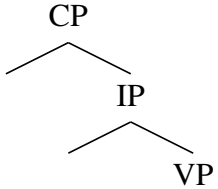
CAS LX 422 ∼ GRS LX 722 Intermediate Syntax
Spring 2021 January 26, 2021
Paul Hagstrom **1.** ***History, reminder, overview…***

The enterprise (reminder)

- Language faculty, part of being human (genetically specific)
- Comprises a system which highly restricts the range of possible human languages
- This system underlies all human languages, we seek to discover its properties
  - Looking *in depth* at the syntactic properties of individual languages
  - Comparing properties across languages to see what the options are
- Once we see patterns we
  - state generalizations
  - form a hypothesis (what causes the generalization?)
  - check other predictions of the hypothesis
  - revise as needed
- As the theoretical hypotheses develop, we see new things to check that we wouldn't have thought to check before. In the process, whether the hypotheses are right or wrong, we have *learned* a log about how language works.
- **We assume that there is a fact of the matter, that the Truth is Out There.** We are just trying to figure out what it is.
- Lots of proposals exist, some probably closer to the Truth than others.
- To evaluate them, we must see what they predict:
  - Do they predict that all of the possible languages/constructions exist?
    - *English sentences must be SVO*
      - I left. *Taxes*, I can deal with, it's the *forms* that bother me.
  - Do they predict that all of the impossible languages/constructions don't exist?
    - *English sentence may have any word order at all.*
      - *Like I pizza. *Bill thinks Mary that left.
- When two hypotheses both seem to predict the data, we get into aesthetics.
  - Which hypotheses are "simpler," more "elegant"?
  - Which hypotheses require the fewest stipulations, assumptions?
  - Which hypotheses make the greatest range of predictions?
  - Which hypotheses can account for the limited *crosslinguistic* options?

The clause structure of "the olden days" (say, 1986)

(1)


- CP: force (declarative, interrogative)
- IP: functional morphology (tense, agreement), auxiliaries
- VP: lexical/thematic elements

But this doesn't seem to be quite good enough to describe/explain the data—progressively, they have been split up into further functional projections. We'll see some of this later this semester even. And more in advanced syntax.

1 A little bit of history

This kind of generative grammar has progressed through several stages, sometimes names

- *Aspects*
- Standard Theory
- Extended Standard Theory
- Revised, Extended Standard Theory
- Government and Binding (GB)
  - *LGB* (1981)
  - *Barriers* (1986)
- (Kayne 1994: Antisymmetry)
- Minimalist Program (MP) (sort of)
  - A Minimalist Program for Linguistic Theory (1993)
  - Chapter Four (1995)
  - Derivation by Phase (1999)
  - Beyond Explanatory Adequacy (2001)
  - ...

There have been offshoots along the way (again, sort of). Some you might hear of:

- Lexical Functional Grammar (LFG)
- Head-driven Phrase Structure Grammar (HPSG)
- Relational Grammar (RG)

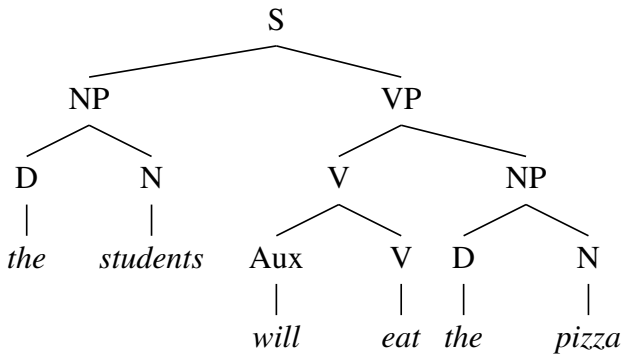
1.1 Early generative tradition

Basically, encoding the observations that sentences are *made up of* certain things. We might say: A sentence (of English) has a subject and a predicate. Subjects can be simple (*John*) or complicated (*the rumor that John asked the manager to write a comprehensive report showing opportunities for downsizing*); predicates come in various types as well (*left*, *surprised me*, *gave the union representative a headache*).

Properties of phrase structure were encoded in *phrase structure rules* ("rewrite rules").

(2) S → NP VP
VP → Verb NP
NP → D N
Verb → Aux V
D → *the*, *a*
Aux → *will*, PAST
N → *students*, *pizza*
V → *eat*, *see*

(3)



(4)
a. S
b. NP VP
c. D N VP
d. D N Verb NP
e. D N Aux V NP
f. D N Aux V D N
g. the N Aux V D N
h. the students Aux V D N
i. the students will V D N
j. the students will eat D N
k. the students will eat the N
l. the students will eat the pizza

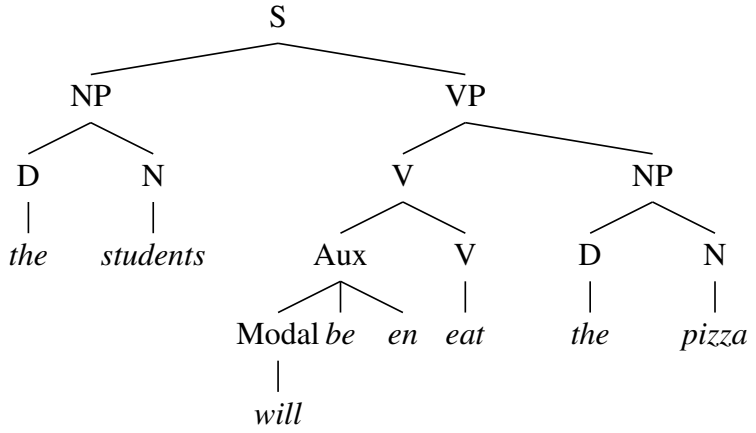
The tree structures indicate rule applications. The goal is to come up with a complete set, which is known to speakers of a language (e.g., English), that will be able to *generate* all of the sentence a native speaker considers to be English, and will not be able to generate any sentences a native speaker does not consider to be English.

Some sentences seem to be related to other sentences, and the idea is that, along with the PS rules that create the Deep Structure (DS), there are *transformations* that can then apply, that reorganize the tree in particular ways. For example, the formation of the passive, or of *wh*-questions, etc.—this is *movement*.

(5) a. The students will eat the pizza.
b. The pizza will be eaten by the students.

(6) ***Passive transformation***
Structural description: NP_1 — Aux — V — NP_2
Structural change: NP_2 — Aux — be+en — V — by — NP_1

(7)



(8) Aux → (Tns) (Modal) (have+en) (be+ing) (be+en)

(9) The sandwich must have been being eaten by Pat

(10) The sandwich must have en be ing be en eat by Pat

(11) ***Affix hopping transformation***
SD: Affix — V
SC: V — Affix

The basic structure of the system, according to this view:

(12) Phrase structure rules
↓
Deep structure → Semantic interpretation
↓
Transformations
↓
Surface structure → Phonetic interpretation

Gets: captures hierarchy, recursion, constituency. **But:** Language sees to be relatively constrained: Lots of PS rules could exist, but don't. Across languages, there is a high degree of similarity. The system is **missing a generalization** in many cases. What are the possible phrase structure rules? What are the possible transformations?

1.2 The possible PS rules: X′ theory

It turns out that there are a number of generalizations about PS rules. For example, no rule looks like this, where a VP has no V within it:

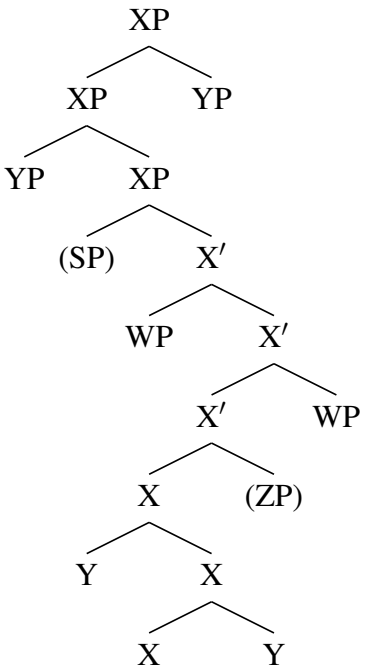
(13) VP → D N

X′-theory is a statement of possible PS rules:

(14)	XP → YP XP	← XP adjunct	X′ → X (ZP)	← Complement
	XP → XP YP	← XP adjunct	X → X Y	← head adjunct
	XP → (SP) X′	← Specifier	X → Y X	← head adjunct
	X′ → WP X′	← X′ adjunct		
	X′ → X′ WP	← X′ adjunct		

"X" here is an abstract category. So the rules above would apply to VP, say, or NP. It says that all phrases have a "bar" node of the same category, which dominates a head of the same category. The phrase might have a daughter (other than the "bar" node), which is a phrase (the "specifier"). The head might have a sister that is a phrase (the "complement"). And the bar node itself might have another bar node as its daughter, with a phrasal sister. We also allow for adjunction to heads and to XPs (where "adjunction" to something means that it is a combination that doesn't change its "bar level"—so WP, YP, and Y above and below are adjuncts of various sorts).

(15)



Another thing that X′-theory sought to capture is certain parallelisms that seem to exist across categories: in particular with respect to gerundive nominals and derived nominals (*John's refusing the offer*, *John's refusal of the offer*).

Casting the sentence (S) in terms of X′ structure was a further development, where it was hypothesized that the auxiliary/tense was really the head of the sentence; hence, S became IP (inflection phrase). And, then the sentence introducing complementizer was brought in as well, to form the CP, making X′ structure a fully general template for the construction of sentence structures.

1.3 Deriving X′-theory

Although we'd made progress by generalizing from PS rules to X′-theory (in the process explaining why certain kinds of PS rules don't exist for natural language), the X′ schema itself is still a stipulation. And we might wonder if the *schema* could be explained.

There were a few attempts at this: The most currently prominent ones are the "bare phrase structure" of the Minimalist Program (basically what we will be doing here) and the "Antisymmetry" approach to phrase structure, which we will discuss at least a bit in the advanced syntax course. Both sought to posit even more general principles from which the properties stipulated under X′-theory arise. (And, perhaps, we're actually better off empirically without those properties that were stipulated but not derived.)

1.4 Government & Binding

Government & Binding (GB) theory was a break from the previous theories in its movement toward more general principles, rejecting specific rules. Asking the question: *Why* is it that way? (In general, in fact, this is how the theories move from one stage to the next.)

The basic shape of the theory is that it consists of a number of somewhat independent modules that govern the well-formedness of trees.

The idea is: Rather than making each derivational step by following one of a set of grammatical rules, you can do whatever you want—so long as you satisfy the requirements imposed by the modules such as these:

θ-theory Each θ-role must be assigned to an argument (chain), and each argument (chain) must get a θ-role.

Case theory Each NP (chain) must get Case.

X′-theory Structures must conform to the X′ schema.

Binding theory Indices must conform to Principles A, B, C.

Bounding theory A moved element cannot be too far from its trace.

Control theory Things dealing with PRO.

1.5 Case in the era of GB

The Case Filter: *NP if NP has phonetic content and has no Case.

This idea was not hit upon right away, particularly because in English, you don't see a lot of evidence for case on NPs. However, if we suppose that NPs need Case (and in other languages it can even be seen), and that Case is only available to NPs in certain places, then we can derive quite a bit about syntax. So, it was thought that V, P, and finite I *assign* case to NPs. But there are restrictions: The reach does not seem to be very far.

(16) a. Subject of a tensed clause: nominative
b. Object of a verb: accusative
c. Object of a preposition: accusative

(17) a. Pete hit me.
b. * Pete hit savagely me.

It seems as if V can only assign case to an adjacent NP. Also, it seems to be possible either to assign case to one's complement (as V and P do) or to one's specifier (as I does).

Plus, there are cases where V and C seem to be able to reach in to the sister and assign case tot he NP in the sister's specifier (the Exceptional Case Marking case):

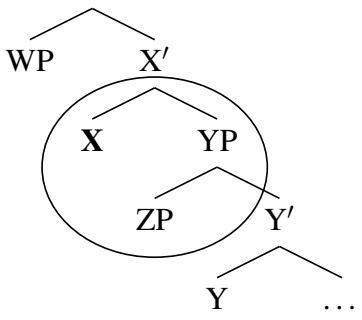
(18) a. Hank considered me to have lost the battle.
b. For me to get there on time, I will have to leave now.

So, what is the requirement on the configuration of case-assigner and case-recipient?

We say that the case-assigner *governs* the case-assignee. This relation of *government* holds between the head and all positions to which it could assign case.

A head *governs*, essentially, its complement, the specifier of its complement (and, sometimes, its own specifier). Heads assign Case to positions they govern. The properties of government differed depending on whether a head is lexical (N, V) or functional (I, C, D).

(19)



(20) John met him.

(21) John wants him to leave.

This still leaves out the assignment of Case to the specifier of IP, however. Perhaps government should be extended to WP as well... and so forth; there was a lot of effort expended by the community in detailing the precise characterization of government, and where it is used.

1.6 The Minimalist Program

Another shift in perspective, toward deriving the principles from more basic "design properties" of the system. The feeling was that this might be getting too complicated to be right. Maybe we can find the complexity in the interaction of simple systems, rather than being complexity within a single system.

C_{HL}, stuck between A–P (PF) and C–I (LF), which impose requirements. **Is human language more than an optimal solution to those requirements?** Is all of the complexity we find in syntax ultimately traceable to the *design requirements*?

(22) **Strong Minimalist Thesis**
Language is an optimal solution to legibility conditions

Some starting points, assuming (22) and the idea that less machinery is better than more.

(23) a. The only linguistically significant levels are the interface levels.
b. The **interpretability** condition: Lexical items have no features other than those interpreted at the interface (properties of sound and meaning).
c. The **inclusiveness** condition: The machinery of syntax (C_{HL}) does not introduce any new features not already contained in lexical items.

1.7 Features and the lexicon

UG makes available a set F of features (linguistic properties) and computational operations C_{HL} (the computational procedure for human language) that make use of F to form expressions. A particular language L uses F to form a particular set of expressions {EXP}. A given EXP is pairing of meaning (LF) and sound (PF), interface representations.

A language L takes those features and bundles them, forming a **lexicon**.

1.8 Bare phrase structure

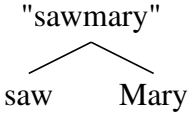
We know that sentences consist of hierarchical structure. In order to get hierarchical structure, we need to be able to take two things and put them together into a combined thing. Like *lunch* and *eat*, which we put together to form what we had called the "VP," for example.

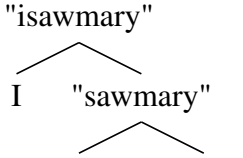
So, what if that's all we have? A set of lexical items (a **numeration** or **lexical array**) that we'll build up into a structure, and a procedure for combining them into successively larger structures. We're building the structure from the bottom up.

(24) Numeration: *I*, *saw*, *Mary*.

(25) Merge: *saw* and *Mary*

(26) Merge: *I* and *"sawmary"*

(27)


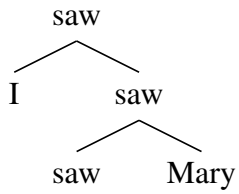
(28)


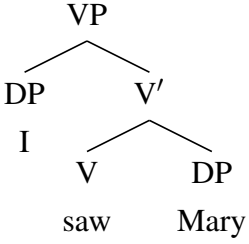
The combination of *saw* and *Mary* is what we'd have called the VP in the past, it has more of the properties of *saw* than it has of *Mary*. If all we are doing is arranging things, what we've really done here is made something like

(29) {*saw*, *Mary*}

(30) {*I*, {*saw*, *Mary*} }

So if we assume something like: *saw* is a transitive verb, and so needs an object and a subject; providing *Mary* and then *I* to *saw* gives it the object and subject it needs; the thing whose need is satisfied is the thing that gets to be the label—then we have a tree that's something like this:

(31)


(32)


And we can kind of "read off" the more familiar positions. The *head* is the lexical item, the first time it is in the tree. The *complement* is the sister to the head, it's the thing that Merges first. The *specifier* if there is a second Merge. There are no actual bars or Ps, but we can tell where they'd be. The difference between head and phrase is possibly muddied, but perhaps that's correct?

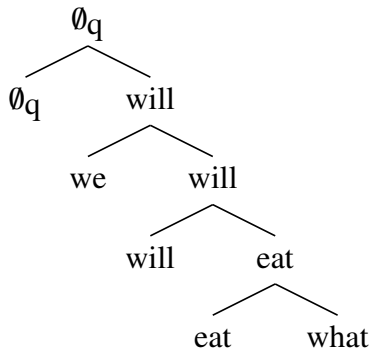
1.9 Movement

Once we have Merge, we can also implement movement as well. Merge allows us to take two previously-formed objects (or things we've taken from the numeration) and combine them into one new object. One of the two things we just combined serves as the label for the combination. (We say that the thing that becomes the label has "projected.")

In the situation above, we took two things from the numeration and made a new thing out of them. But what if you take one of the two things from *inside* the other one instead? That could count, the things inside the structure are also things. One might even argue that it would be *more complicated* to state a restriction that says you aren't allowed to pick objects within previously-formed objects.

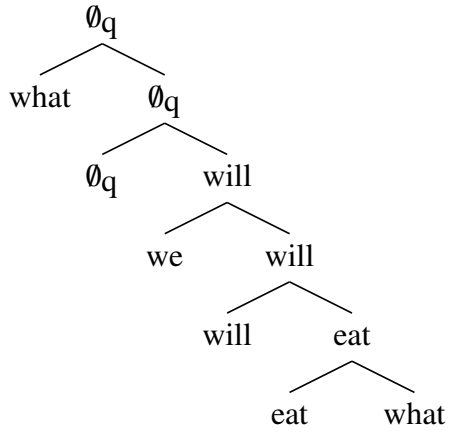
So, suppose you have something like this:

(33)



And then we Merge that with *what* within it. We wind up with this:

(34)



That is to say, it is *wh*-movement of *what*. We imagine that there is something about $\emptyset_q$ that indicates that it needs to Merge with a *wh*-word and a search of its tree reveals *what*, which it then Merges with itself. We now have two copies/instances of *what*. In English at least, we'd pronounce the top one, and not the bottom one. The bottom one is the "trace" of the movement.

Notice one other thing that is interesting here: the moved position of *what* c-commands the trace. Generally speaking, that's something we believed to be true of movement before: movement is always to a c-commanding position. It is always upward, to a position that c-commands its original position.

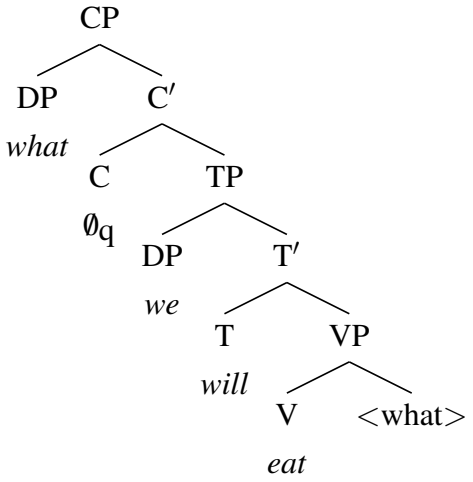
But what is c-command? What is movement? If you look at this, c-command is a relationship that is established by Merge. If you Merge X and Y, X c-commands Y and the things contained in Y (and Y c-commands X and the things contained in X). If movement is taking something inside X and Merging it with X, the fact that a moved item c-commands its trace is hardly newsworthy. It's basically defined to be that way, it can't *not* c-command its trace.

1.10 So

Looking at syntax this way has significantly simplified things. We no longer have "XP" or "X′" nodes in our structures, our structures are only just what we needed—hierarchical arrangements of lexical items. The "XP" and "X′" nodes were just there for our viewing convenience, but nothing in the system refers to them or relies on them. The definition of c-command just falls out, when you Merge two things together, it's the relationship of one to the other. We derived the apparent fact that movement has to be upward to a c-commanding position. We have very little in our system at all: we have a lexicon, we have a numeration, we have Merge.

However, it's also true that these "bare phrase structure" trees are kind of hard to look at and read, particularly for people who have been doing syntax for a while. So it is has become conventional to just write them like we always did—but now with the understanding that the "′" and "P" notations aren't doing any work, they're just there to decorate the trees on the page. So we'll still draw the trees more like this:

(35)



But with the understanding that a lot of that stuff is now purely decorative. We just put a "P" on something when it no longer projects (when it is at the top or its sister projected), and we put a "′" on everything that projects except when it just came out of the numeration. We use the category (something we assume is a feature of lexical items) as the first part of the label, so if the lexical item is *eat*, we put "V" on it and its projections. We put $< \ldots >$ around something if it is c-commanded by another copy of itself. But this is still just decoration, a shorthand to make it easier to read the trees and compare them to trees from previous work.

We still need to consider when we're allowed to use Merge, and what it means to have an arrangement that works (versus one that does not). And that's kind of what the rest of the semester is about. But we are well on our way.

To preview the kinds of things that we might say makes a good arrangement: you need to use everything in the numeration (you have to wind up with a single object). Some lexical items have "needs" that must be satisfied by combining them with something else (easyish examples: a transitive verb needs to be combined with an object, or a *wh*-question C needs to be combined with a *wh*-word). We need to figure out how to characterize those "needs" and figure out what different kinds of needs there are (and what different kinds of ways to satisfy them there might be). The overarching idea is that if you don't satisfy those needs, then the system that needs to interpret this structure won't be able to. Either it will be semantically incoherent in some way (unable to be interpreted by the conceptual-intensional interface), or it will be structurally/linearly incoherent in some way that prevents it from being pronounced/externalized. Some of this will still require a little bit of stipulation, realistically, but the idea is to minimize what we have to assume/stipulate as much as possible.