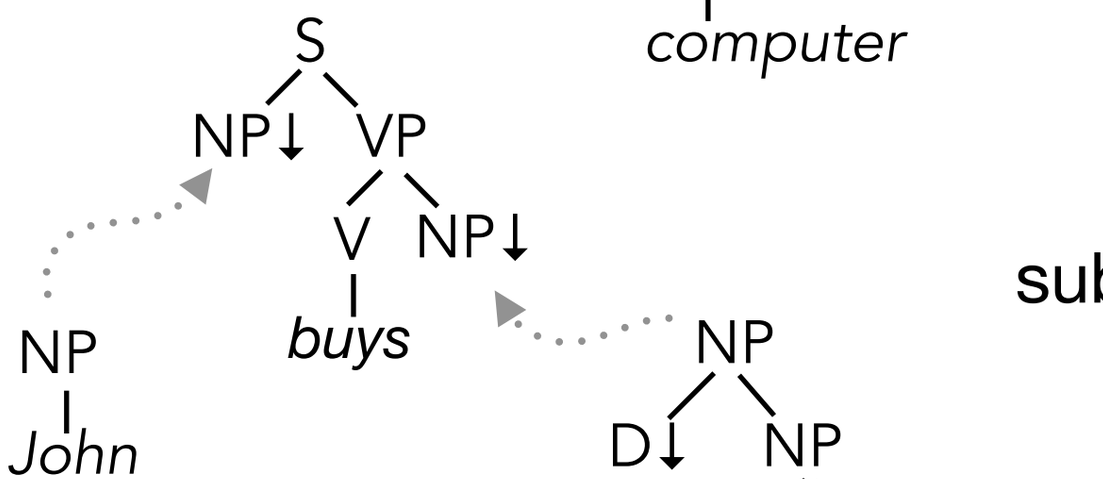
- Just like CFG, but rule right-hand sides are *elementary trees (tree fragments)* instead of strings.

Tree Substitution Grammars

Elementary Trees



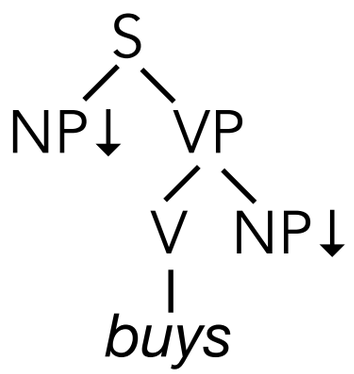


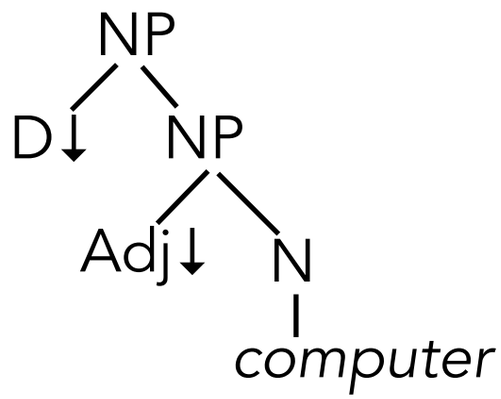


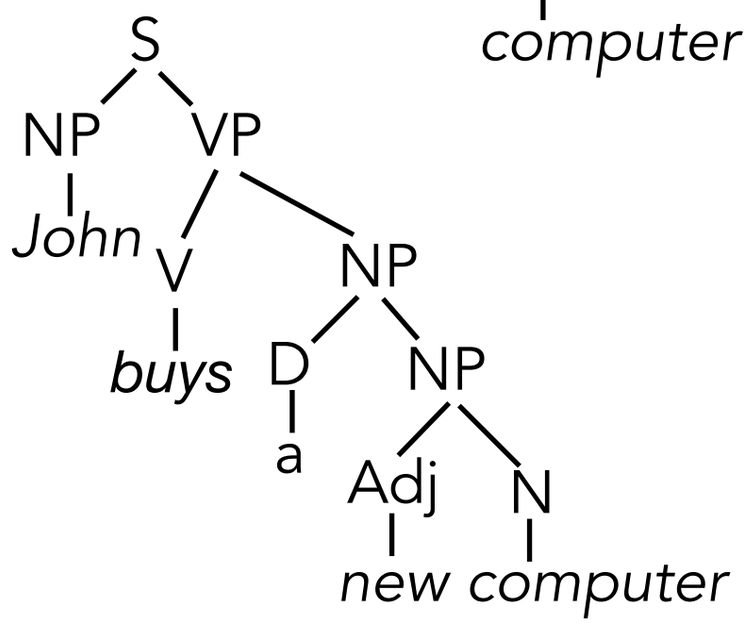
substitution operation

Tree Substitution Grammars

Elementary Trees







Note:

- No production rules.
- Arguments are in the local domain of their predicate.
- Good idea, but sadly not more expressive than CFG.

Tree Adjoining Grammars

- A Tree Adjoining Grammar is defined by
  - Set of **terminal symbols** Σ.
  - Set of **non-terminal symbols** N.
  - A **start symbol** $S \in N$.
  - I: a finite set of **initial trees.**
  - A**:** a finite set of **auxiliary trees.**

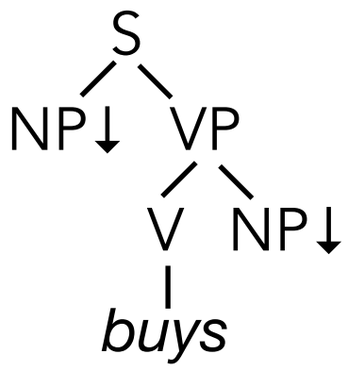
Two types of elementary trees!

We will consider **Lexicalized Tree Adjoining Grammars (LTAG):** Each tree has at least one terminal "anchor".

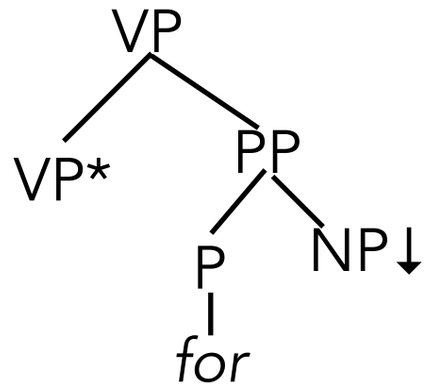
Joshi (1985), Joshi & Schabes (1997)

Adjunction

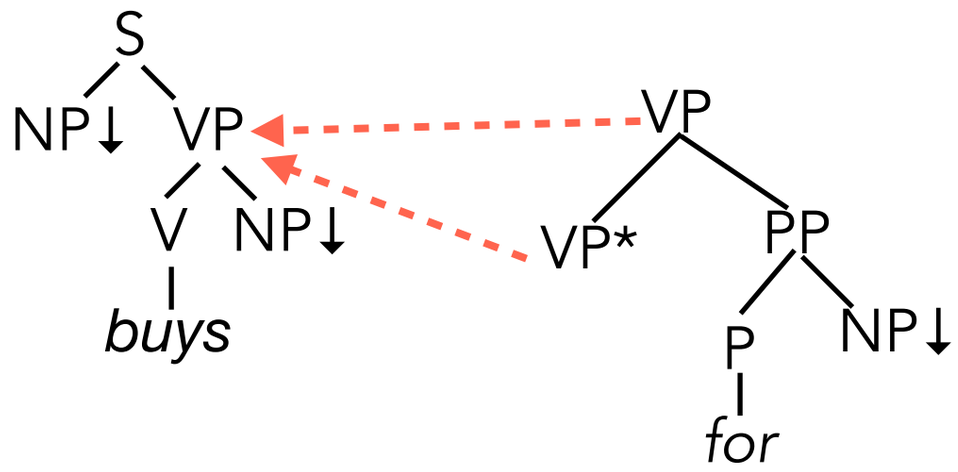
Initial Tree

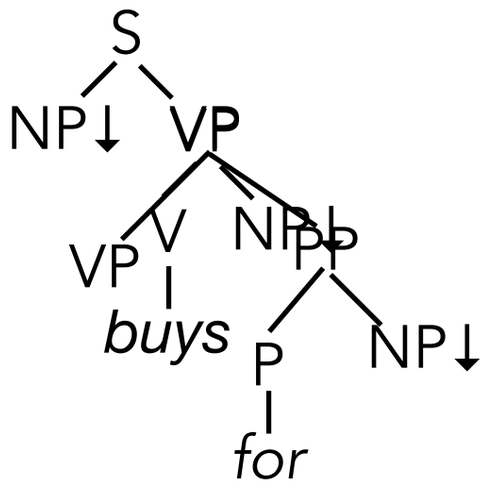


Auxiliary Tree



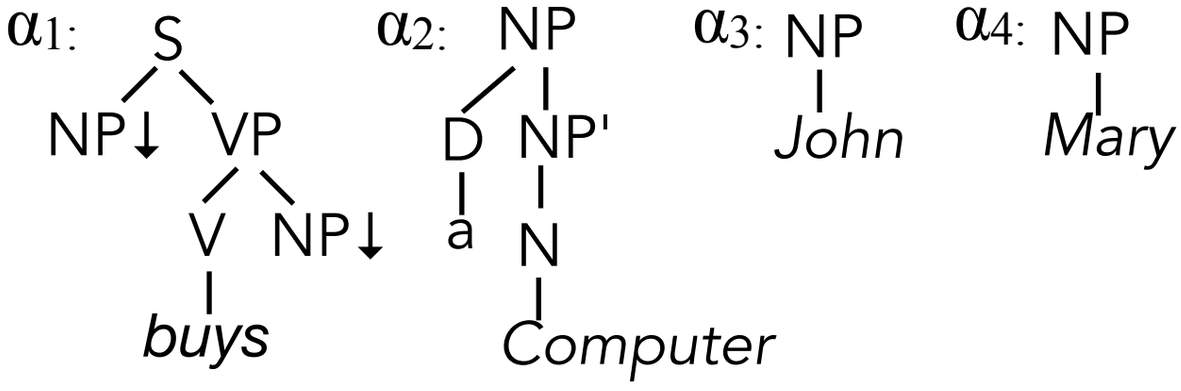
- Auxiliary trees have a special **foot node** (marked with *)
- Adjunction may take place on any internal node in the host tree (the target node)
- A copy of the auxiliary tree is inserted.
  The children under the target node are attached to the foot node.
  The root of the auxiliary tree is fused with the target node.



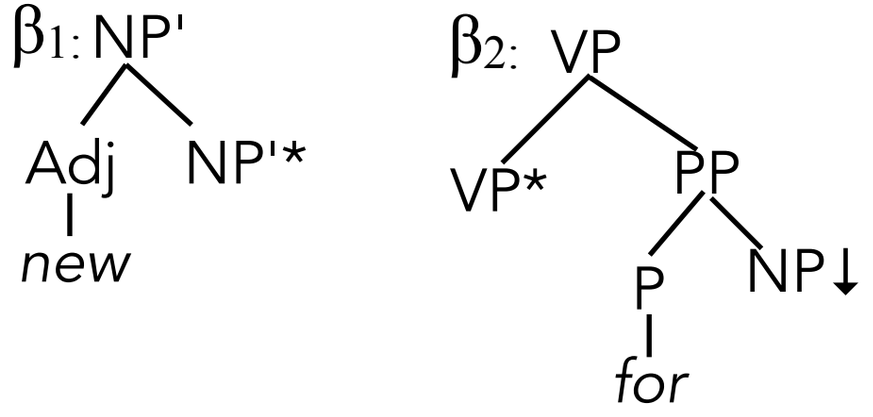


LTAG Example

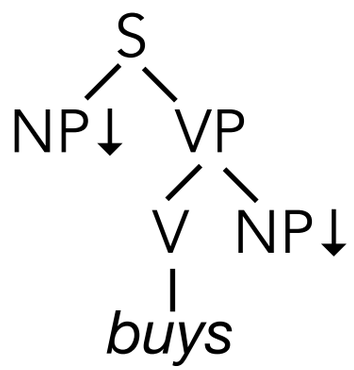
Initial Trees



Auxiliary Trees



Derived Tree

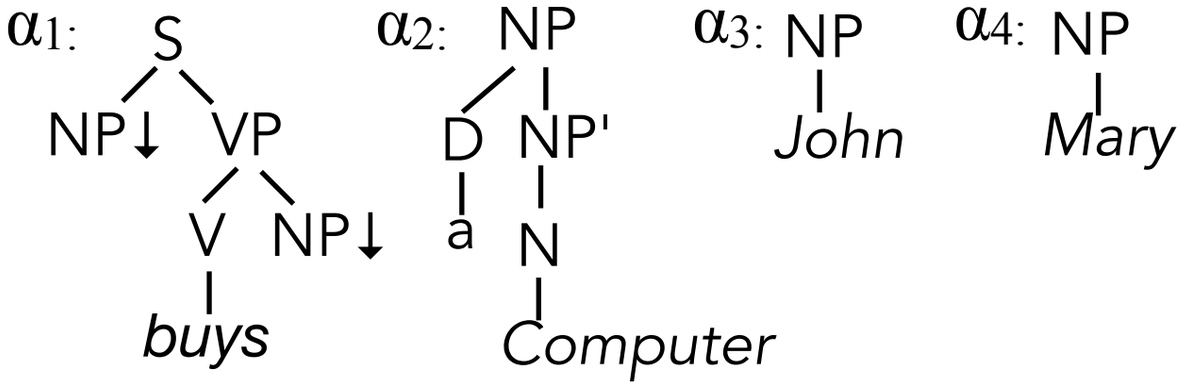


Derivation Tree

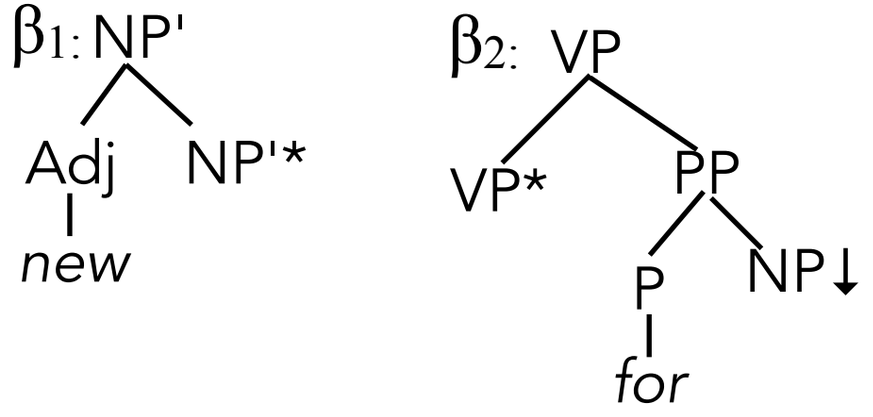
α_1

LTAG Example

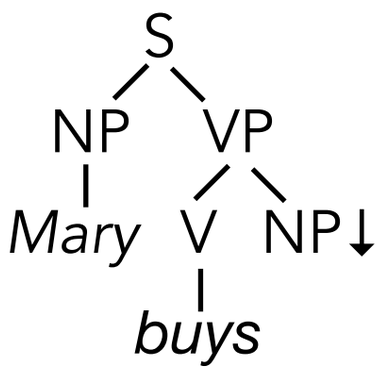
Initial Trees



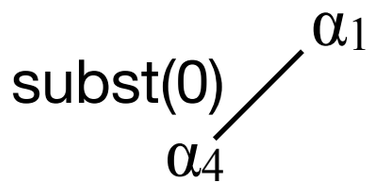
Auxiliary Trees



Derived Tree

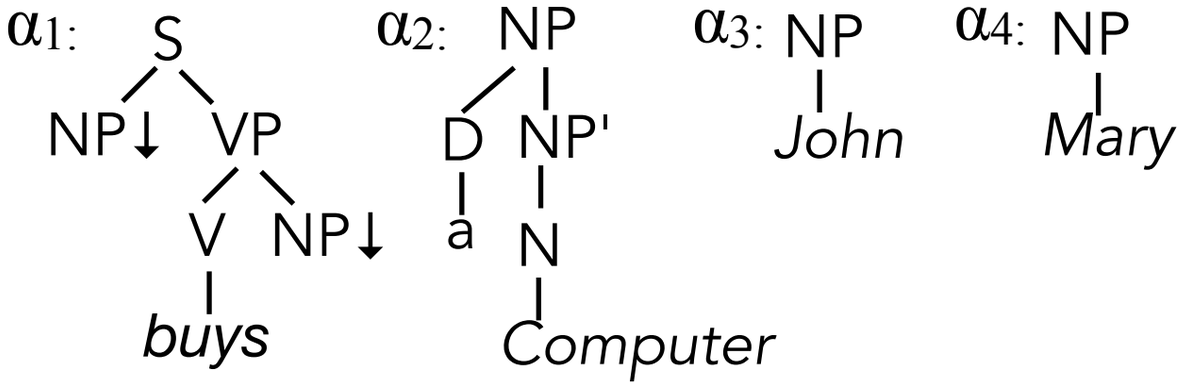


Derivation Tree

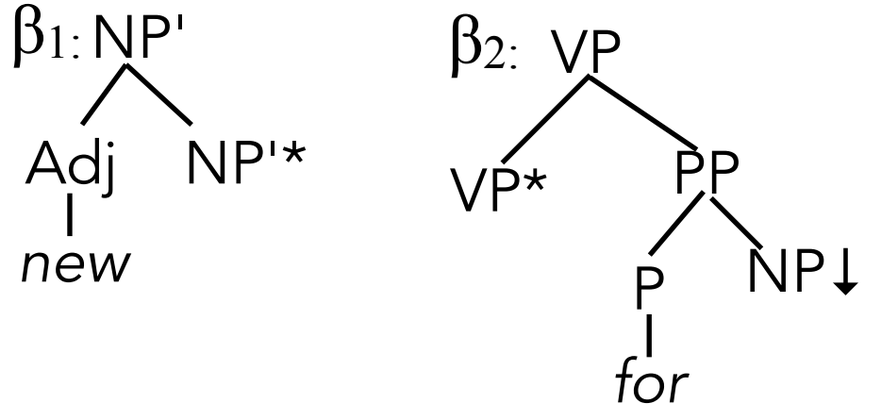


LTAG Example

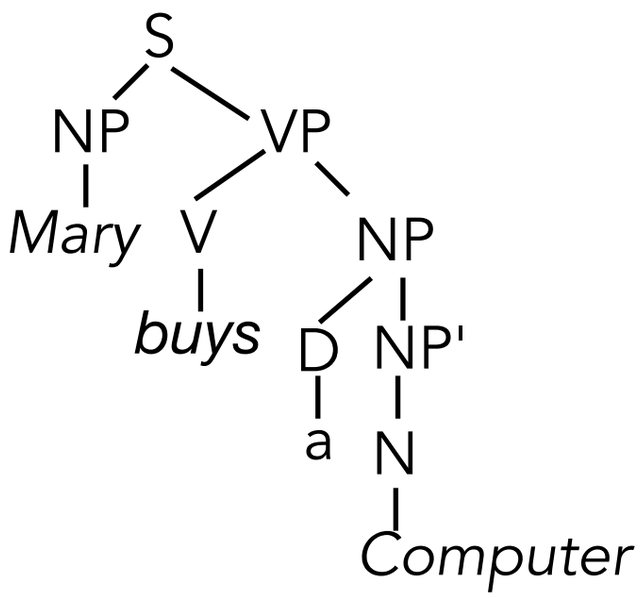
Initial Trees



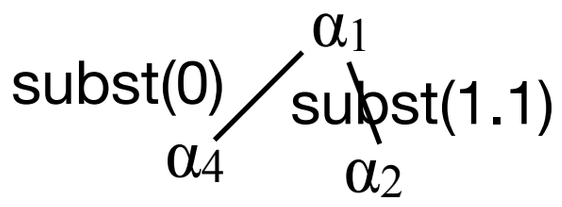
Auxiliary Trees



Derived Tree

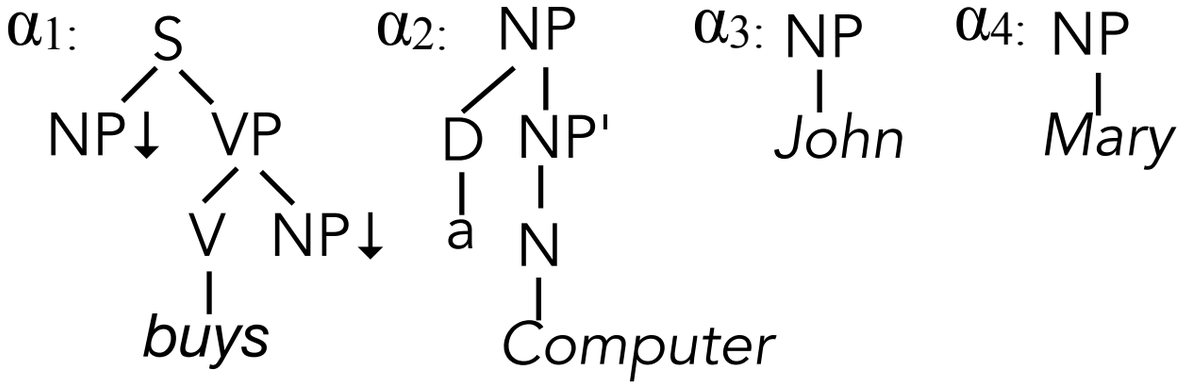


Derivation Tree

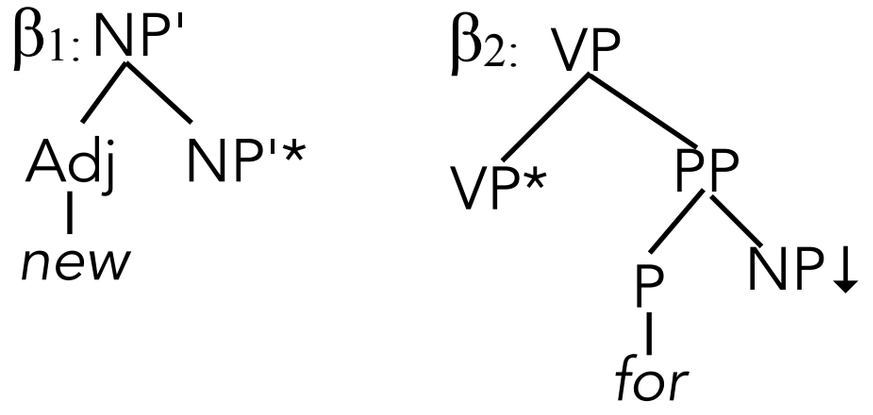


LTAG Example

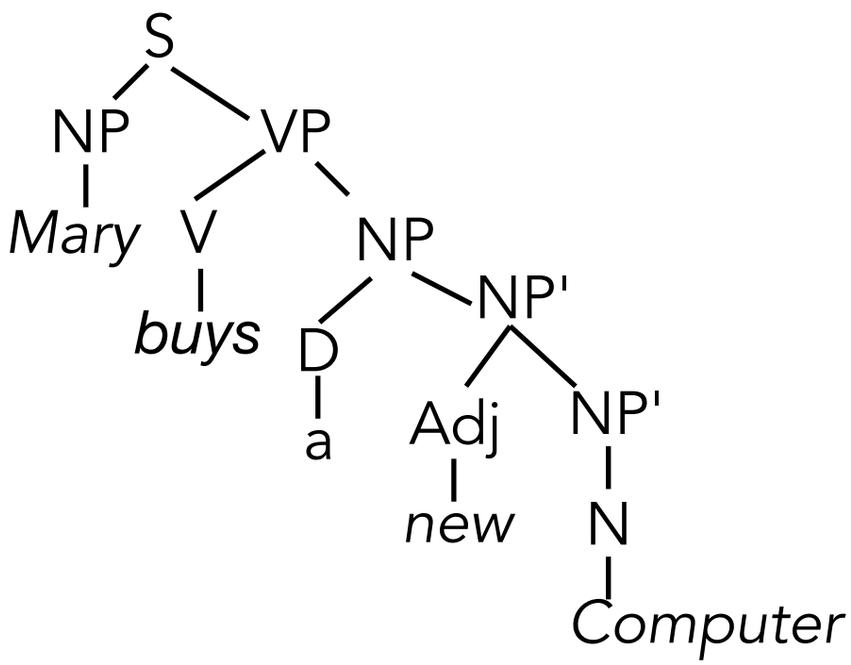
Initial Trees



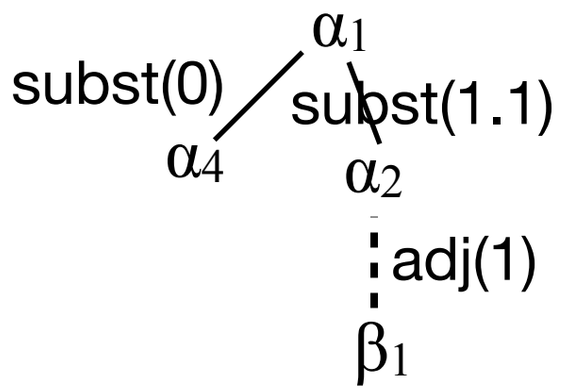
Auxiliary Trees



Derived Tree

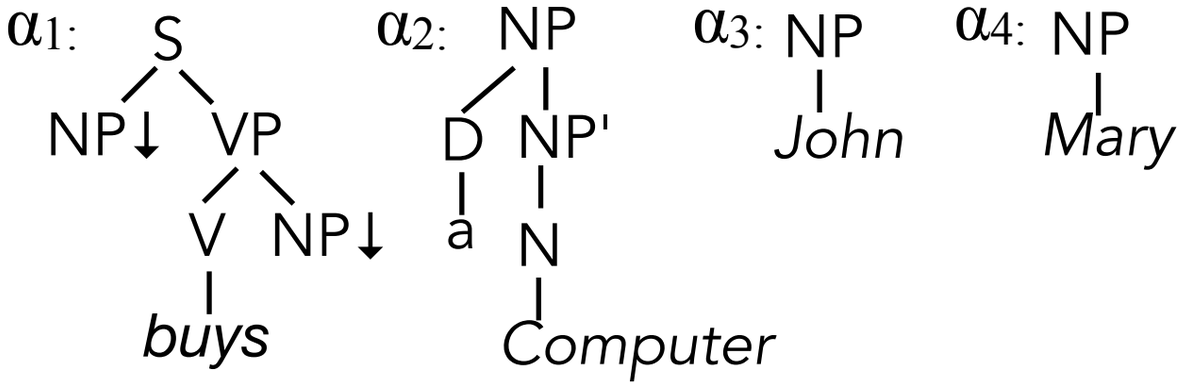


Derivation Tree

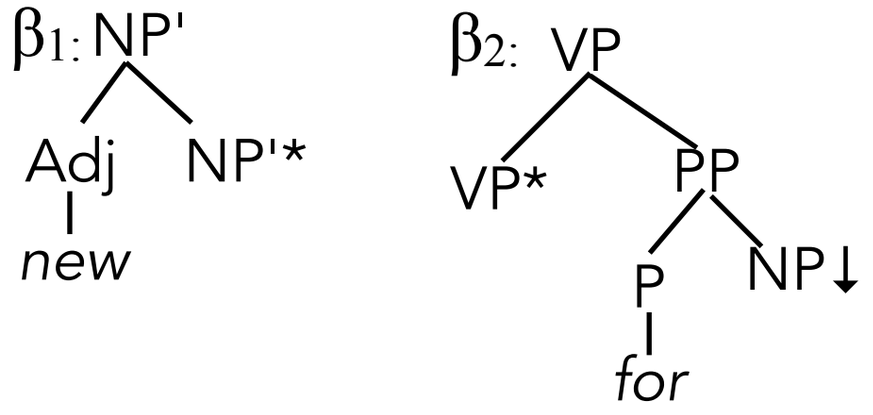


LTAG Example

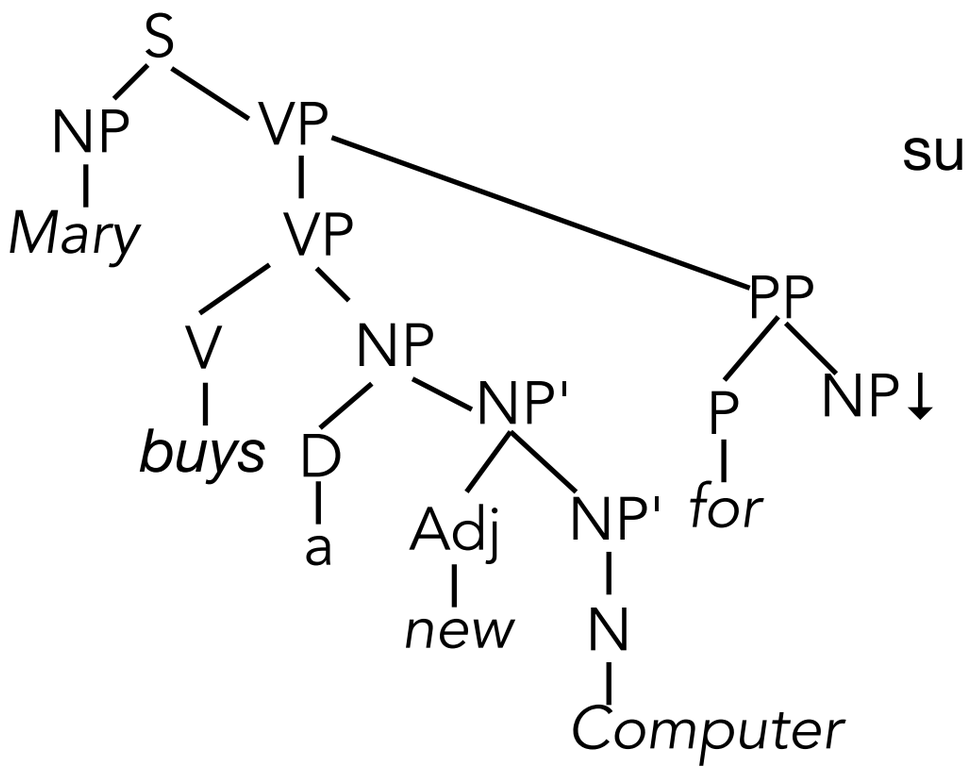
Initial Trees



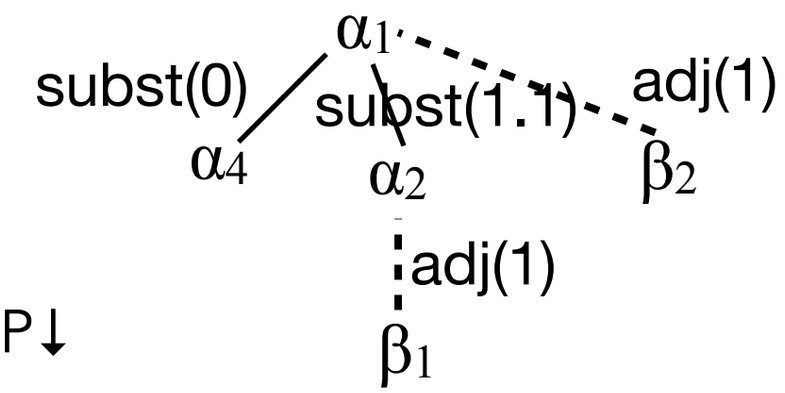
Auxiliary Trees



Derived Tree

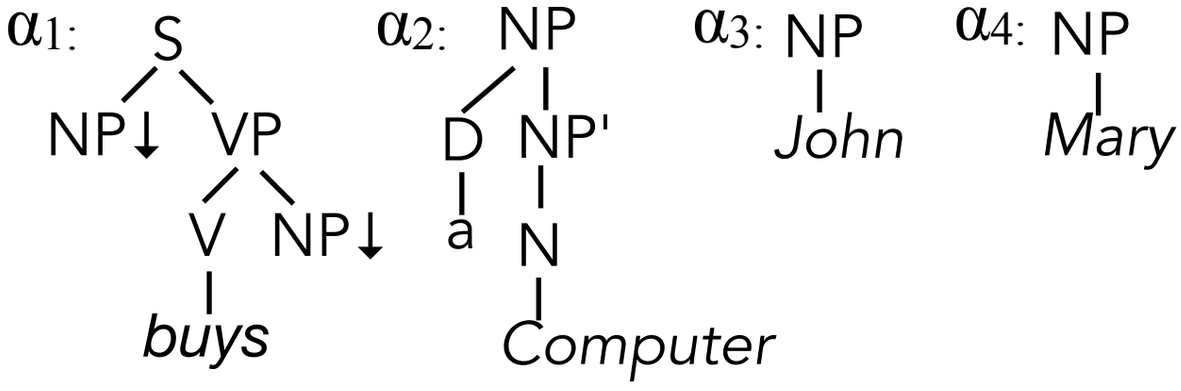


Derivation Tree

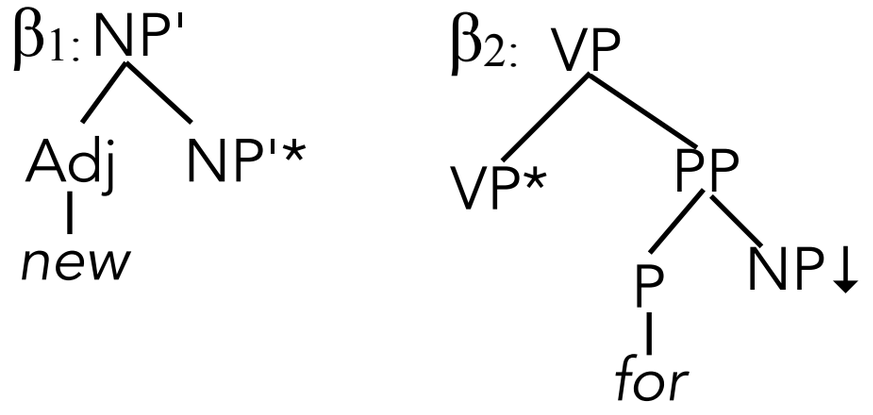


LTAG Example

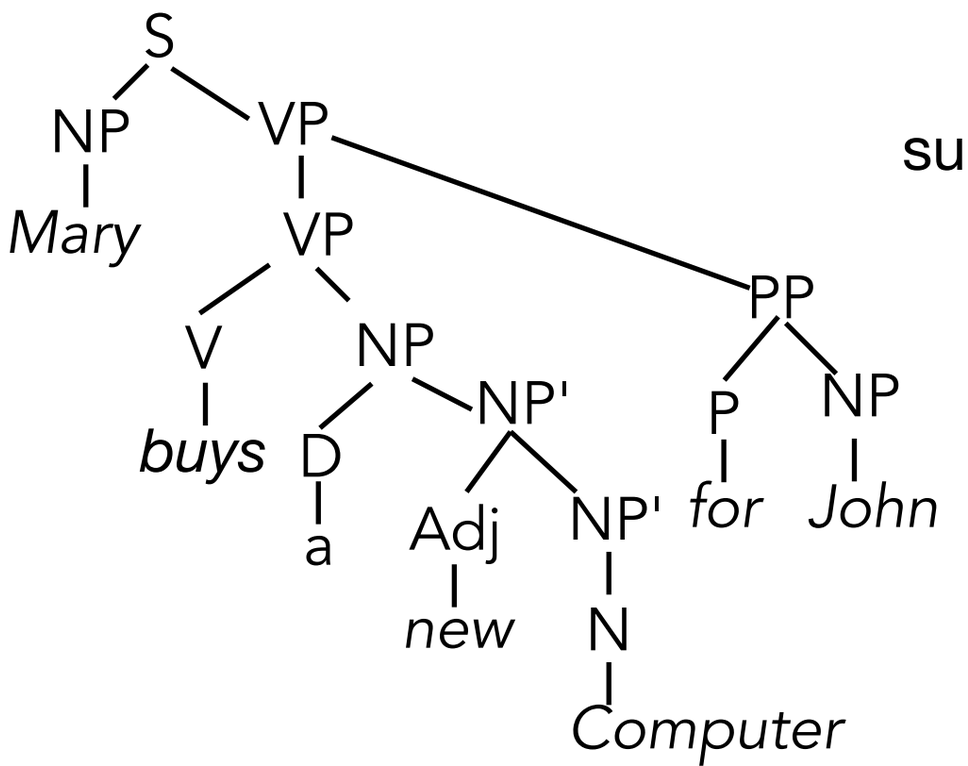
Initial Trees



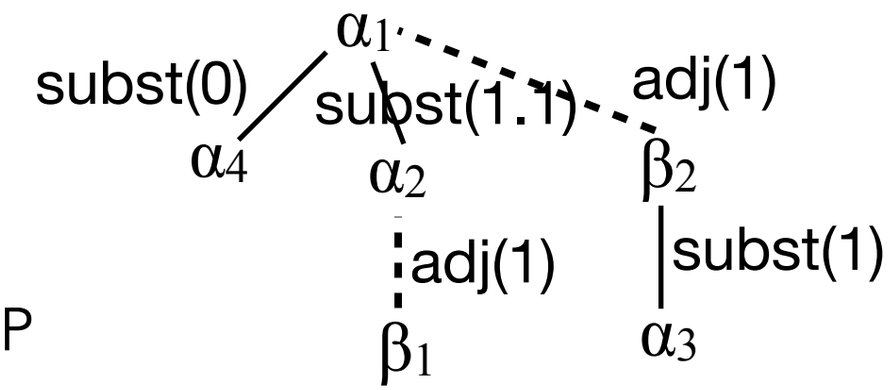
Auxiliary Trees



Derived Tree

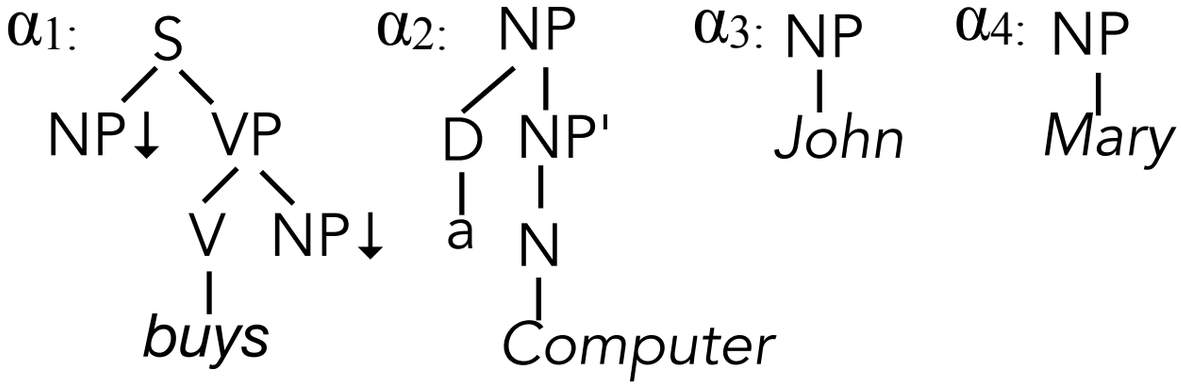


Derivation Tree

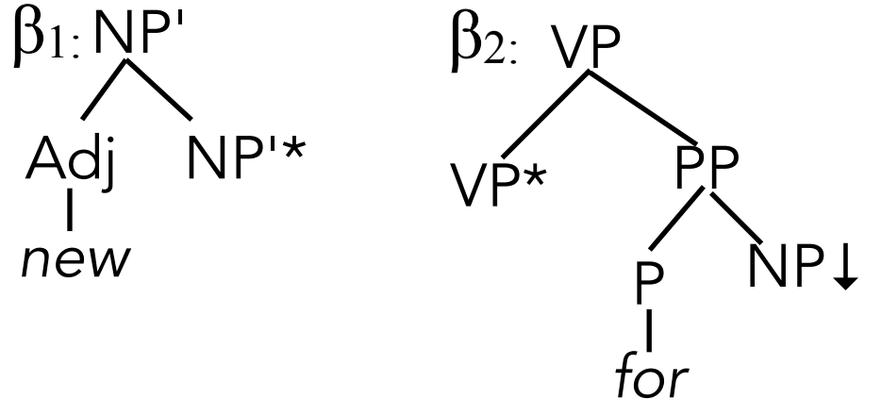


LTAG Example

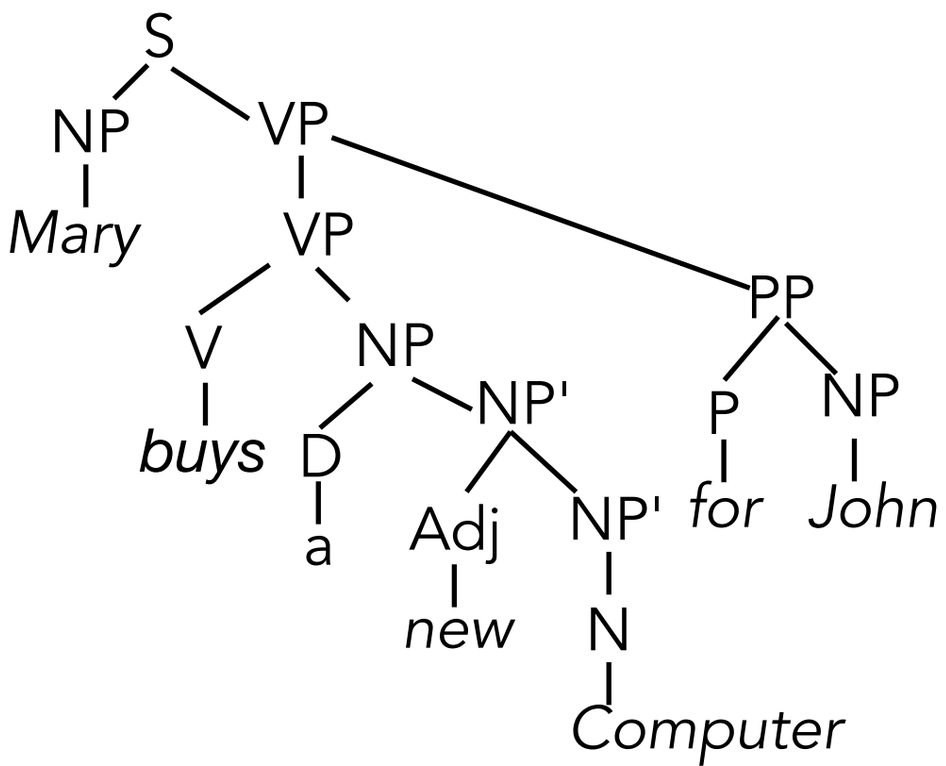
Initial Trees



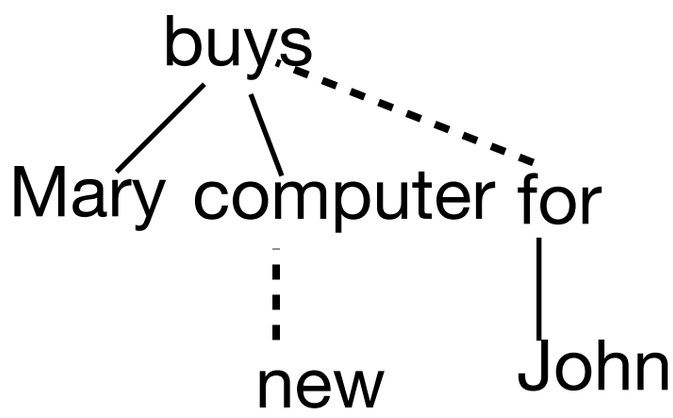
Auxiliary Trees



Derived Tree

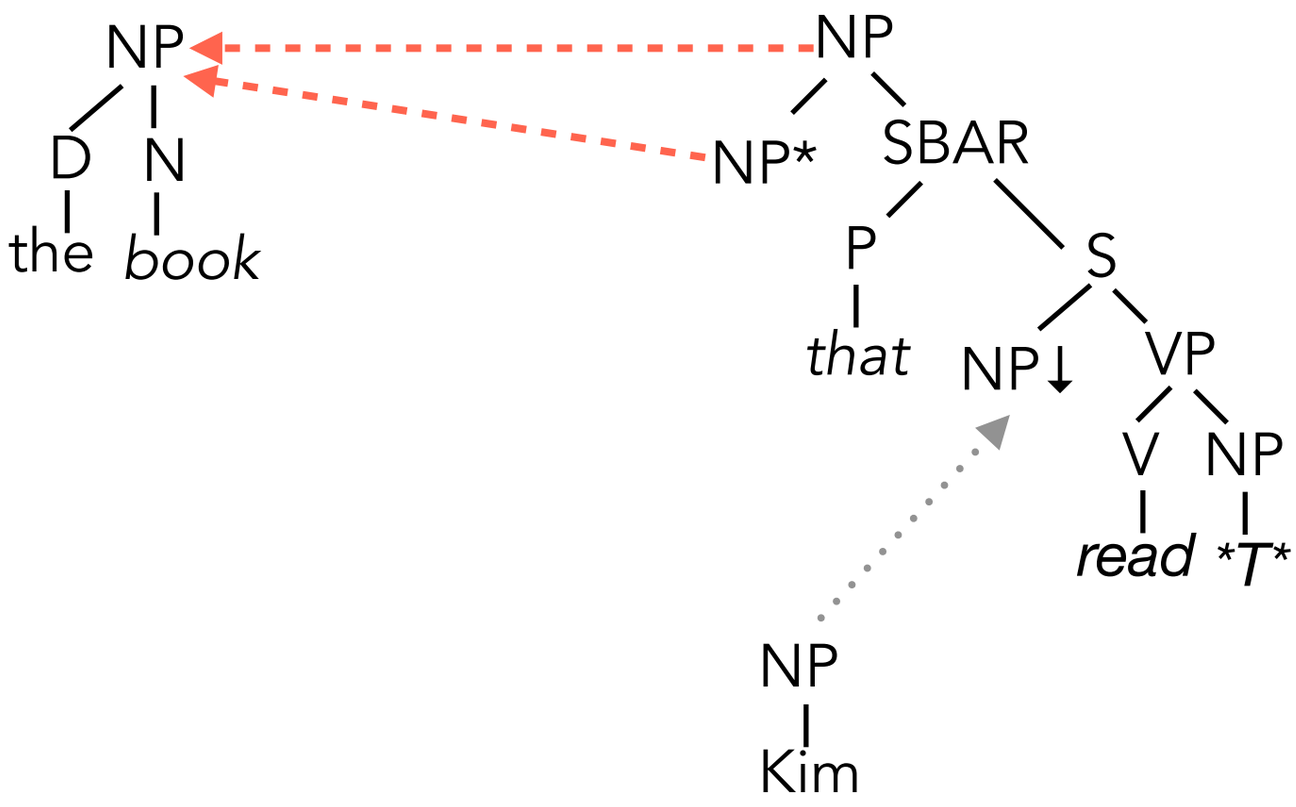


Derivation Tree

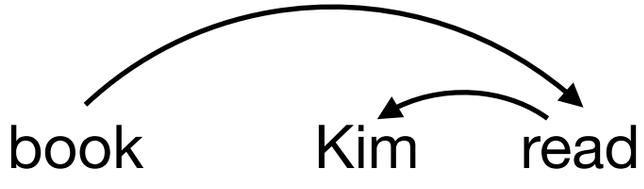


derivation tree looks like a dependency tree!

Long-Distance Dependencies in LTAG



Correct dependency structure:



Tree Adjoining Grammars - Summary and Observations

- LTAG derivation trees resemble dependency trees (arguments and adjuncts are local, "extended domain of locality").

- TAG string languages: Same set of "mildly context sensitive" languages as CCG, parseable in $O(N^6)$.
  [However, not *strongly* equivalent to CCG, e.g. not the same set of derivation structures]

- Many extensions (multi-component TAG etc.)

- "Supertagging" (Bangalore and Joshi, 1999): Assign an elementary tree to each word, then parse.