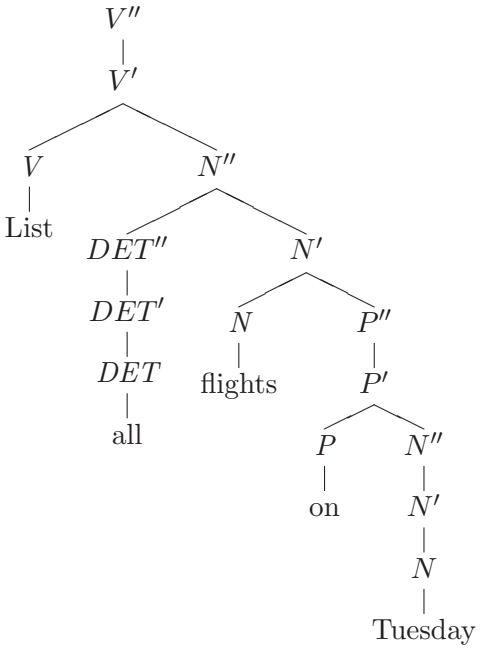
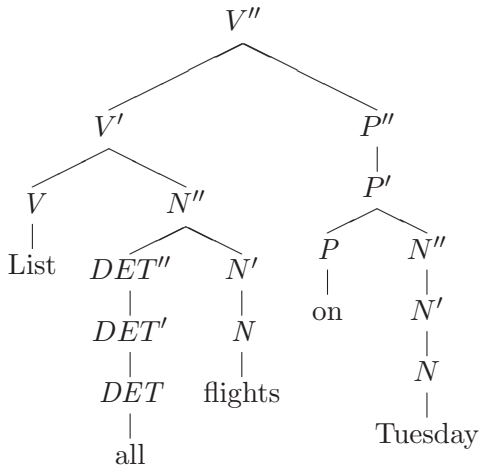
Computation, Information, and Intelligence (COMS/ENGRI/INFO/COGST 172), Fall 2005
10/24/05: **Lecture 25 aid — Context-free grammars**

Agenda: More on X-bar(-like) theory; context-free grammars

Follow-up: Alert classmates later reported also seeing Google perform query expansion or mutation: try (with no quotes) "liberachi", "slimmest camera", or "ACL 2005" (a conference on computational linguistics). Google has apparently been testing this idea since at least mid-August.

I. Two possible X-bar-theory-flavored[1] analyses Generally speaking, a "modifier"[2] double-bar constituent M (which generally is of a very different "type" from its parent) modifies or relates to the child of its parent that contains the parent's *head*.





Note that these differ from the more "intuitive" analyses given on the lecture aid for last time.

II. Context-free grammar (CFGs): definition Four components must be specified:

- the *terminals*: a finite set of at least one symbol;
- the *non-terminals* a finite set of at least one symbol, where no symbol can be both a terminal and a non-terminal;
- a single designated start non-terminal; and
- the *rewrite rules*: a finite set of at least one rule describing how a single non-terminal can be rewritten as a sequence of terminals and/or nonterminals (possibly intermixed[3]).

[1] We're oversimplifying; note, for example, that in the second tree there's no "room" for the (implicit) subject of the sentence. Different theories handle this problem differently. One way to proceed is to change the V'' at the top to a V', and then add a new V'' as root whose only child is the new V'. Allowing an X' to have an X' as child (in situations like this, anyway) can be motivated on linguistic grounds.

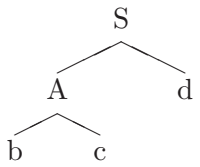
[2] We are using the term "modifier" loosely to include things like (non-optional) direct objects.

[3] And possible *empty*, although in this class we will try to avoid dealing with empty strings.

III. Related concepts

Sentences — sequences of terminal symbols — are generated by rewriting the start non-terminal until no non-terminals are left. They are also known as *strings (of terminals)*.
The *language* generated by a CFG is the set of sentences the CFG generates.

We can represent the rewriting process by *parse trees* (which, abusing terminology, we will also refer to as being "generated" by CFGs). In a parse tree, the interior nodes are labelled by non-terminals, the root is labelled with the start non-terminal, the leaves are labelled by terminals, and the children of an internal node represent, in order, the result of rewriting the non-terminal labelling the node according to one of the rewrite rules in the grammar. That is, if we have the following parse tree:



then the CFG generating this parse tree must contain the rewrite rules $A \rightarrow bc$ and $S \rightarrow Ad$.

IV. Example CFG

- Terminals: a, b
- Non-terminals: S
- Start non-terminal: S
- Rewrite rules: $S \rightarrow$aSb and $S \rightarrow$ab

V. Example language that can be generated by a CFG All and only sentences of the following form, where $n \geq 1, m \geq 0$:

$$\underbrace{\text{a}\ldots\text{a}}_{n}\underbrace{\text{b}\ldots\text{b}}_{n}\underbrace{\text{c}\ldots\text{c}}_{m}\underbrace{\text{d}\ldots\text{d}}_{m}$$

VI. A linguistically-motivated CFG For the sake of brevity, this grammar is geared toward our "intuitive" analyses from last lecture of "list all flights on Tuesday", rather than the X-bar-like analyses on the front of this handout.

- Terminals: list, all, flights, on, Tuesday
- Non-terminals: S, NP, N', PP, V, DET, N, P
- Start non-terminal: S
- Rewrite rules:

(1)	S	$\rightarrow$	$V\ NP$	(7)	N	$\rightarrow$	flights
(2)	S	$\rightarrow$	$V\ NP\ PP$	(8)	N'	$\rightarrow$	$N\ PP$
(3)	V	$\rightarrow$	list	(9)	PP	$\rightarrow$	$P\ NP$
(4)	NP	$\rightarrow$	$DET\ N'$	(10)	P	$\rightarrow$	on
(5)	NP	$\rightarrow$	$DET\ N$	(11)	NP	$\rightarrow$	N
(6)	DET	$\rightarrow$	all	(12)	N	$\rightarrow$	Tuesday