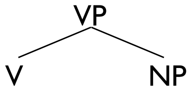
CAS LX 522
Syntax I

6

Merge, feature checking
(3.6-4.2)

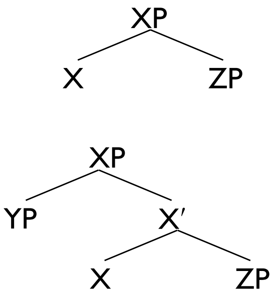
Node labeling conventions

- When we Merge two objects, the features of one of them projects to become the features of the new object.
- The label for new node comes in two pieces:
  - The category (projected from the head)
  - The projection "level":
    - P = maximal projection
    - ° or nothing = minimal projection
    - ′ = intermediate projection
- An XP is any node that does not project its features up.
- An X° (or X) node comes from the lexicon.

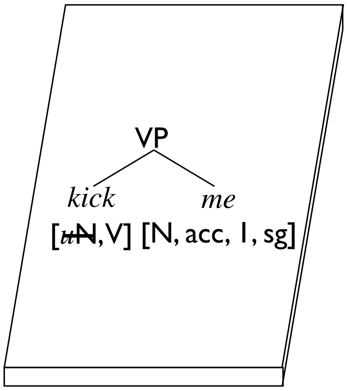


Maximal v. Minimal v. Intermediate

- Notice that whenever you Merge two things, the result is going to be a maximal projection. An "XP".
- But if in the next step if projects when you Merge it with something, that same node is now an intermediate projection.



Features and checking



- When we combine two things with Merge and check an uninterpretable feature, we cross it out.
- For simplicity, we can simply write the features under the head, and cross them out there.
  - This is as opposed to copying all but the checked feature and into a feature specification of the VP node.
  - This is just about how we write it down, it is the same system either way.

Adjuncts

- *Pat put the book.
- Pat put the book on the shelf.
- Pat put the book on the shelf dramatically.
- Pat put the book on the shelf dramatically on Tuesday.
- Pat put the book on the shelf dramatically on Tuesday before several witnesses.

- Some things are required. Some things are not.
  - **Arguments** get θ-roles and are **required**.
  - **Adjuncts** are modificational and are **optional**.

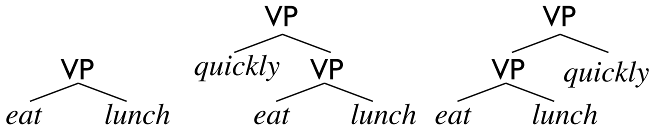
Adjuncts and distribution

- Adjuncts are relatively "transparent"— having an adjunct does not seem to change the distributional characteristics.
  - Pat wants to eat lunch (quickly).
  - Pat wants to dine.
  - *I like to draw eat lunch (quickly).
  - I like to draw (happy) elephants.
  - *Pat wants to (happy) elephants.

- Idea: A verb (phrase) with an adjunct is still a verb (phrase), just as if it didn't have an adjunct.

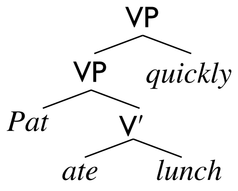
Adjoin

- The operations Merge and Adjoin are two different ways to combine two objects from the workbench.
- Merge takes two objects and creates a new object (with the label/features inherited from one of them).
- Adjoin attaches one object to the top of another one.
  - The linear order of adjuncts does not appear to be set parametrically, so they can either before or after the object they attach to.



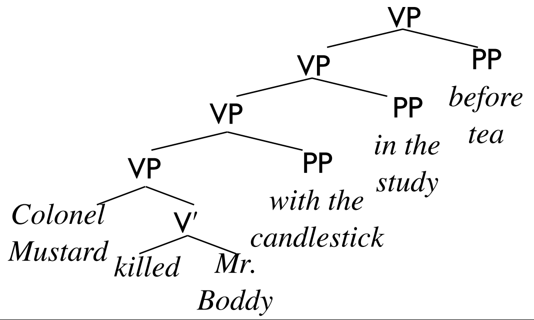
The luxury of adjunction

- We will also assume that Adjoin only applies to maximal projections.
- That is: If a syntactic object still has a selectional feature, Adjoin cannot attach something to it. Merge must happen first. Once all of the things that *need* to happen are taken care of, *then* you have the luxury of adjunction.



The luxury of adjunction

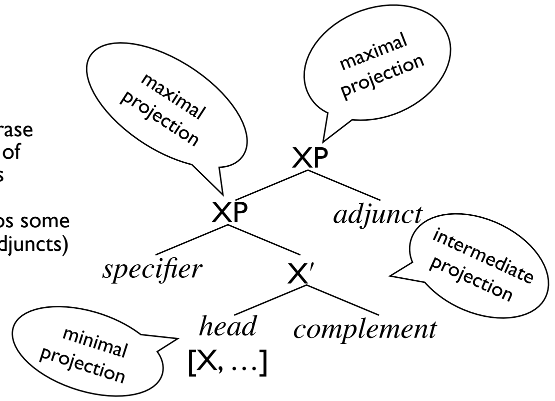
- Any number of adjuncts can be added, generally in any order. Adjuncts come in many different categories— "adjunct" is not a category, but rather a structural description.



A phrase

- So, a full phrase can have all of these pieces

  (plus perhaps some additional adjuncts)



Complements vs. adjuncts

- PPs seem to be freely reorderable— when adjuncts.
  - I ate lunch on Tuesday at Subway with Pat
  - I ate lunch on Tuesday with Pat at Subway
  - I ate lunch with Pat on Tuesday at Subway
  - I ate lunch on Tuesday with Pat at Subway
- But consider *glance at Chris*.
  - I glanced at Chris on Tuesday
  - *I glanced on Tuesday at Chris
- Ok: Why?

Mary saw him

- A pronoun like *him* refers to somebody in (our mental model of) the world.
- A pronoun refers to somebody or something that's been part of the conversation, or that you are pointing at.
- When you hear a pronoun and want to interpret it, you have to resolve its reference.

John arrived. Mary saw him.

- Here, *him* is likely to refer to John.
- Though we could be pointing at Bill, in which case *him* refers to Bill.
- The person who *hears* this has to figure it out.
- The person who *says* this knows who they meant.
- And had the grammar that generated the sentence.

Indices

- To describe what the speaker intended (that is, which sentence the speaker actually used), we use an *index* on each referent.

1) John$_i$ arrived. Mary$_j$ saw him$_i$.

2) John$_i$ arrived. Mary$_j$ saw him$_k$.

- The index represents what you are “pointing at” (perhaps just mentally).
- Two noun phrases that share an index *necessarily* share the same reference. They are coreferential.

Seeing him in the mirror

- Regard: Ike$_i$, Jim$_j$, Kristin$_k$.

1) There’s Ike$_i$. Kristin$_k$ saw him$_j$ in the mirror.

2) There’s Jim$_j$. Kristin$_k$ saw him$_j$ in the mirror.

3) There’s Ike$_i$. Jim$_j$ saw him$_i$ in the mirror.

4) There’s Jim$_j$. *Jim$_j$ saw him$_j$ in the mirror.

- What’s wrong with that last one?

Seeing himself in the mirror

- Right, ok. Jim$_j$ saw *himself*$_j$ in the mirror.
- For some reason, when *Jim* is the subject and *him* is an object, *him* can’t refer to *Jim*. Furthermore:

1) Jim$_j$’s father$_k$ saw him$_{i/j/*k}$ in the mirror.

2) Jim$_j$’s father$_k$ saw himself$_{k/*j/*i}$ in the mirror.

3) Jim$_j$’s father$_k$ said that Mary$_m$ saw him$_{i/j/k}$ in the mirror.

4) Mary$_m$ introduced Jim$_j$ to him$_{i/*j}$.

5) Mary$_m$ introduced Jim$_j$’s father$_k$ to him$_{i/j/*k}$.

Binding Theory

- **Binding Theory** consists of three Principles that govern the allowed distribution of NPs.

- Pronouns: *he, her, it, she, ...*
- Anaphors: *himself, herself, itself, ...*
- R-expressions: *Pat, the student, ...*

R-expressions and anaphors

- R-expressions are NPs like *Pat*, or *the professor*, or *an unlucky farmer*, which get their meaning by referring to something in the world. Most NPs are like this.

- An anaphor does *not* get its meaning from something in the world—it depends on something else in the sentence.

1) John saw himself in the mirror.

2) Mary bought herself a sandwich.

Pronouns

- A pronoun is similar to an anaphor in that it doesn't refer to something in the world but gets its reference from somewhere else.

1) John told Mary that he likes pizza.

2) Mary wondered if she agreed.

- …but it doesn't *need* to be something in the sentence.

1) Mary concluded that he was crazy.

Constraints on coreference

1) John$_i$ saw himself$_i$.

2) *Himself$_i$ saw John$_i$.

3) *John$_i$'s mother saw himself$_i$.

- It is impossible to assign the same referent to *John* and *himself* in the (2) and (3). What is different between the good and bad sentences?

John's mother

- *John's mother* is an NP.

1) [John's mother]$_i$ saw herself$_i$.

2) She saw John.

- But it's an NP that is made up of smaller pieces (*John's* and *mother*).
- So what is the internal structure of the NP *John's mother*?

[$_{NP}$ John's mother]

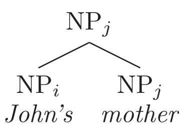
- Remember that pronouns come in three distinguishable forms (in English):

I, he, she	nominative
Me, him, her	accusative
My, his, her	genitive

- The genitive case forms seem to have pretty much the same kind of "possessive" meaning that *John's* does.
- So, let's suppose that *John's* is the genitive case form of *John*.

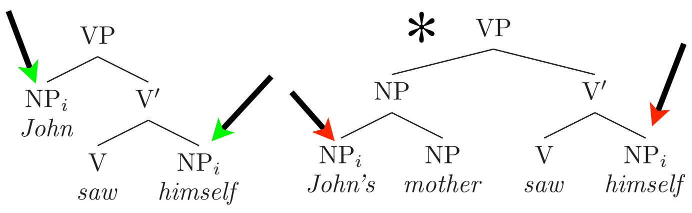
[$_{NP}$ John's mother]

- Another point about *John's mother* is that it seems that the head should be *mother*.
- *John's* sort of modifies *mother*.
- Sort of like an adjective does… sort of like an adverb does for a verb…
- Let's suppose that *John's* is just adjoined to the NP *mother*.
  - Only for now! To be revised in ch. 7.
  - This is kind of hard to draw clearly.



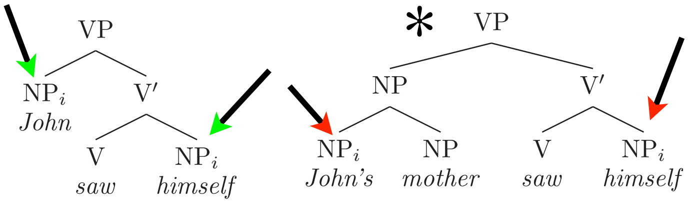
Command domains

- What is the difference between the relationship between *John* and *himself* in the first case and in the second case?

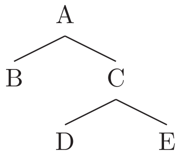


Command domains

- We think of the position that *John* occupies in in the first tree as being a position from which it "commands" the rest of the tree. It is hierarchically superior in a particular way. (Really, "non-inferior")



Tree relations



- A node X c-commands its sisters and the nodes dominated by its sisters.
  - B c-commands C, D, E.
  - D c-commands E.
  - E c-commands D.
  - C c-commands B.
  - A c-commands nothing.