

Python Course: Lecture 17

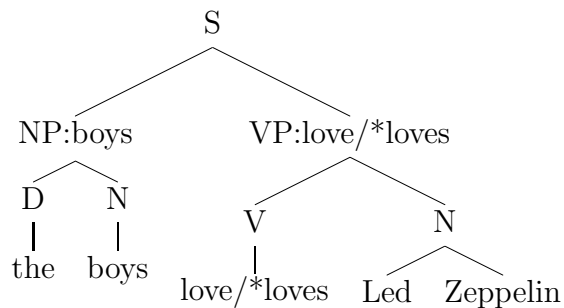
February 13, 2006

1 Syntactic Effects in Terms of Trees

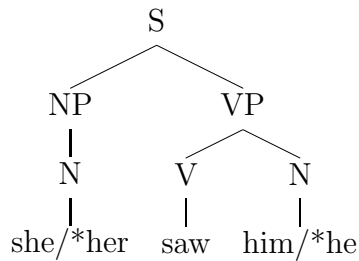
We can go back and describe all the syntactic phenomena from the last lecture more precisely using the language of trees.

It would be difficult to account for the same linguistic phenomena in terms of just strings of words.

- *Agreement*
 - The subject is the NP dominated by S.
 - The main verb is the VP dominated by S.
 - The number of the NP head must agree with the number of the VP head.



- *Case*
 - The direct object is the NP dominated by the main verb.
 - Nominative case pronouns must appear in subject position.
 - Accusative case pronouns must appear in direct object position.



- *Subcategorization*

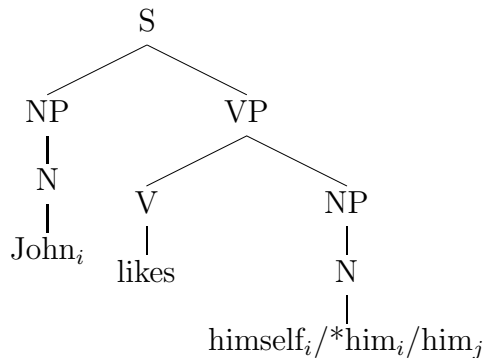
Different verbs must have different numbers and kinds of objects. For example, *kill* and *devour* take a single NP object, whereas *eat* takes an optional NP object. The verb *give* takes an NP object followed by a PP headed by *to*, or two NP objects.

- *Anaphora*

Define a relationship called *c-command*. α c-commands β iff

- α does not dominate β
- β does not dominate α
- A sibling of α dominates β

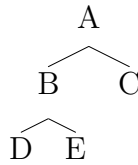
Reflexives pronouns such as *himself* and *herself* must be c-commanded by their antecedents and referential pronouns such as *him* and *her* may not be c-commanded by their antecedents.¹



2 Tree Geometry

We need to define the following concepts of tree geometry in order to be able to express the above rules formally. For any pair of nodes in a tree, let the following relationships be defined.

¹The story is more complicated, but this is a good rough cut of the English anaphora rule.



- *Dominance*

Dominates is the transitive closure of the *parent of* relationship. In the above tree, A dominates B, C, D, and E. B dominates D and E. C does not dominate anything.

- *Precedence*

The *precedes* relationship is defined for all nodes that do not have a dominance relationship. In the above tree, B precedes C. D and E precedes C.

More complex relationships can be defined in terms of these basic relationships.

3 Tree Data Structure

- How do you build a tree data structure?
- One implementation: make each node in the tree a **Node** class.
- Each **Node** instance will have a parent (pointing to its parent node or **None**), a children list of nodes, and a content property containing its label.