Computational Linguistics 1

CMSC/LING 723, LBSC 744

Kristy Hollingshead Seitz
Institute for Advanced Computer Studies
University of Maryland

Lecture 17: 1 November 2011

Agenda

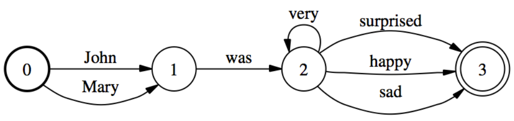
- HW4, due today!
- Questions, comments, concerns?
- Schedule changes on the syllabus
- Chomsky Hierarchy revisited
- Context-sensitive grammars
  - Unification
  - Tree-adjoining grammars (TAG)
  - Combinatory Categorial Grammars (CCG)

Chomsky Hierarchy

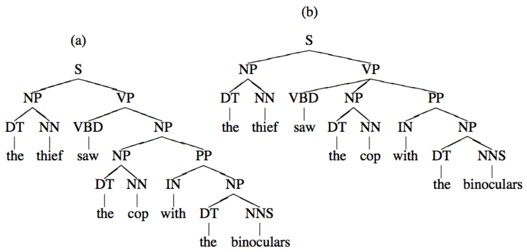
Language	Mechanisms	Examples
Regular	Regular expressions Regular grammars Finite-state automata Finite-state transducers WFSAs/WFSTs	xa^ny Morphology Phonology Taggers
Context-free	Context-free grammars (CFGs) Pushdown automata	a^nb^n Most syntax
Context-sensitive	**Unification grammars** Lexicalized formalisms (e.g., TAG, CCG)	$a^nb^mc^nd^m$ Cross-serial dependencies

---

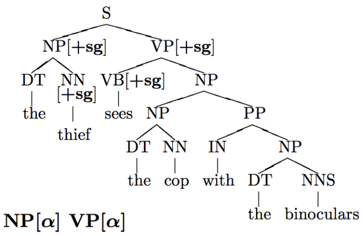
Finite-State



Context-Free



Context-Sensitive: Unification



$$\mathbf{S}[\alpha] \rightarrow \mathbf{NP}[\alpha]\ \mathbf{VP}[\alpha]$$

$$\mathbf{NP}[\alpha] \rightarrow \mathbf{DT}\ \mathbf{NN}[\alpha]$$

$$\mathbf{VP}[\alpha] \rightarrow \mathbf{VB}[\alpha]\ \mathbf{NP}$$

the thieves see …

Unification: Feature Structures

$$\begin{bmatrix} \text{FEATURE}_1 & \text{VALUE}_1 \\ \text{FEATURE}_2 & \text{VALUE}_2 \\ \vdots & \vdots \\ \text{FEATURE}_n & \text{VALUE}_n \end{bmatrix}$$

e.g.,

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix}$$

Feature Structures

- What do feature structures provide?
- A mechanism to bring lexical features to bear on syntactic structure
- A formal mechanism for handling how smaller constituents combine to form larger constituents
- A mechanism to enforce constraints on syntactic structures, e.g.,
  - Agreement
  - Grammatical heads
  - Subcategorization
  - Long-distance dependencies

Feature Structures as Values

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{CAT} & \text{S} \\ \text{HEAD} & \begin{bmatrix} \text{AGREEMENT} & (1) \begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & (1) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Unification (⊔)

- Unification (⊔) is an operation on feature sets
- Matches in values succeed; mismatches fail
- Feature values can be underspecified
- Unification with an underspecified value forces a match, e.g.,

$$\begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix} \sqcup \begin{bmatrix} \text{NUMBER} & [\,] \end{bmatrix} = \begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix}$$

- Features not explicitly represented are underspecified

$$\begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix} \sqcup \begin{bmatrix} \text{NUMBER} & \text{SG} \end{bmatrix} = \begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix}$$

Underspecification: Example

- Consider the noun “sheep”, which is either plural or singular
- In other words, the category as a subject noun will be

$$\begin{bmatrix} \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & [\,] \end{bmatrix} \end{bmatrix}$$

- Then plural verbs like *are* will force a plural unification, and singular verbs like *is* a singular unification

e.g., *The goshdern sheep are chasing my dog* versus
The goshdern sheep is chasing my dog

More complicated unification

$$\begin{bmatrix} \text{AGREEMENT} & (1) \begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & (1) \end{bmatrix} \end{bmatrix}$$

$$\sqcup \begin{bmatrix} \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \text{AGREEMENT} & (1) \begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{PERSON} & 3 \end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & (1) \end{bmatrix} \end{bmatrix}$$

"Copying" via Unification

- What if we don't yet know values, but know they should match?
  - e.g., an S node: NP and VP may be either singular or plural, but should definitely match

$$\begin{bmatrix} \text{AGREEMENT} & (1) \\ \text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & (1)\end{bmatrix} \end{bmatrix}$$

$$\sqcup \begin{bmatrix} \text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & \begin{bmatrix}\text{NUMBER} & \text{SG} \\ \text{PERSON} & 3\end{bmatrix}\end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \text{AGREEMENT} & (1) \\ \text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & (1)\begin{bmatrix}\text{NUMBER} & \text{SG} \\ \text{PERSON} & 3\end{bmatrix}\end{bmatrix} \end{bmatrix}$$

Failed Unification

$$\begin{bmatrix} \text{AGREEMENT} & (1)\begin{bmatrix}\text{NUMBER} & \text{SG} \\ \text{PERSON} & 3\end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & (1)\end{bmatrix} \end{bmatrix}$$

$$\sqcup \begin{bmatrix} \text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & \begin{bmatrix}\text{NUMBER} & \text{PL} \\ \text{PERSON} & 3\end{bmatrix}\end{bmatrix} \end{bmatrix}$$

= Failure

Feature Structures

- What do feature structures provide?
- A mechanism to bring lexical features to bear on syntactic structure
- A formal mechanism for handling how smaller constituents combine to form larger constituents
- A mechanism to enforce constraints on syntactic structures, e.g.,
  - Agreement
  - Grammatical heads
  - Subcategorization
  - Long-distance dependencies

Features Example: S-node Agreement

- S → NP VP

$$\begin{bmatrix} \text{CAT} & \text{S} \\ \text{HEAD} & \begin{bmatrix}\text{AGREEMENT} & (1) \\ \text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & (1)\end{bmatrix}\end{bmatrix} \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{AGREEMENT} & (1) \end{bmatrix} \begin{bmatrix} \text{CAT} & \text{VP} \\ \text{AGREEMENT} & (1) \end{bmatrix}$$

- e.g., *This flight serves breakfast* or *These flights serve breakfast* not *This flight serve breakfast* or *These flights serves breakfast*

Features Example: NP Agreement

- NP → DT Noun

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{AGREEMENT} & (1) \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT} & \text{DT} \\ \text{AGREEMENT} & (1) \end{bmatrix} \begin{bmatrix} \text{CAT} & \text{Noun} \\ \text{AGREEMENT} & (1) \end{bmatrix}$$

$$\begin{bmatrix} \text{CAT} & \text{DT} \\ \text{AGREEMENT} & \text{SG} \end{bmatrix} \rightarrow \text{this} \qquad \begin{bmatrix} \text{CAT} & \text{Noun} \\ \text{AGREEMENT} & \text{SG} \end{bmatrix} \rightarrow \text{flight}$$

$$\begin{bmatrix} \text{CAT} & \text{DT} \\ \text{AGREEMENT} & \text{PL} \end{bmatrix} \rightarrow \text{these} \qquad \begin{bmatrix} \text{CAT} & \text{Noun} \\ \text{AGREEMENT} & \text{PL} \end{bmatrix} \rightarrow \text{flights}$$

- e.g., *this flight* or *these flights* not *this flights* or *these flight*
- How would the determiner "*the*" be categorized? (SG or PL)

Features: Heads

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{AGREEMENT} & (1) \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT} & \text{DT} \\ \text{AGREEMENT} & (1) \end{bmatrix} \begin{bmatrix} \text{CAT} & \text{Noun} \\ \text{AGREEMENT} & (1) \end{bmatrix}$$

- Features for most categories are copied from one child, known as the *head* child
- Put AGREEMENT features under HEAD feature, and copy it all:

$$\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{HEAD} & (1) \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT} & \text{DT} \\ \text{AGREEMENT} & (2) \end{bmatrix} \begin{bmatrix} \text{CAT} & \text{Noun} \\ \text{HEAD} & (1)\begin{bmatrix}\text{AGREEMENT} & (2)\end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{CAT} & \text{VP} \\ \text{HEAD} & (1) \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT} & \text{Verb} \\ \text{HEAD} & (1)\begin{bmatrix}\text{AGREEMENT} & (2)\end{bmatrix} \end{bmatrix} \begin{bmatrix} \text{CAT} & \text{NP} \\ \text{AGREEMENT} & (3) \end{bmatrix}$$

Head Constituents

- A common notion in both Linguistics and NLP is the *head* constituent, i.e., most important or driving constituent
- Example: in English, VP tends to be head of S
- Can define a recursive relation, down to lexical heads
- (S (NP The dog) (VP (VBD bit) (NP the mailman))):
  - the main verb is the head of the VP
  - the VP is the head of the S
  - thus "*bit*" is the lexical head of the S
  - final noun is typically considered head of NP (*dog* and *mailman*) although some linguists argue for the determiner to be head (DP)

Subcategorization

- Like the notion of a head child, subcategorization is a widespread idea
- Certain verbs require/allow certain arguments, e.g.,

• *give* NP NP	*give the library the book*
• *give* NP PP	*give the book to the library*
• *donate* NP PP	*donate the book to the library*
* *donate* NP NP	*donate the library the book*

- These are *syntactic* constraints
- Semantic constraints are called *selectional restrictions* e.g., *eat* selects for edible objects
  - "Fuzzier" restrictions, more easily violated

Subcategorization using Unification

$$\begin{bmatrix}\text{CAT} & \text{VP}\\ \text{HEAD} & (1)\end{bmatrix} \rightarrow \begin{bmatrix}\text{CAT} & \text{Verb}\\ \text{HEAD} & (1)\begin{bmatrix}\text{SUBCAT} & \text{INTRANS}\end{bmatrix}\end{bmatrix}$$

$$\begin{bmatrix}\text{CAT} & \text{VP}\\ \text{HEAD} & (1)\end{bmatrix} \rightarrow \begin{bmatrix}\text{CAT} & \text{Verb}\\ \text{HEAD} & (1)\begin{bmatrix}\text{SUBCAT} & \text{TRANS}\end{bmatrix}\end{bmatrix}\begin{bmatrix}\text{CAT} & \text{NP}\end{bmatrix}$$

$$\begin{bmatrix}\text{CAT} & \text{VP}\\ \text{HEAD} & (1)\end{bmatrix} \rightarrow \begin{bmatrix}\text{CAT} & \text{Verb}\\ \text{HEAD} & (1)\begin{bmatrix}\text{SUBCAT} & \text{DITRANS}\end{bmatrix}\end{bmatrix}\begin{bmatrix}\text{CAT} & \text{NP}\end{bmatrix}\begin{bmatrix}\text{CAT} & \text{NP}\end{bmatrix}$$

$$\begin{bmatrix}\text{CAT} & \text{Verb}\\ \text{HEAD} & \begin{bmatrix}\text{SUBCAT} & \text{INTRANS}\end{bmatrix}\end{bmatrix} \rightarrow \textit{died} \qquad (1)$$

$$\begin{bmatrix}\text{CAT} & \text{Verb}\\ \text{HEAD} & \begin{bmatrix}\text{SUBCAT} & \text{DITRANS}\end{bmatrix}\end{bmatrix} \rightarrow \textit{gave} \qquad (2)$$

Long-distance Dependencies

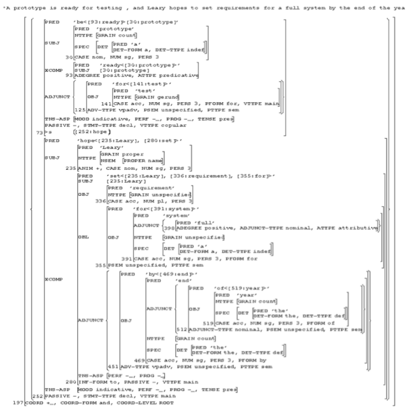
- Now that there are subcategorization constraints, a verb had better get its arguments
- What about the following: *you give the book*
  No good as a stand-alone sentence
  (infinitive verb, missing an argument)
- *To which library did you give the book?*
- Need some mechanism for allowing argument *gaps*
- These dependencies can be quite distant
  Which flight do you want me to have the travel agent book?

Existing Unification Approaches

- Lexical Functional Grammar (LFG)
  Bresnan and Kaplan (1982)
- Generalized Phrase Structure Grammar (GPSG)
  Gazdar, Klein, Pullum and Sag (1985)
- Head-Driven Phrase Structure Grammar (HPSG)
  Pollard and Sag (1994)

- Feature structures have found their way into other approaches
- Ideas like head children and subcategorization are widespread

LFG structure, from Riezler et al., 2003



Agenda

- HW4, due today!
- Questions, comments, concerns?
- Schedule changes on the syllabus
- Chomsky Hierarchy revisited
- Context-sensitive grammars
  - Unification
  - Tree-adjoining grammars (TAG)
  - Combinatory Categorial Grammars (CCG)

Tree-adjoining Grammars (TAG)

- Initial, auxiliary and elementary trees
- Substitution and Adjunction
- Derived and derivation trees

Tree-adjoining Grammars

A Tree-adjoining grammar (TAG) G = (V, T, $S^\dagger$, I, A)

- a set of non-terminal variables V
- a set of terminals T
- a special start symbol $S^\dagger \in V$
- a set of initial trees I
  - Non-terminals on frontier marked for substitution
- a set of auxiliary trees A
  - One non-terminal on frontier marked as *foot* node
  - Otherwise like initial trees

Elementary trees (slide taken from Joshi & Schabes, 1997)

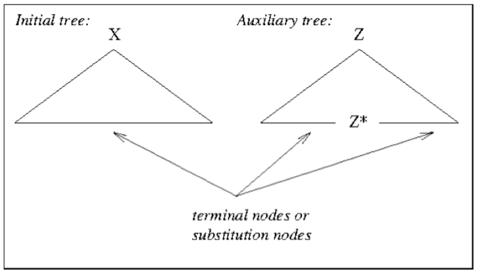
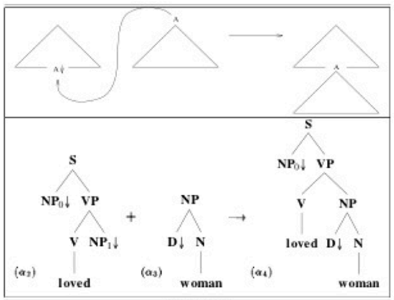


Fig. 2.1. Schematic initial and auxiliary trees.

TAGs

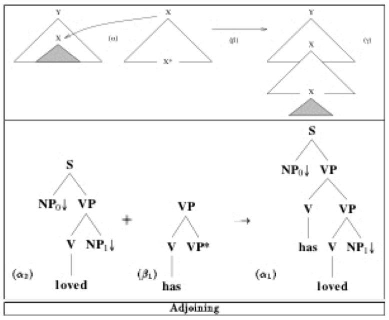
- Elementary trees are of type X where X is the root category
- Foot node must be of same category as the root
- Lexicalized TAG (LTAG) requires at least one terminal item (the *anchor*) on every elementary tree
- Two operations defined on trees
  - Substitution
  - Adjunction

Substitution (slide taken from Joshi & Schabes, 1997)



Substitution

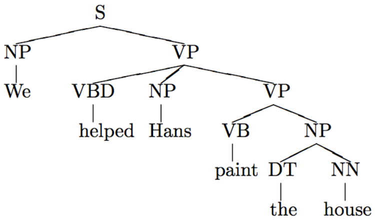
Adjunction (slide taken from Joshi & Schabes, 1997)



Derivation

- Derivation begins with a set of elementary trees like a set of rules in a CFG
- Trees can combine via substitution or adjunction
- Can define a “derives” relation, as with CFGs
- A string is in the “language” if there is a sequence of derives steps from the root symbol to a tree with the terminals at the frontier
- TAGs can generate cross-serial dependencies
- Derivation results in two trees:
  - derived tree
  - derivation trees

Context-Sensitive: Cross-serial Dependencies

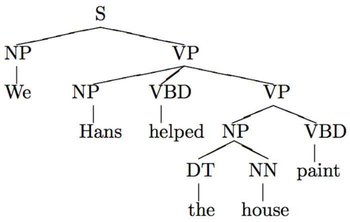


We helped Hans paint the house

We Hans helped the house paint

We Hans the house helped paint

Context-Sensitive: Cross-serial Dependencies

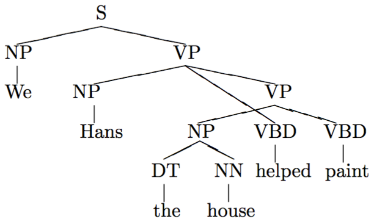


We helped Hans paint the house

We Hans helped the house paint

We Hans the house helped paint

Context-Sensitive: Cross-serial Dependencies



We helped Hans paint the house

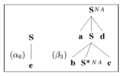
We Hans helped the house paint

We Hans the house helped paint

Cross-serial Dependencies in TAG

(slide taken from Joshi & Schabes, 1997)

- *Example 2.2.* Consider the TAG $G_2 = (\{a, b, c, d, e\}, \{S\}, \{\alpha_6\}, \{\beta_3\}, S)$ below



G_2 generates the context-sensitive language $L_2 = \{a^n b^n e c^n d^n | n \geq 1\}$. For example, in Fig. 2.7, α_9 has been obtained by adjoining β_3 on the root node of α_6 and α_{10} has been obtained by adjoining β_3 on the node at address 2 in in the tree α_9.

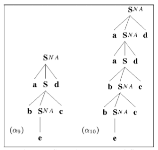


Fig. 2.7. Some derived trees of G_2

Elementary trees (slide taken from Joshi & Schabes, 1997)

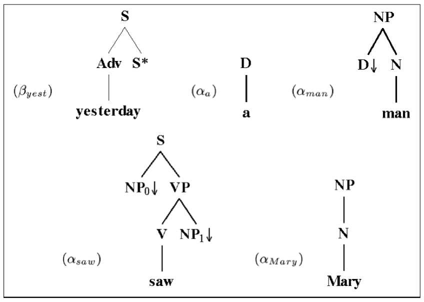


Fig. 2.4. Some elementary trees.

Derived tree (slide taken from Joshi & Schabes, 1997)

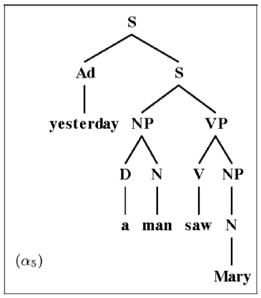


Fig. 2.3. Derived tree for: *yesterday a man saw Mary.*

Derivation tree (slide taken from Joshi & Schabes, 1997)

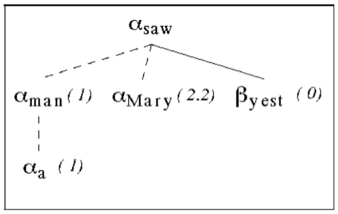
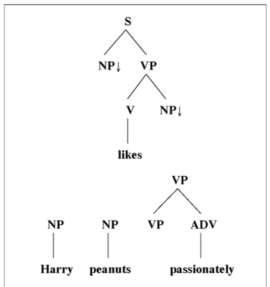


Fig. 2.5. Derivation tree for *Yesterday a man saw Mary.*

Example (slide taken from Joshi & Schabes, 1997)



Operations: (1) Substitution (for nodes marked with ↓). (2) Adjoining.

Fig. 8.2. Elementary Trees for a TAG, G_1.

Properties of TAGs

- TAGs are quite interesting formally, and a lot of work in formal language theory has been done
- The *derivation* trees are context-free, i.e. the derivation sequences form a context-free language
- There is a kind of pushdown automata that is weakly equivalent to TAGs
- Parsing complexity O(n^6) compared to O(n^3) for CFG

Agenda

- HW4, due today!
- Questions, comments, concerns?
- Schedule changes on the syllabus
- Chomsky Hierarchy revisited
- Context-sensitive grammars
  - Unification
  - Tree-adjoining grammars (TAG)
  - Combinatory Categorial Grammars (CCG)
- Homework 5 online tonight

End of lecture, 3 Nov.

Agenda

- HW4, due today!
- Questions, comments, concerns?
- Schedule changes on the syllabus
- Chomsky Hierarchy revisited
- Context-sensitive grammars
  - Unification
  - Tree-adjoining grammars (TAG)
  - Combinatory Categorial Grammars (CCG)

Categorial Grammars

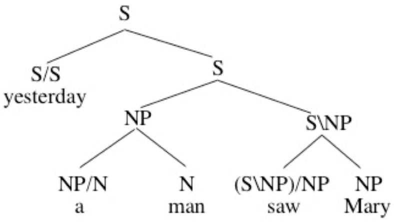
- Approach has been around since the 50s (Bar-Hillel and Lambek)
- Closely tied to the Formal Semantics of Montague using lambda calculus
- TAG and CG have the same generative power: every TAG grammar has a weakly equivalent CG grammar (and vice versa)
- Notion of strong compositionality: syntactic structure and interpretation are derived in lockstep

---

Lexical Categories and Function Application

- Every word in the lexicon is associated with a complex grammatical category
- Two function application schemas, describing how to combine two categories to form a new category
  - X/Y Y ⇒X
  - Y X\Y ⇒X
- Interpret X/Y as requiring a Y on the right to make an X
- Interpret X\Y as requiring a Y on the left to make an X
- Example: *the* is of category NP/N; *big* is N/N; *car* is N
- These categories are similar to elementary trees in TAG

CCG Example



Type-Lifting

- In function application there is a category that needs something and a category that fills that need
  e.g., NP S\NP ⇒ S
  (S\NP *needs* an NP category on its left to give an S)
- One might also argue that an NP category needs an S\NP category on its right to give an S
- Something that needs an S\NP on its right to give an S is S/(S\NP)
- Type lifting converts a category X to Y/(Y\X) or Y\(Y/X) for an arbitrary Y

Function Composition

- Consider *John saw Mary*
- *John* and *Mary* are of category NP
  saw is of category (S\NP)/NP
- We can type-lift *John* to S/(S\NP) giving:

John	*saw*	*Mary*
NP	(S\NP)/NP	NP
S/(S\NP)	(S\NP)/NP	NP

- S/(S\NP) needs something to its right that (S\NP)/NP will provide, once it gets an NP to its right
- Function composition allows these to combine as follows:
  X/Y Y/Z ⇒X/Z   or   X\Y Z\X ⇒Z\Y

Non-constituent Coordination

- With type-lifting and function composition, Categorial grammar gets non-constituent coordination for free by defining *and* as (α/α)\α
- Example: *John fetched and Mary read the paper*

John	*fetched*	*and*	*Mary*	*read*	*the*	*paper*
NP	(S\NP)/NP	$(\alpha/\alpha)\backslash\alpha$	NP	(S\NP)/NP	NP/N	N
S/(S\NP)	(S\NP)/NP	$(\alpha/\alpha)\backslash\alpha$	S/(S\NP)	(S\NP)/NP	NP/N	N
S/NP		((S/NP)/(S/NP))\(S/NP)	S/NP		NP	
(S/NP)/(S/NP)			S/NP		NP	
S/NP					NP	
S						

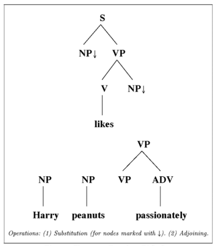
Compositional Semantics

- Great selling point: semantic categories associated with syntactic categories
- Lambda calculus provides natural formalism for deriving meaning of a constituent from the meaning of its children
- All operations discussed here have semantic correlate
  - Function application
  - Type lifting
  - Function composition

Lexicalized Grammar Formalisms

Tree Adjoining Grammar

substitution and adjunction



Operations: (1) Substitution (for nodes marked with ↓). (2) Adjoining.

Fig. 8.2. Elementary Trees for a TAG, G_1.

Categorial Grammar

function application and composition, type-lifting

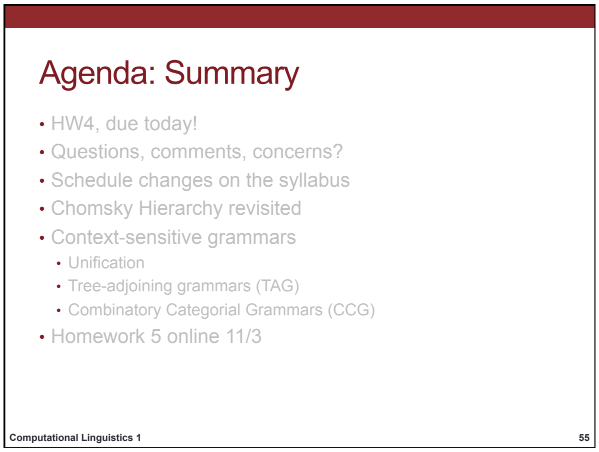
likes	(S\NP)/NP
Harry	NP
peanuts	NP
passionately	(S\NP)\(S\NP)

Lexicalized Grammar Formalisms

- TAG and CG are known as lexicalized grammars
- Have been shown to have weakly-equivalent generative capacity
- Lexical categories, not rules, specify how words combine
- Subcategorization is handled by the lexical categories of verbs
- Natural notion of lexical heads, also based on lexical categories
- Clear that many of the dependencies are lexical

Statistical Approaches

- As with unification grammars, each of these have had statistical approaches
- Some of the statistical approaches have involved finite-state approximations
  - “Supertagging” involves building a POS-tagger, with full TAG style lexical categories
- Others involve log-linear models
- Many statistical context-free parsing approaches are influenced by these formalisms and unification
  - Weighted, not categorical, constraints



Agenda: Summary

- HW4, due today!
- Questions, comments, concerns?
- Schedule changes on the syllabus
- Chomsky Hierarchy revisited
- Context-sensitive grammars
  - Unification
  - Tree-adjoining grammars (TAG)
  - Combinatory Categorial Grammars (CCG)
- Homework 5 online 11/3