Constituent Structure and Structural Relations: Overview

Constituent Structure and Structural Relations

Overview

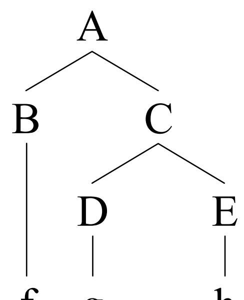
Tree Structure
(AKA Phrase-Structure Tree, Phrase Marker)

Tree Structure
(AKA Phrase-Structure Tree, Phrase Marker)

A phrase-structure tree is a **diagram**, which exhibits the **syntactic structure** of linguistic expressions. Its basic elements are points and lines**.** The points are called **nodes** and the lines are called the **branches** of the tree. The nodes in a tree are **labeled** (e.g., NP, IP, V, etc.,) by names of syntactic **categories** (except for nodes at the bottom of a tree, but wait). The branches connect pairs of labeled nodes. A branch represents a **simple structural relation** between the pair of labeled nodes it connects. More **complex structural relations** may be represented in a tree by **two or more** branch**es** in such away that a node at one end of a branch is **indirectly** related to a node at the end of **another branch** (as in the case of some precedence relations, wait now, see below).
Any pair of nodes contained in a tree will be related by one of two different types of elementary relation, namely <u>either</u> by **dominance** <u>or</u> by **precedence** (which mutually exclude each other, see below). Dominance is the simpler of the two. Precedence is a little more complex (but still elementary, see above and wait now, then see below). A tree structure has only **one top node**.

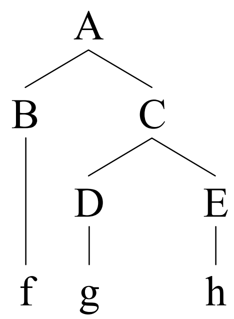
For example:

(1)



Each node in the tree in (1) carries a label. The tree contains the following labeled nodes: A, B, C, D, E, f, g, and h.
The nodes <u>at the bottom of a complete</u> tree are **terminal nodes:** f, g, & h in (1).
Non-empty terminal nodes are labeled with a **lexical item** (a word or other **morpheme**): f, g, and h in
All other nodes are **non-terminal**: A, B, C, D, and E in (1).
Non-terminal nodes carry **category labels**: A, B, C, D, and E in (1).

(1)



The node labeled A **dominates** all other nodes. This is the top node.
Node C dominates nodes D and E, but neither D nor E dominates C.
Node B **precedes** nodes D and E, as well as g and h.
Nodes can be **branching** or non-branching. Node A branches into nodes B and C; node C branches into nodes D and E. Node B is a **non-branching** node.

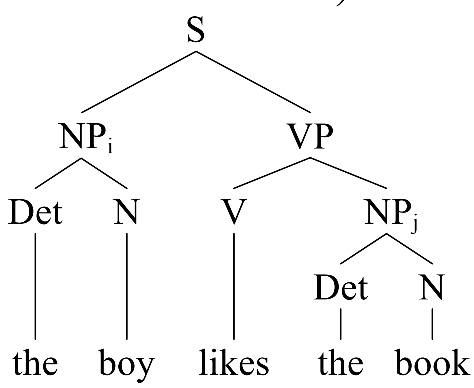
Mothers, daughters, and sisters

Two nodes B and C are **sisters** iff there is a node A (their **mother**) which immediately dominates both B and C. B and C are both **daughters** of A.
A category X **immediately dominates** a category Y when X dominates Y and every other category dominating Y dominates X. In (1), A immediately dominates B and C, and C immediately dominates D and E, but A does not immediately dominate D or E, because C intervenes.

Dominance

The relation between a category and its constituents: A category **dominates** all of its **constituents**. A node A dominates another node B iff A is higher up the tree than B such that you can trace a line from A to B going only downwards. When tracing downwards, you may (non-immediate dominance) or may not (immediate dominance) cross one or more intervening nodes between A and B.

(2)



The category VP in (1) dominates the categories V, NP_j, Det, and N.

VP **immediately** dominates NP_j, but it does not immediately dominate N since another category, NP, intervenes between VP and N.

V and NP_j are **immediate constituents** of VP.

Immediate dominance: A category X immediately dominates a category Y when X dominates Y and every other category dominating Y dominates X.
Immediate constituent: When X immediately dominates Y, then Y is an immediate constituent of X.

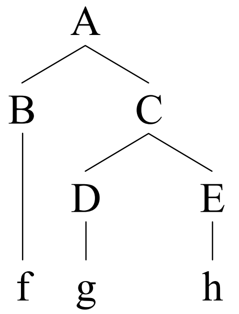
The Dominance relation has the following logical properties:

- irreflexivity: A node does not dominate itself.
- asymmetry: If A dominates B, B does not dominate A.
- transitivity: If A dominates B, and B dominates C, then A dominates C.

Precedence (or linear order of constituents)

Precedence is a binary relation between nodes in a tree. A **precedes** B iff A is to the left of B and A does not dominate B and B does not dominate A.

(1)



Node B precedes nodes C, D and E, as well as the terminal nodes g and h.

B does not precede f, since it dominates f. C, D and E do not precede B, since they are to the right of B. Also, A does not precede any of the other nodes since it dominates all of them. Node D **immediately precedes** node E: there is no intervening node between D and E, i.e. there is no node X such that X is preceded by D and precedes E. Node B precedes E, but does not immediately precede it, since there is an intervening node: D, which precedes E and is preceded by B. Node C does not count as an intervening node between B and E: although it is preceded by B, it does not precede E, since it dominates it.

C-command

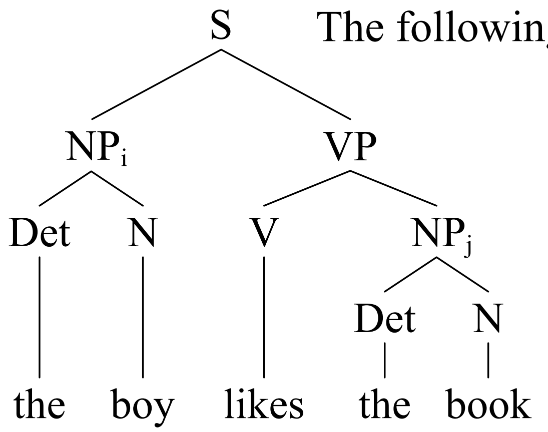
C-command = constituent command. C-command is a binary relation between nodes in a tree structure

Strict c-command:

A category α c-commands a category β iff the category that **immediately dominates** α also dominates β, and neither α nor β dominates the other.

Or: A category α c-commands a category β if and only if the **first branching node** that dominates α also dominates β, and neither dominates the other.

(2)



The following (strict) c-command relations hold in (2):

a. NP$_i$ c-commands NPj
b. NPj does not c-command NPi
c. NPj c-commands V
d. V c-commands NPj
e. NPj c-commands V
f. (so) V and NP$_j$ c-command each other
g. V does not c-command NP$_i$

etc.

C-command domain: The category that immediately dominates a c-commanding element α (= the minimal phrase that contains a c-commanding element α) is the c-command domain of α.

In the example, S is the c-command domain of NP$_i$ and VP, and VP is the c-command domain of V and NP$_j$.

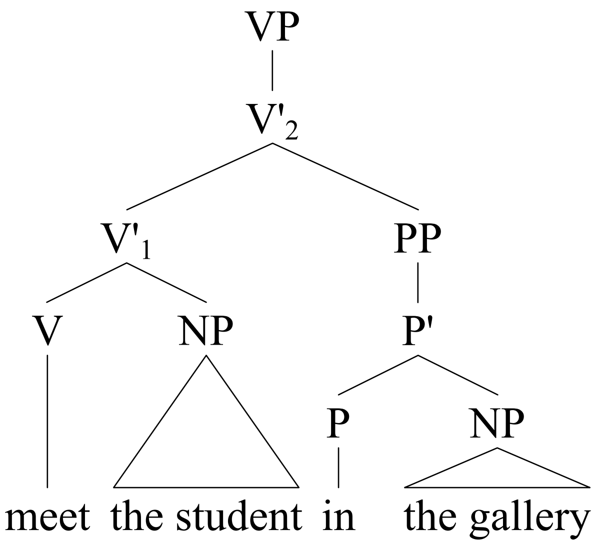
M-command

"m" in m-command stands for "maximal." The main difference from c-command is that we count only maximal projections.

M-command:
A category α m-commands a category β if and only if the **first maximal projection** that dominates α also dominates β, and neither dominates the other.

M-command domain: The smallest maximal projection that dominates an m-commanding element α (= the minimal maximal projection that contains an m-commanding element α) is the m-command domain of α.

(3)



V in (3) m-commands both the NP *the student* and the PP *in the gallery* and what is contained in them.

The m-command domain of V is VP, since VP is the smallest maximal projection that contains V.

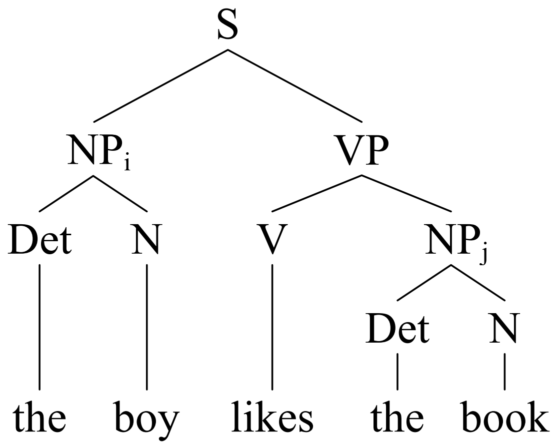
V c-commands (and m-commands) the NP *the student*, since the first branching node (V'_1) that dominates V also dominates the NP, but it does not c-command the PP *in the gallery*, since V'_1 does not dominate the PP, although it m-commands it, since VP dominates both.

P does not m-command V, because there is a maximal projection PP which dominates P and does not dominate V. Therefore, the m-command domain of P is PP, not VP, since PP is the smallest maximal projection that contains P.

Government

Government is a structural relation between a head of a phrase and its complement. A lexical head governs its complement in the phrase of which it is a head.
A lexical head is an X^0 category and a complement is a phrase (an XP).

(2)



The following government relation holds in (2):

V governs NP_j

Binding

Binding is a coreference relation between NPs. An NP_i is said to bind another NP_j iff NP_i c-commands NP_j and they are co-indexed. An NP that is not bound is free.

Principles of Binding Theory

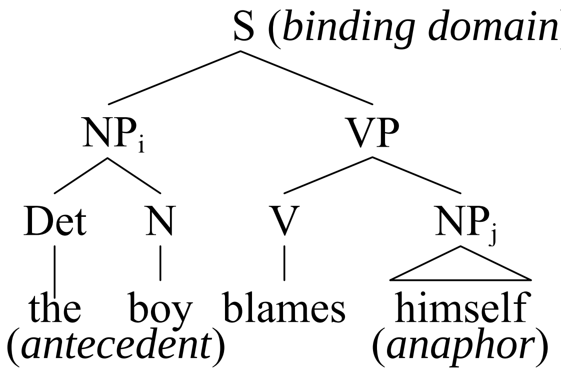
A. An **anaphor** is **bound** in its binding domain.
B. A **pronoun** is **free** in its binding domain.
C. All other NPs are always free.

Binding domain

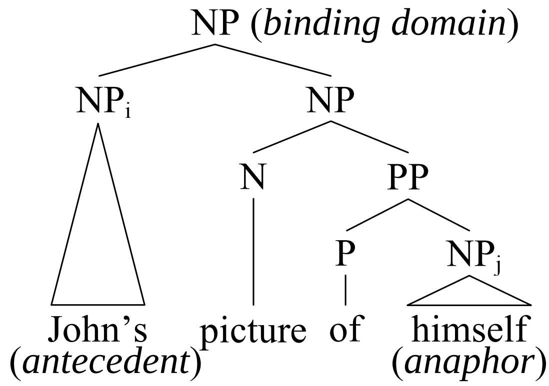
The binding domain of α is the minimal phrase (NP or S) containing α and a possible c-commanding antecedent of α.

Examples

(4)



(5)



The following binding relations hold in (4) and (5): the NP *the boy* binds the NP *himself* in (4) and the (same) anaphor is bound by the NP *John* in (5).

Head–Complement

The head–complement relation is a central notion in syntactic theory. It captures the following general insights:

- **A head** determines the properties of the phrase that contains it. (A phrase inherits certain of its grammatical properties from its head.)
- **A head** determines what else a phrase may/must contain in addition to the head.

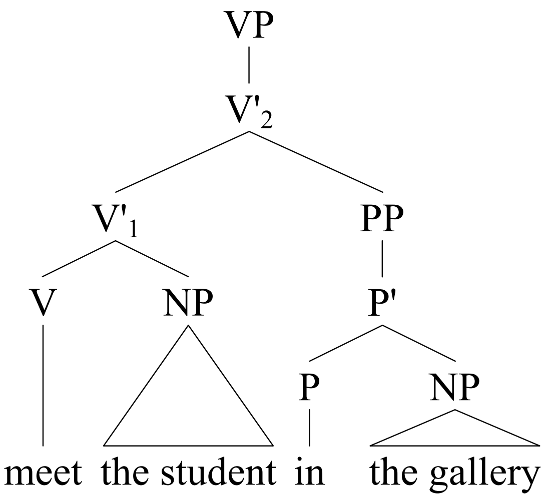
Therefore, in general, we assume that

- every phrase is headed by some elementary syntactic category (V, N, A, Adv, or P) (= every phrase is projected by some head), and
- every elementary syntactic category heads a phrase (= every head projects a phrase).

These are very general assumptions about how morphemes of a language enter into phrase structure.

For example

(3)



- The V head of the largest containing phrase determines its category – this is why the topmost category is VP (and not PP, e.g., although VP contains a P; but this P is the head of a different phrase, PP, contained in VP).

- The V head *meet* requires an NP complement, cf.

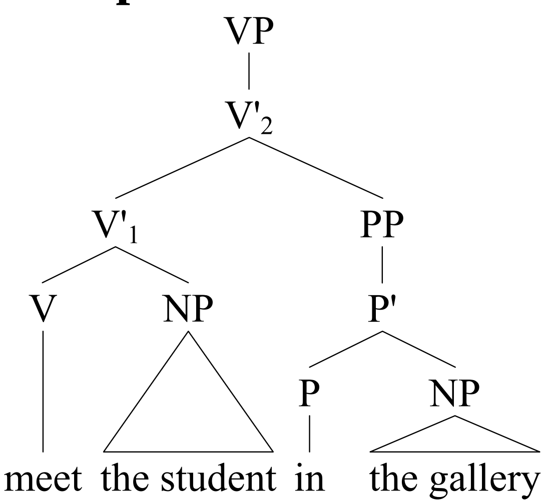
(6) a. *[$_{VP}$ meet] is bad, because it does not contain a complement.
b. *[$_{VP}$ meet [$_{AP}$ very hungry]] is bad, because it contains the wrong kind of complement (an AP).
c. [$_{VP}$ meet [$_{NP}$ the student]] is OK, because it contains a complement of the right sort (an NP).

All this because of grammatical properties of the V *meet*.

A **complement** is a phrase (an XP, a maximal projection) that is sister to a head.

For example

(3)



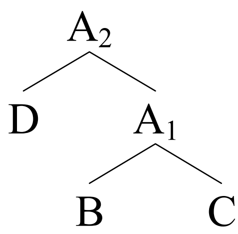
The NP *the student* is the complement of V in V'$_1$ and the NP *the gallery* is the complement of P in P'.

Adjuncts vs. Complements

Complements and adjuncts occupy different structural positions. A complement is sister to a head in a phrase (distinct from either), an adjunct is not.
An **adjunct** is a **constituent** in an **adjoined position**.

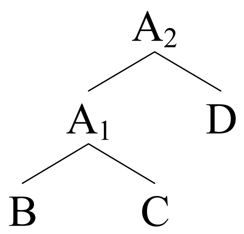
For example

(7) Left-Adjunction



(8) Right-Adjunction

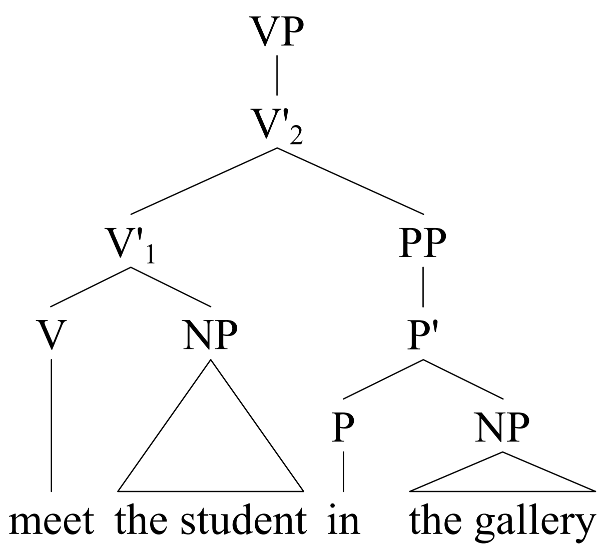
d.



The constituent D in (7) and (8) is adjoined to A (as its sister), to yield another segment A_2 of the category A. In (7), D is adjoined to the left of A (left-adjunction), in (8) it is adjoined to the right of A (right-adjunction).

Contrast this with the complement position of C. C is complement of B in A (assuming that B is the head of A).

(3)



PP in (3) is adjoined to V' (*meet the student*), to yield another instance of V' (V'_2, the same category, except a little "richer" than V'_1):
[$_{V'}$ V' PP].

Adjunction does not "convert" a category into a different category, complementation does. Adjunction: [$_{V'}$ **V'** PP]. Complementation: [$_{V'}$ **V** NP].

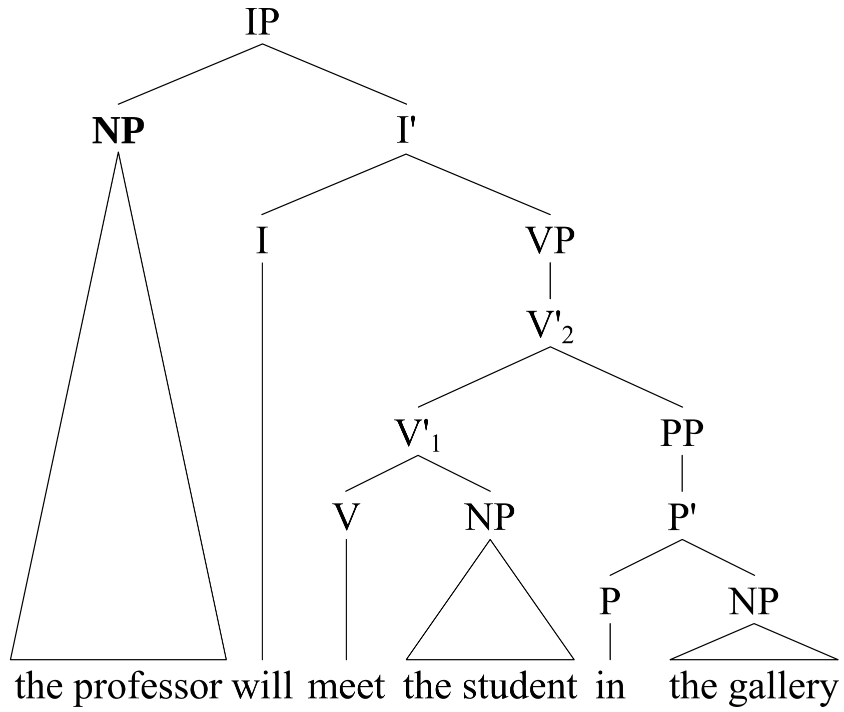
Complements are typically **obligatory** constituents. Adjuncts are typically **optional**. Contrast:

	Complement (*the student*)		Adjunct (*in the gallery*)
a.	meet **the student**	c.	meet the student **in the gallery**
b.	*meet ________	d.	meet the student _______

Subject

The subject of a sentence is the category that occupies the specifier position of IP ([Spec, IP]). **For example**

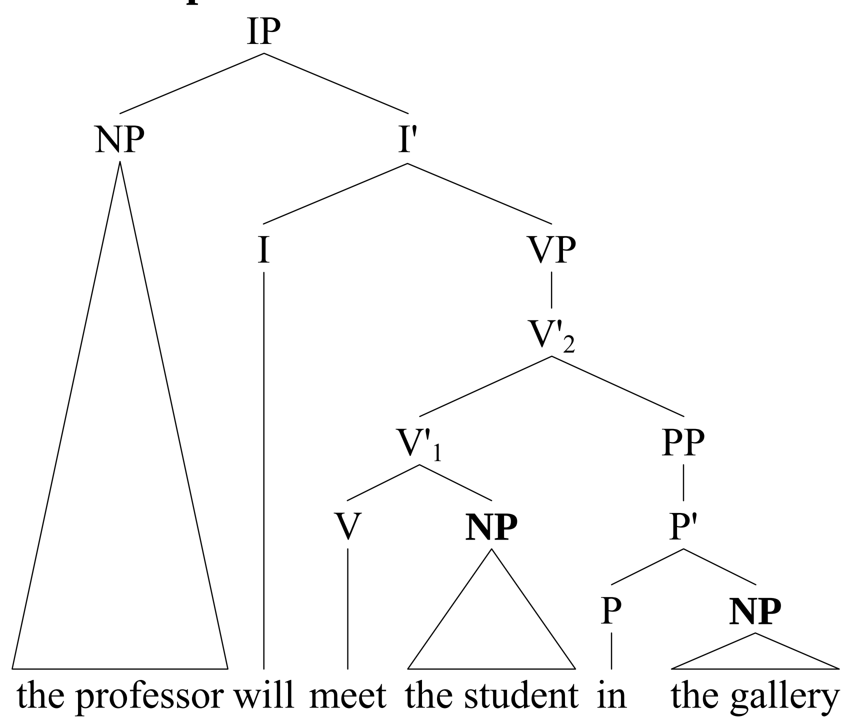
(9)



Object

An object is a **complement** which is selected and directly governed by a lexical head. **For example**

(9)



The NP *the student* is the object of the V *meet.*

The NP *the gallery* is the object of the P *in.*