



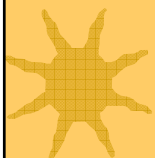
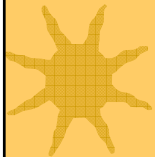
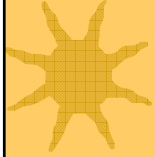
## *CMSC 723 / LING 723: Computational Linguistics I*

---

November 28, 2007

Lecture 13 (Madnani):  
CCG and Tree-Adjoining Grammar  
(Chapter 12 and other readings)

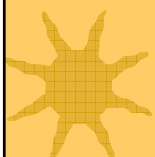
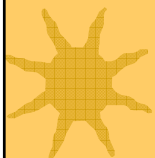
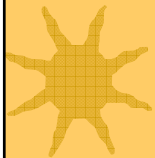
Prof. Bonnie J. Dorr  
Co-instructor: Nitin Madnani  
TAs : Hamid Shahri, Alexandros Tzannes



## *Context-Free Grammars → Dependency Trees*

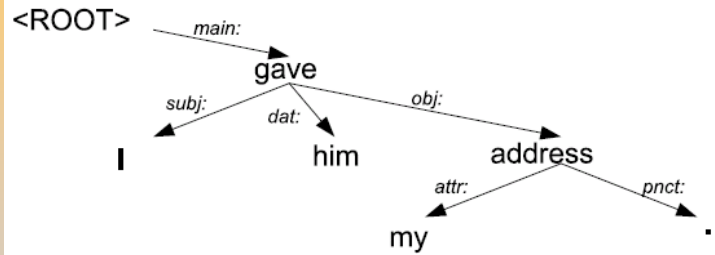
---

- ★ Most grammars studied so far have been context free. (Why?)
  - Many available Treebanks and parsers are based on this kind of syntactic representation.
- ★ Context-Free Grammars are based on constituents (like Verb, Noun, VP, NP, etc.)
- ★ Dependency grammar is different: Words are related by binary semantic and syntactic relations.





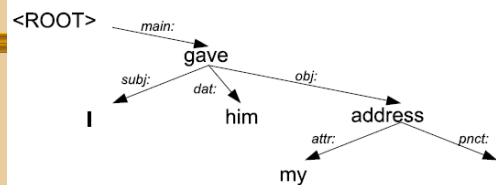
## Sample Dependency Grammar Parse



- ★ No non-terminals or phrasal nodes.
- ★ Each link holds between two lexical nodes.
- ★ Augmented with special <ROOT> node
- ★ Links have names (subj, obj, etc.)



## Link Names



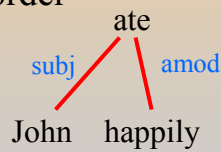
- ★ Drawn from fixed inventory of about 35 relations.
- ★ Most relations represent grammatical functions
- ★ Some semantic relations (not too deep).

| Dependency | Description   |
|------------|---|
| subj       | syntactic subject                                     |
| obj        | direct object (incl. sentential complements)          |
| dat        | indirect object                                       |
| pcomp      | complement of a preposition                           |
| comp       | predicate nominals (complements of copulas)           |
| tmp        | temporal adverbials                                   |
| loc        | location adverbials                                   |
| attr       | premodifying (attributive) nominals (genitives, etc.) |
| mod        | nominal postmodifiers (prepositional phrases, etc.)   |



## Advantages of Dependency Tree Formalisms

- ★ Strong predictive power
- ★ Useful for semantic understanding/analysis than constituency trees
- ★ Ability to handle languages with “free word order”

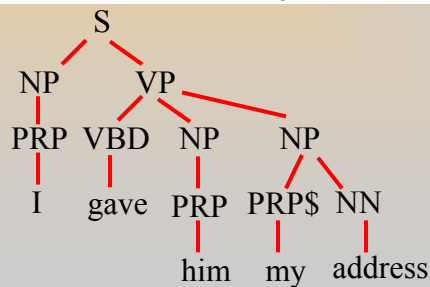
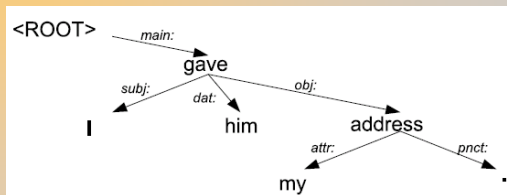


- ★ Non-projective: Crossing branches
- ★ Projective: No crossing branches

- John ate happily  
- John happily ate



## Relationship Between Dependencies and Phrasal Heads





# How to Convert from a Constituency Tree to a Dependency Tree

## Head percolation table

VP (r VP VBD VB VBN VBG VBZ )(l)  
 NP (r NX PRP) (r NNS NN NNP NNPS) (l NP)  
 (r CD FW SBAR RBS) (r)  
 PP (r IN TO) (r VBG PP) (r RB RP) (r)  
 S (l VP)(l)  
 SBAR (r WHADVP WHNP IN SBAR VP )(r)  
 ADVP (r RB IN JJ)(l NN RBR DT RP)(l)  
 QP (l CD RB)(l)  
 WHADVP (r WRB)(r)  
 WHADJP (r ADJP JJ)(l)  
 WHPP (r IN TO)(r)  
 WHNP (r WDT WP)(r)  
 FRAG (r NP SBAR ADJP PP ADVP NNS )(r)  
 SINVP (r VP)(r)  
 SQ (r VP VBD VBZ MD)(r)  
 RRC (r ADJP ADVP)(r)  
 ADJP (r JJ JJR JJS) (l DT NN)(l)  
 CONJP (l IN RB)(l)  
 INTJ (l VB UH)(l)  
 NAC (l NX PRP NNS NN NNP NNPS) (l CD  
 NP SBAR RBS)(l)  
 NX (l NX PRP NNS NN NNP NNPS) (l CD  
 NP SBAR RBS)(l)  
 PRT (r RP)(r)  
 PRN (r S NP NN SBAR ADJP PP )(r)  
 UCP (l NNS NN NNP NNPS NP) (l S) (l  
 ADVP ADJP )(l)

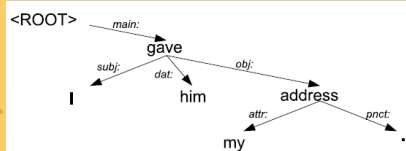
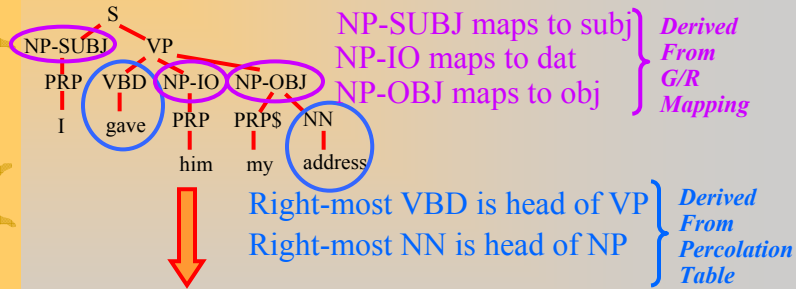
VP's most likely head is the right-most child that is a VP or VBD or VB or VBN, etc.; If all else fails, pick left-most child (default)

## Grammatical/Relationship Mapping

|               |                  |
|---------------|------------------|
| (JJ → adj)    | (S → i)          |
| (NR → mod)    | (DT → det)       |
| (NN → mod)    | (AD → amod)      |
| (NT → mod)    | (LOC → location) |
| (PN → mod)    | (PRD → pred)     |
| (VV → vmod)   | (TMP → time)     |
| (ADV → amod)  | (APP → appo)     |
| (CD → num)    | (NP → obj)       |
| (PUNC → punc) | (IO → dat)       |
| (PP → p)      | (OBJ → obj)      |
| (P → p)       | (SBJ → subj)     |
| (VP → vmod)   | (WH → wh)        |



# Example of Mapping: Constituency to Dependency Tree





## Collin's Head Percolation Table

| Parent Non-terminal | Direction | Priority List   |
|---------------------|-----------|---|
| ADJP                | Left      | NNS QP NN \$ ADVP JJ VBN VBG ADJP JJR NP JJS DT<br>FW RBR RBS SBAR RB |
| ADVP                | Right     | RB RBR RBS FW ADVP TO CD JJR JJ IN NP JJS NN                          |
| PRN                 | Left      |   |
| PRT                 | Right     | RP  |
| QP                  | Left      | \$ IN NNS NN JJ RB DT CD NCD QP JJR JJS                               |
| S                   | Left      | TO IN VP S SBAR ADJP UCP NP   |
| SBAR                | Left      | WHNP WHPP WHADVP WHADJP IN DT S SQ SINV<br>SBAR FRAG                  |
| VP                  | Left      | TO VBD VBN MD VBZ VB VBG VBP VP ADJP NN NNS<br>NP                     |



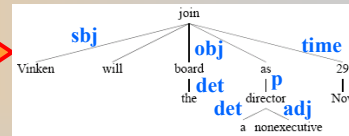
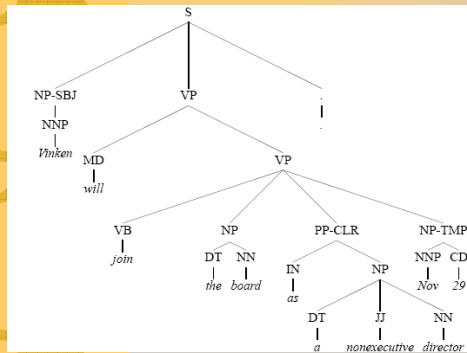
## Finding the Head of NP using Collins 1999 Rules

NP (r POS) (r NN NNP NNPS NX POS JJR) (l NP)  
(r \$ ADJP PRN) (r CD) (r JJ JJS RB QP) (r)

- If the last word is tagged POS, return last-word.
- Else search from right to left for the first child which is an NN, NNP, NNPS, NX, POS, or JJR.
- Else search from left to right for the first child which is an NP.
- Else search from right to left for the first child which is a \$, ADJP, or PRN.
- Else search from right to left for the first child which is a CD.
- Else search from right to left for the first child which is a JJ, JJS, RB or QP.
- Else return the last word



## Another CFG to Dependency Tree Example



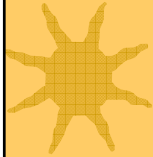
## Combinatory Category Grammar (CCG)

- ★ Categorical Grammar: Lexicon bears main responsibility for defining syntactic form
- ★ “Lexicalized” Theory of Grammar
- ★ Adheres to “principle of compositionality”
  - Homomorphism between syntax & semantics
  - We’ve seen this before:

[S [NP X] [VP [V saw] [NP Y]]] ↔  
 [SEE\_perc [Person X] [AT\_perc [Thing Y]]]



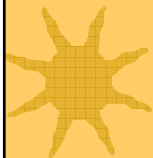
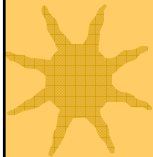
## CCG Notation



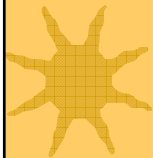
★ Similar to dependency tree: Associates functional type/category with all grammatical entities

★ Two types of categories:

- Arguments (e.g., nouns): have simple category, like N
- Functors: these are predicates, e.g., Verbs, Determiners. These have a complex category, e.g., X/Y or YX.
- Combinatory Rules: Allow functions and arguments to be combined.
  - $X/Y \ Y \rightarrow X$
  - $Y \ X/Y \rightarrow X$



## CCG Functors



★ X/Y is a function from Y to X, i.e., something that combines Y on its right to produce X.

– Examples:

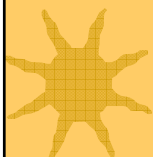
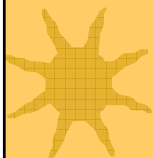
- “the” (determiner) has category NP/N: combines with N on its right to produce NP
- “eat” (transitive verb) has category VP/NP: combines with NP on its right to produce VP.
- “give” (ditransitive verb) has category (VP/NP)/NP: combines with NP on its right to yield transitive verb VP/NP. (Note extra required combination with NP to produce VP).

– Important: in CCG, we generally refer to the primitive VP category as S\NP. Lexical entry for eat: (S\NP)/NP. (Related notes below.)

★ X\Y is also a function from Y to X, i.e., something that combines with Y on its left to produce X.

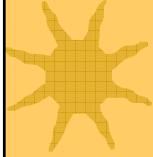
– Examples:

- “eat apples” has category S\NP: combines with NP on its left to produce S.
- “eat” has lexical entry (S\NP)/N [Replacement of VP above with S\NP]
- “give” has lexical entry ((S\NP)/NP)/NP

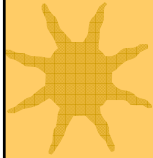




## Parsing with CCGs

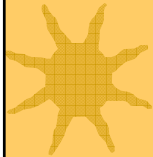


★ It's possible to specify a CYK algorithm for CCGs



★ Some operations that allow this to work:

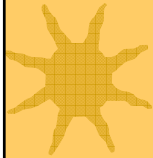
- $X \rightarrow X/Y \ Y$
- $X \rightarrow Y \ X/Y$
- $X/Z \rightarrow X/Y \ Y/Z$



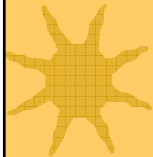
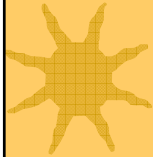
★ You need to understand how to come up with real NLP examples that are subject to operations like the ones above (and possibly others not listed here).



## Example Parse

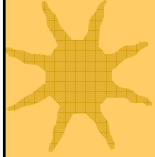


|  |                  |               |
|--|------------------|---------------|
| <u>Harry</u>                           | <u>eats</u>      | <u>apples</u> |
| NP                                     | <u>(S\NP)/NP</u> | NP            |
| <hr style="border: 1px solid black;"/> |                  |               |
| S\NP                                   |                  |               |
| <hr style="border: 1px solid black;"/> |                  |               |
| S                                      |                  |               |

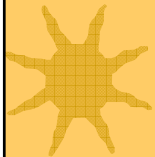




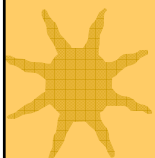
## *Tree Adjoining Grammars (TAGs): Mildly context Sensitive Formalism*



★ TAGs are “mildly context-sensitive”



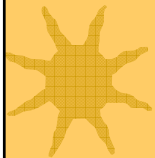
★ Beyond context-free, but not fully context sensitive.



★ Properly contain context-free languages and are properly contained by context-sensitive languages.



## *TAGs*



★ Introduced in [Joshi & Takahashi, 1975]. Reviewed in [Joshi and Schabes, 1997] (see syllabus).

★ A formal *tree* rewriting system

★ Today we will look at the basics of TAG Formalism

– Primitive elements: elementary trees

- Initial trees
- Auxiliary trees

– Operations

- Substitution
- Adjoining

– Derivations

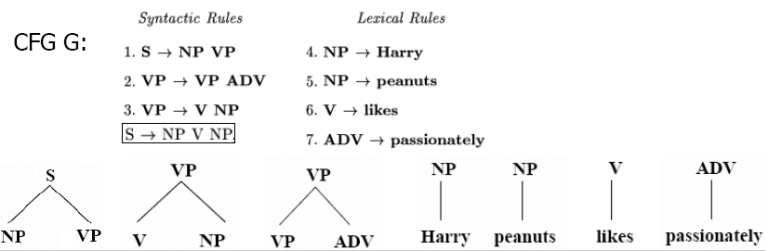
- Derived trees
- Derivation trees

★ TODAY'S SLIDES ARE ADAPTED FROM YUQING GUO, NCLT Seminar, 2006



## How do TAGs differ from CFGs?

- ★ In CFGs, each rule corresponds to a one level tree.
- ★ Not every rule is lexicalized.



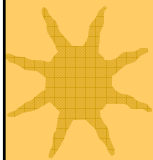
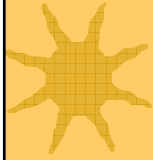
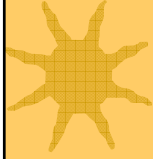
## What is missing from CFGs?

- ★ Rules are not lexicalized, e.g., there is no “lexical anchor” for the rule  $S \rightarrow NP VP$
- ★ To specify a lexical anchor, e.g., “eat”, we would need to build rules that go down more than one level:  

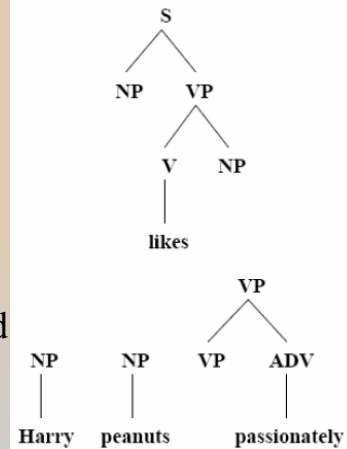
$$S \rightarrow [NP_1] [VP [V \underline{eat}] [NP_2] ]$$
- ★ Without multi-level, lexicalized rules there is no way to associate arguments (e.g.,  $NP_1$ ,  $NP_2$ ) with a lexical anchor.
- ★ Note that if we added these components to a CFG, it would no longer be a CFG—it would be something more powerful. (The one-level nature of CFG rules is what ensures efficient parsing times.)



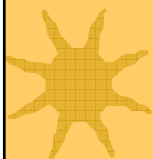
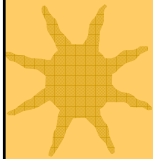
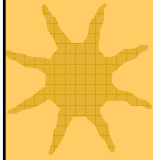
## Lexicalization of Grammars



- ★ Elementary objects are elementary trees
- ★ Each tree contains at least one frontier node labeled with a terminal symbol (lexical anchor)
- ★ Lexicalized trees are stored in lexical entries, e.g., “like: [S [NP] [VP [V like] [NP]]”



## TAG Definitions



- ★ Initial Tree:
  - all interior nodes are labelled with non-terminal symbols
  - the nodes on the frontier are either labelled with terminal symbols, or with non-terminal symbols marked for substitution ( ↓ )
 (Used for substitution operations)
- ★ Auxiliary tree:
  - one of its frontier nodes must be marked as foot node (\*)
  - the foot node must be labeled with a non-terminal symbol which is identical to the label of the root node.
 (Used for adjunction operations)



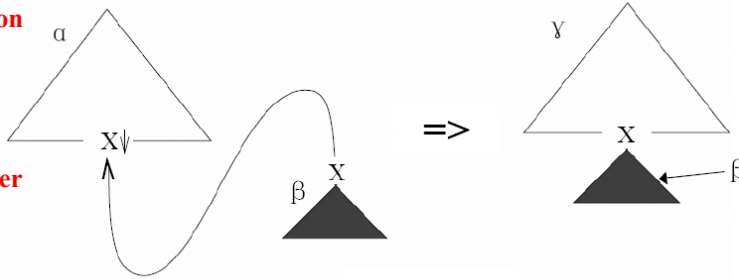
# Initial Trees and Substitution

**Initial Tree:**

$\alpha$

The most common usage for Substitution is on Initial Trees, but Substitution may also be done at frontier nodes of auxiliary and derived trees.

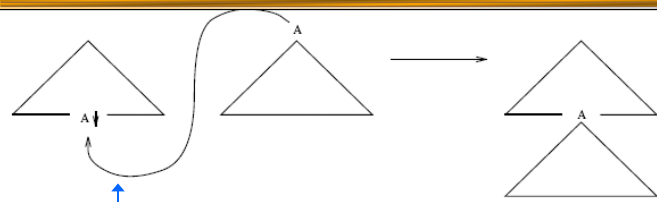
- all interior nodes are labelled with non-terminal symbols
- the nodes on the frontier are either labelled with terminal symbols, or with non-terminal symbols marked for substitution ( $\downarrow$ )



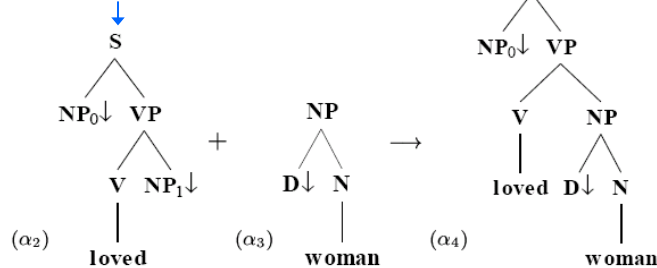
**Substitution:** Takes place on non-terminal nodes of the frontier of a tree  $\alpha$  (usually an initial tree). The node marked for substitution is replaced by the tree  $\beta$  to be substituted.



# Substitution Operation



Initial Tree

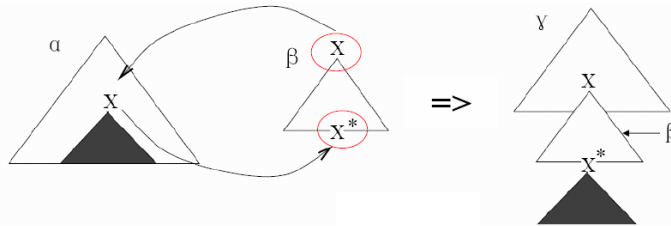




# Auxiliary Trees and Adjunction

## Auxiliary Tree $\beta$ :

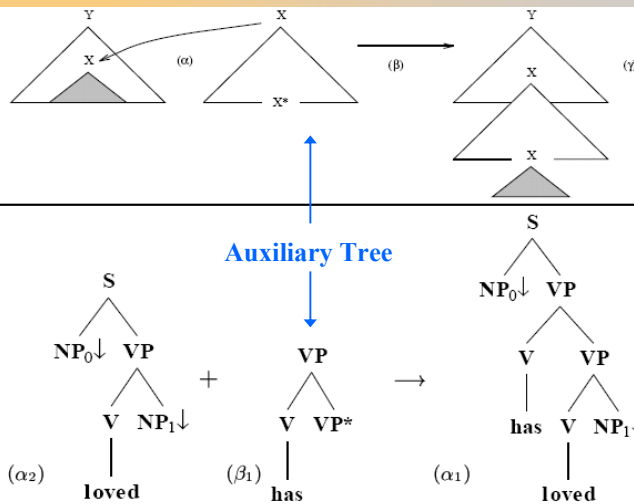
- one of its frontier nodes must be marked as foot node (\*)
- the foot node must be labeled with a non-terminal symbol which is identical to the label of the root node.



**Adjunction:** Builds a new tree from an auxiliary tree  $\beta$  (with root/foot node X) and a tree  $\alpha$  (with internal node X). The sub-tree at internal node X in  $\alpha$  is excised and replaced by  $\beta$ ; the excised sub-tree is then attached to the foot node of  $\beta$ .



# Adjunction Operation

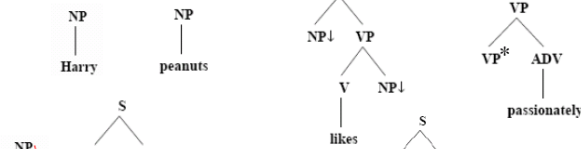




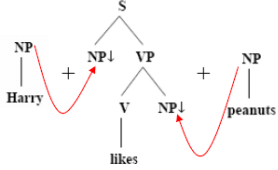
# Example of TAG Parsing

e.g. 'Harry likes peanuts passionately'

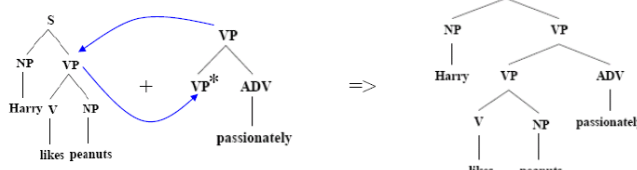
TAG G



step 1 substitution



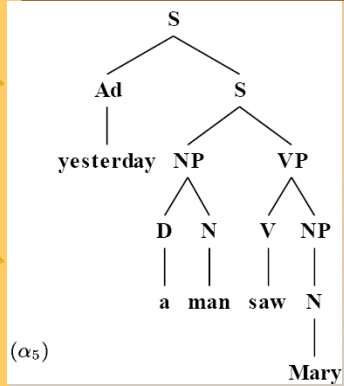
step 2 adjoining



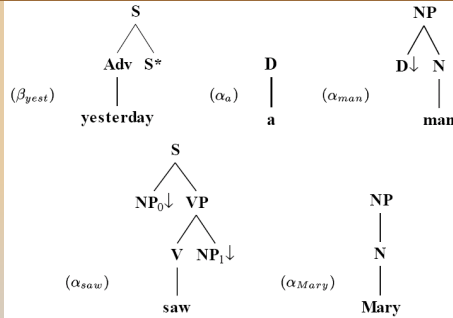
Derived Tree



# Derived Trees and Derivation Trees



Derived Tree



Elementary Trees used to produce Derived Tree at left

## Derived Trees and Derivation Trees (continued)

(α<sub>y</sub>)

**Derived Tree**

**Elementary Trees are combined as specified by Derivation Tree at right.**

**Derivation Tree**

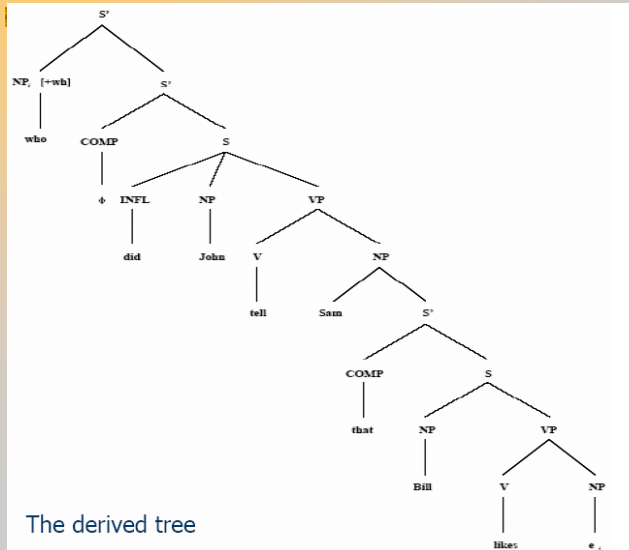
substitution  
adjoining

## Long Distance Dependencies

e.g. 'Who<sub>i</sub> did John tell Sam that Bill likes e<sub>i</sub>'



## Long Distance Dependencies (continued)



The derived tree



## Properties of TAGs

- ★ Let CFL = languages recognized by CFGs
  - ★ Let TAL = languages recognized by TAGs
  - ★ Let CSL = languages recognized by CSGs
  - ★ Context-free languages are strictly included in tree-adjoining languages, which themselves are strictly included in context-sensitive languages:
- Mildly context-sensitive:  $CFL \subset TAL \subset CSL$**
- ★ So what does this mean in terms of parsing??
    - Efficiently parsable (with modified CYK approach)
    - Polynomial time, like CFLs!
    - How can this be? Dependencies are localized!



## *CCG and TAG*

### ★ Similarities

- Both are lexicalized
- Both increase information present in the lexicon using a specific representation
- Both project items from the lexicon and combine them in various ways
- Both place responsibility for syntax in the lexicon

### ★ Differences

- Lexical items in CCG are single words with categories; in TAG they are (elementary) trees
- CCG is a string rewriting system whereas TAG is a tree-rewriting system
- CCG combines items using composition whereas TAG combines using adjunction and substitution

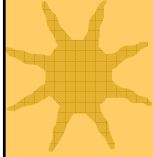


## *Next Time*

- ★ Readings from “Lecture 12” that we skipped last week.
- ★ Start putting notes together for final exam. Watch the website for some review questions, some of which we will go over next time.
- ★ You will get your take-home midterm exam next Wednesday, December 5th. It due is at 4pm on December 19th. Topics include:
  - N-gram language models
  - Lexical Semantics
  - Semantics
  - Dependency trees
  - TAGs
  - CCGs
  - Noisy channel model
  - Transformation-based learning
- ★ You will have six problems. We will drop the lowest score and grade your exam on the basis of the five best answers.



## *Important Notes about the Final*



\* You must be present in class next week to pick up your exam. You must also turn in your exam by the deadline by slipping it under my door in room AVW 3153 by 4pm on December 19<sup>th</sup>. WE WILL NOT ACCEPT A LATE EXAM!

\* Once the exam is handed out, you will not be able to use the web for searching on NLP topics or to ask us (or each other) about anything about the course topics!!! So it is important that you prepare all the materials you think you will need in advance. Specifically, the restriction on the exam is worded as follows:

- You are free to use the textbook, your notes, any reading specified in the Schedule of Topics, and any other hardcopy materials (even, say, another textbook).
- Once the exam is handed out, and until it is turned in, you are forbidden to search the Web for explanations, notes, tutorials, or any other materials. Nor are you permitted to follow links from allowed material to other material that has not been mentioned explicitly. Except as specified above, or added explicitly by the instructor, the only Web pages you are permitted to access in relation to this examination are the new textbook chapters at <http://www.cs.colorado.edu/~martin/slp2.html>.

**THIS MEANS NO WIKIPEDIA, FOR EXAMPLE!!**

