CSC 2517: Discrete Mathematical Models of Sentence Structure

- Gerald Penn, PT 283A,
  gpenn@cs.toronto.edu,
  Tel: (416)978-7390
- This class will meet during A&S reading week!
- But we will not meet on Wednesday, 22nd March.

CSC 2517: Discrete Mathematical Models of Sentence Structure

- This is an advanced graduate seminar:
  - I will assume that you are familiar with the material of CSC 2501, although graduate seminars do not formally enforce prerequisites.
  - No programming assignments, although your final paper may involve some.
  - Classes will hopefully be more interactive than normal lectures.
  - ***You*** will do much of the presenting.
- If any of this doesn't sound like what you signed up for, then you probably belong in CSC 2511, Natural Language Computing, which is also being offered this term.

CSC 2517: Discrete Mathematical Models of Sentence Structure

- This year, presentations will be of papers chosen from among the following topics:
  - Algebraic Topology
  - Algebraic Invariance
  - Graphical/Algebraic Methods for Natural Language
  - Geometric Deep Learning

CSC 2517: Discrete Mathematical Models of Sentence Structure

- How I will compute your final mark for the class:
  - 30% your presentation(s) and participation in the seminar.
  - 70% a final paper, due on Friday, 28th April.
  - Paper proposals are due on Tuesday, 14th March.
  - Auditors are welcome, but they must present, just like everyone else.
  - Group papers must be approved in advance. If I approve, everyone in the group will receive the same mark.
  - Your final paper must be on the subject of mathematical linguistics, broadly construed. It needn't be on one of the presentation topics.
  - You may submit research you are conducting as part of your thesis or dissertation.
  - This is a Methods Area 1 class. In terms of length and style, think of your final papers as MOL or WoLLIC conference papers. Actually submitting to such a conference is encouraged but not required.

CSC 2517: Discrete Mathematical Models of Sentence Structure

- Rules for presentations:
  - Students will pick topics two weeks in advance of their presentation date.
  - I reserve the right to reject papers on the grounds that they are:
    - unsuitably difficult,
    - unsuitably bad,
    - insufficiently related to the topics of this seminar, or
    - excessively devoted to material already covered.
  - Unless invited, you may not present your own research.
  - I will also be offering pre-approved paper(s) to present.

CSC 2517: Discrete Mathematical Models of Sentence Structure

- More rules for presentations:
  - The papers that we will be reading are highly technical. Expect your presentations to be ≈1 hour long, or 1 hour of a 2-hour presentation that you will jointly give with another student (for longer papers and collections of related papers).
  - Your job as presenter is to *teach* the material. That not only includes the research presented in the paper, but the background material necessary to understand it.
    - Assume that no one has read the paper or understands what we have not discussed in class yet.
  - Your job as a non-presenter is to refute this assumption: read the paper, look up the background you are missing and come to class with questions.
  - Presenters must defend the work that they are presenting.
- Think of these as opportunities not to present a conference paper, but to teach a class – it's good practice.

Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG)

- Categorial grammar (CG) is one of the oldest grammar formalisms
- *Combinatory* Categorial Grammar now well established and computationally well founded (Steedman, 1996, 2000)
  - Account of syntax; semantics; prosody and information structure; automatic parsers; generation

Combinatory Categorial Grammar (CCG)

- CCG is a lexicalized grammar
- An elementary syntactic structure – for CCG a lexical category – is assigned to each word in a sentence

  walked: S\NP "give me an NP to my left and I return a sentence"
- A small number of rules define how categories can combine
  - Rules based on the combinators from Combinatory Logic

CCG Lexical Categories

- Atomic categories: S , N , NP , PP , . . . (not many more)
- Complex categories are built recursively from atomic categories and slashes, which indicate the directions of arguments
- Complex categories encode subcategorisation information
  - intransitive verb: S \NP *walked*
  - transitive verb: (S \NP)/NP *respected*
  - ditransitive verb: ((S \NP)/NP)/NP *gave*
- Complex categories can encode modification
  - PP nominal: (NP \NP)/NP
  - PP verbal: ((S \NP)\(S \NP))/NP

Simple CCG Derivation

$$
\begin{array}{ccc}
\mathit{interleukin-10} & \mathit{inhibits} & \mathit{production} \\
\hline
NP & (S\backslash NP)/NP & NP \\
 & \multicolumn{2}{c}{\rule{8em}{0.4pt}>} \\
 & \multicolumn{2}{c}{S\backslash NP} \\
\multicolumn{3}{c}{\rule{16em}{0.4pt}<} \\
\multicolumn{3}{c}{S}
\end{array}
$$

$>$ forward application

$<$ backward application

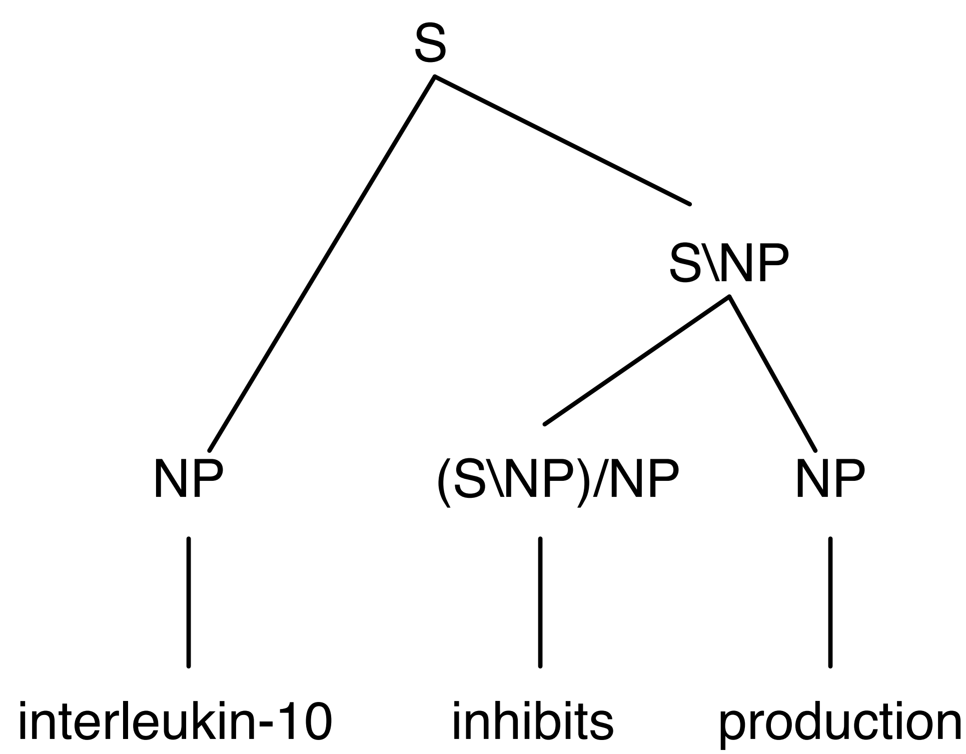
Function Application Schemata

- Forward ($>$) and backward ($<$) application:

$$\begin{array}{llll} X/Y \quad Y & \Rightarrow & X & (>) \\ Y \quad X\backslash Y & \Rightarrow & X & (<) \end{array}$$

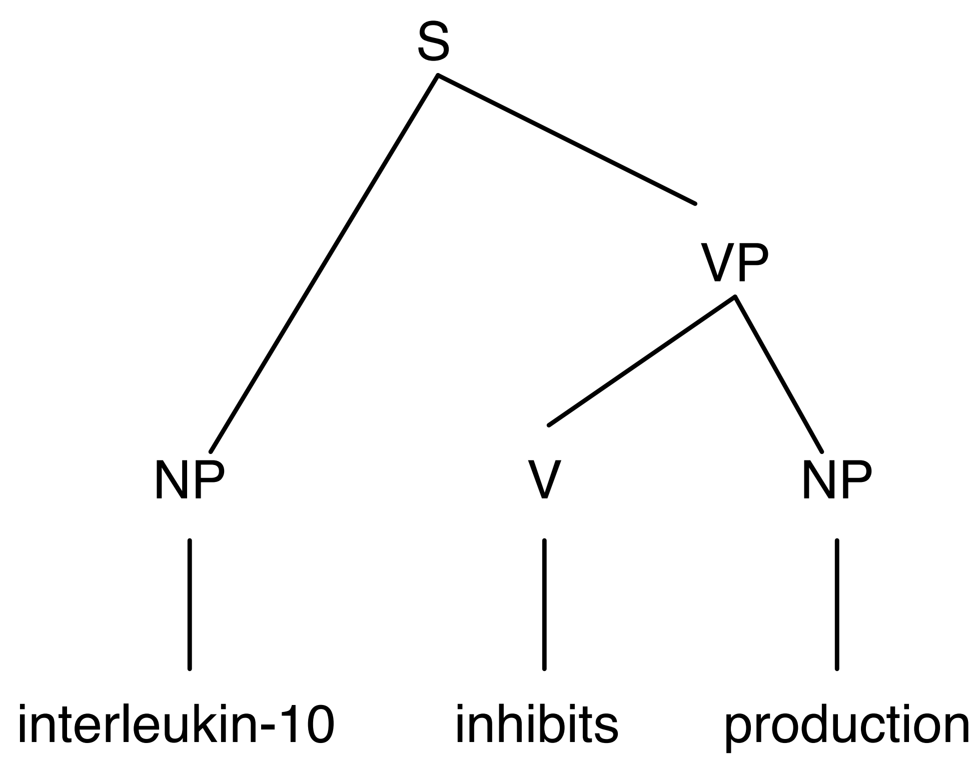
Classical Categorial Grammar

- 'Classical' Categorial Grammar only has application rules
- Classical Categorial Grammar is context free



Classical Categorial Grammar

- 'Classical' Categorial Grammar only has application rules
- Classical Categorial Grammar is context free



Extraction out of a Relative Clause

$$\frac{\mathit{The}}{NP/N} \quad \frac{\mathit{company}}{N} \quad \frac{\mathit{which}}{(NP\backslash NP)/(S/NP)} \quad \frac{\mathit{Microsoft}}{NP} \quad \frac{\mathit{bought}}{(S\backslash NP)/NP}$$

Extraction out of a Relative Clause

$$\begin{array}{ccccc}
\textit{The} & \textit{company} & \textit{which} & \textit{Microsoft} & \textit{bought} \\
\overline{NP/N} & \overline{N} & \overline{(NP\backslash NP)/(S/NP)} & \overline{NP} & \overline{(S\backslash NP)/NP} \\
 & & & \overline{S/(S\backslash NP)}^{\,>\mathbf{T}} &
\end{array}$$

$>\mathbf{T}$ type-raising

Extraction out of a Relative Clause

$$
\begin{array}{ccccc}
\textit{The} & \textit{company} & \textit{which} & \textit{Microsoft} & \textit{bought} \\
\hline
NP/N & N & (NP\backslash NP)/(S/NP) & NP & (S\backslash NP)/NP \\
 & & & \overline{S/(S\backslash NP)}^{>\mathbf{T}} & \\
 & & & \multicolumn{2}{c}{\overline{S/NP}^{>\mathbf{B}}}
\end{array}
$$

$>\mathbf{T}$ type-raising

$>\mathbf{B}$ forward composition

Extraction out of a Relative Clause

$$
\begin{array}{ccccc}
\textit{The} & \textit{company} & \textit{which} & \textit{Microsoft} & \textit{bought} \\
\hline
NP/N & N & (NP\backslash NP)/{\color{blue}(S/NP)} & NP & (S\backslash NP)/NP \\
\cline{4-4}
 & & & >\mathbf{T} & \\
 & & & S/(S\backslash NP) & \\
\cline{4-5}
 & & & \multicolumn{2}{r}{>\mathbf{B}} \\
 & & & \multicolumn{2}{c}{{\color{blue}S/NP}} \\
\cline{3-5}
 & & \multicolumn{3}{r}{>} \\
 & & \multicolumn{3}{c}{NP\backslash NP}
\end{array}
$$

Extraction out of a Relative Clause

$$\begin{array}{ccccc}
\mathit{The} & \mathit{company} & \mathit{which} & \mathit{Microsoft} & \mathit{bought} \\
\hline
NP/N & N & (NP\backslash NP)/(S/NP) & NP & (S\backslash NP)/NP
\end{array}$$

$NP/N \;\; N \;\Rightarrow_{>}\; NP$

$NP \;\Rightarrow_{>\mathbf{T}}\; S/(S\backslash NP)$

$S/(S\backslash NP) \;\; (S\backslash NP)/NP \;\Rightarrow_{>\mathbf{B}}\; S/NP$

$(NP\backslash NP)/(S/NP) \;\; S/NP \;\Rightarrow_{>}\; NP\backslash NP$

$NP \;\; NP\backslash NP \;\Rightarrow_{<}\; NP$

Forward Composition and Type-Raising

- Forward composition ($>_{\mathbf{B}}$):

$$X/Y \quad Y/Z \quad \Rightarrow \quad X/Z \qquad (>_{\mathbf{B}})$$

- Type-raising ($\mathbf{T}$):

$$X \quad \Rightarrow T/(T\backslash X) \qquad (>_{\mathbf{T}})$$
$$X \quad \Rightarrow T\backslash(T/X) \qquad (<_{\mathbf{T}})$$

- Extra combinatory rules increase the weak generative power to mild context -sensitivity

"Non-constituents" in CCG – Right Node Raising

$$\begin{array}{cccccc}
\textit{Google} & \textit{sells} & \textit{but} & \textit{Microsoft} & \textit{buys} & \textit{shares} \\
\hline
NP & (S\backslash NP)/NP & conj & NP & (S\backslash NP)/NP & NP \\
\overline{S/(S\backslash NP)}^{>\mathbf{T}} & & & \overline{S/(S\backslash NP)}^{>\mathbf{T}} & &
\end{array}$$

$>\mathbf{T}$ type-raising

"Non-constituents" in CCG – Right Node Raising

$$
\begin{array}{cccccc}
\textit{Google} & \textit{sells} & \textit{but} & \textit{Microsoft} & \textit{buys} & \textit{shares}\\
\hline
NP & (S\backslash NP)/NP & conj & NP & (S\backslash NP)/NP & NP \\
\hline
S/(S\backslash NP) \; {>}\mathbf{T} & & & S/(S\backslash NP) \; {>}\mathbf{T} & & \\
\hline
\multicolumn{2}{c}{S/NP \; {>}\mathbf{B}} & & \multicolumn{2}{c}{S/NP \; {>}\mathbf{B}} &
\end{array}
$$

$>$ **T** type-raising
$>$ **B** forward composition

"Non-constituents" in CCG – Right Node Raising

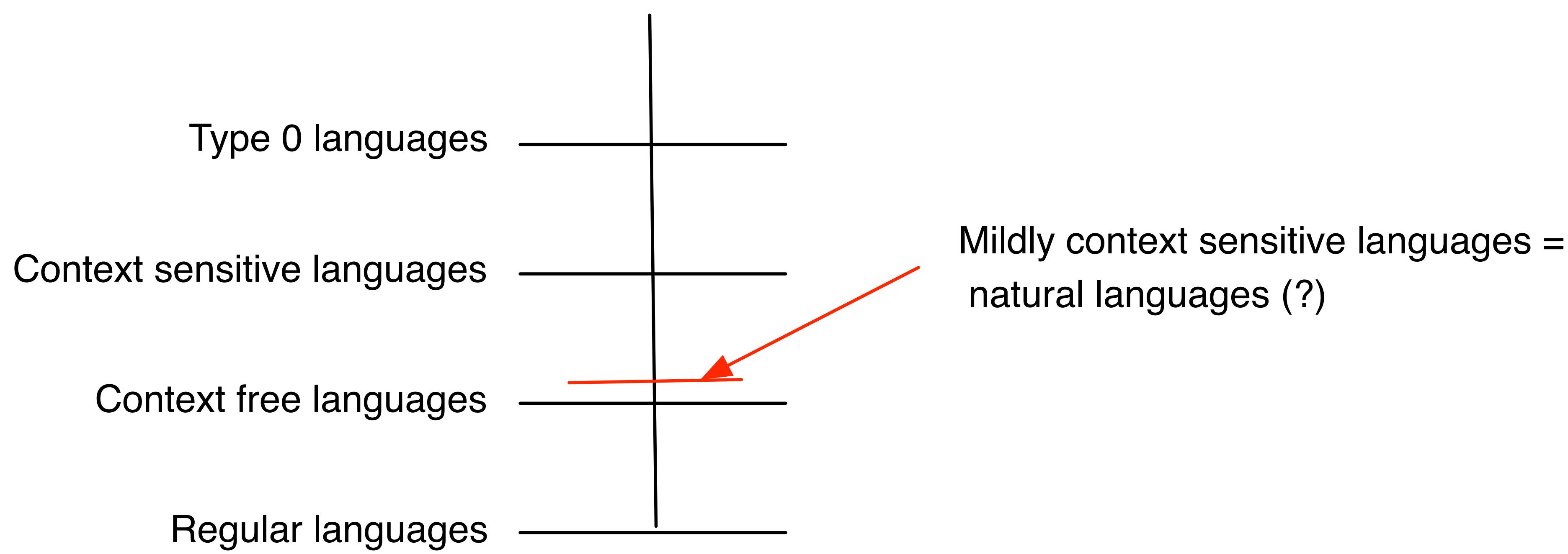
$$
\begin{array}{cccccc}
\textit{Google} & \textit{sells} & \textit{but} & \textit{Microsoft} & \textit{buys} & \textit{shares} \\
\hline NP & (S\backslash NP)/NP & conj & NP & (S\backslash NP)/NP & NP \\
\hline S/(S\backslash NP) \; {>}\mathbf{T} & & & S/(S\backslash NP) \; {>}\mathbf{T} & & \\
\multicolumn{2}{c}{S/NP \; {>}\mathbf{B}} & & \multicolumn{2}{c}{S/NP \; {>}\mathbf{B}} & \\
\multicolumn{5}{c}{S/NP \; {<}\Phi{>}} &
\end{array}
$$

“Non-constituents” in CCG – Right Node Raising

$$
\begin{array}{cccccc}
\textit{Google} & \textit{sells} & \textit{but} & \textit{Microsoft} & \textit{buys} & \textit{shares} \\
\hline
NP & (S\backslash NP)/NP & conj & NP & (S\backslash NP)/NP & \color{blue}{NP} \\
S/(S\backslash NP)\ {>}\mathbf{T} & & & S/(S\backslash NP)\ {>}\mathbf{T} & & \\
\multicolumn{2}{c}{S/NP\ {>}\mathbf{B}} & & \multicolumn{2}{c}{S/NP\ {>}\mathbf{B}} & \\
\multicolumn{5}{c}{S/\color{blue}{NP}\ {<}\Phi{>}} & \\
\multicolumn{6}{c}{S\ {>}}
\end{array}
$$

Combinatory Categorial Grammar

- CCG is *mildly* context sensitive
- Natural language is provably non-context free
- Constructions in Dutch and Swiss German (Shieber, 1985) require more than context free power for their analysis
  - these have *crossing* dependencies (which CCG can handle)



CCG Semantics

- Categories encode argument sequences
- Parallel syntactic combinator operations and lambda calculus semantic operations

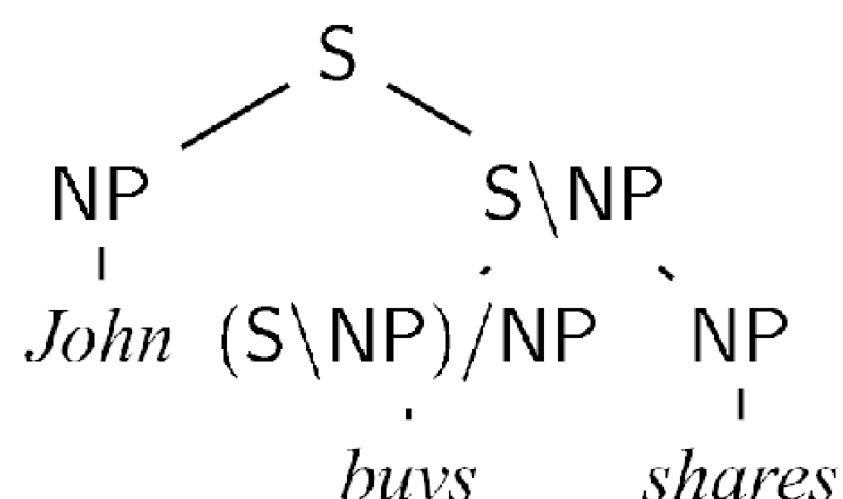
$John \vdash \mathsf{NP} : john'$

$shares \vdash \mathsf{NP} : shares'$

$buys \vdash (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.buys' xy$

$sleeps \vdash \mathsf{S}\backslash\mathsf{NP} : \lambda x.sleeps' x$

$well \vdash (\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP}) : \lambda f.\lambda x.well'(fx)$



CCG Semantics

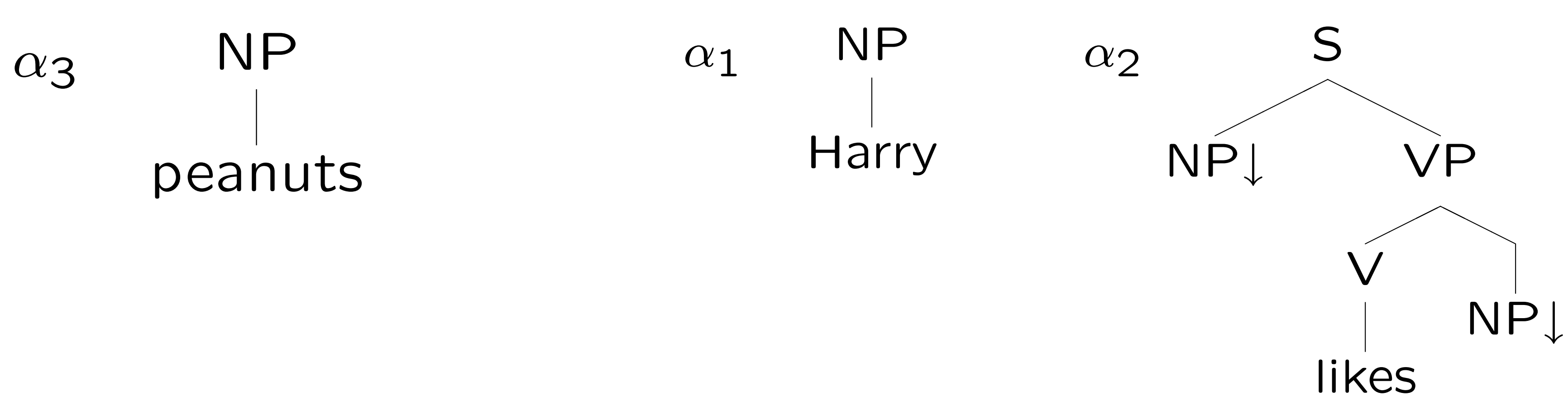
Left arg.	Right arg.	Operation	Result
X/Y : f	Y : a	Forward application	X : f(a)
Y : a	X\Y : f	Backward application	X : f(a)
X/Y : f	Y/Z : g	Forward composition	X/Z : λx.f(g(x))
X : a		Type raising	T/(T\X) : λf.f(a)

etc.

Tree Adjoining Grammar

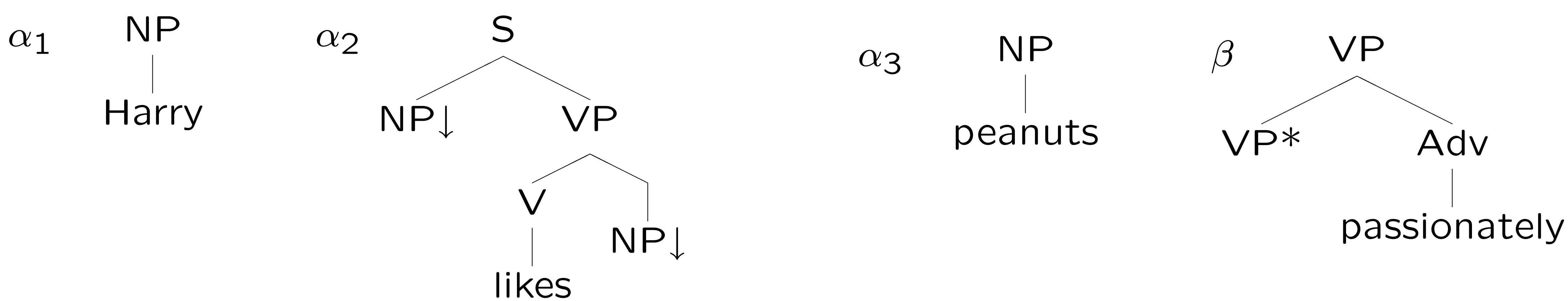
TAG Building Blocks

- Elementary trees (of many depths)
- Substitution at ↓
- Tree *Substitution* Grammar equivalent to CFG

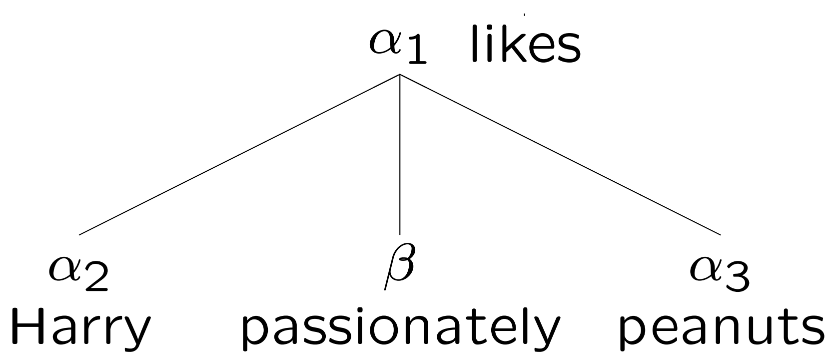


TAG Building Blocks

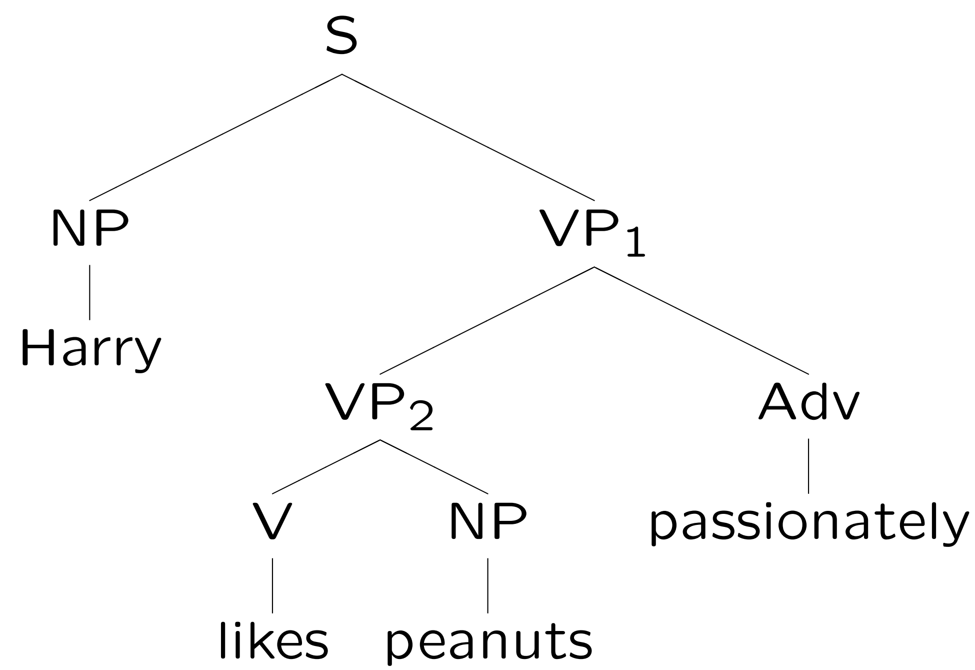
- Auxiliary trees for *adjunction*
- Adds extra power beyond CFG



Derivation Tree



Derived Tree



Semantics

$$Harry(x) \wedge likes(e, x, y) \wedge peanuts(y) \wedge passionately(e)$$

Why supertag?

- If lexical items have more description associated with them, parsing is easier
  - Only useful if the supertag space is not huge

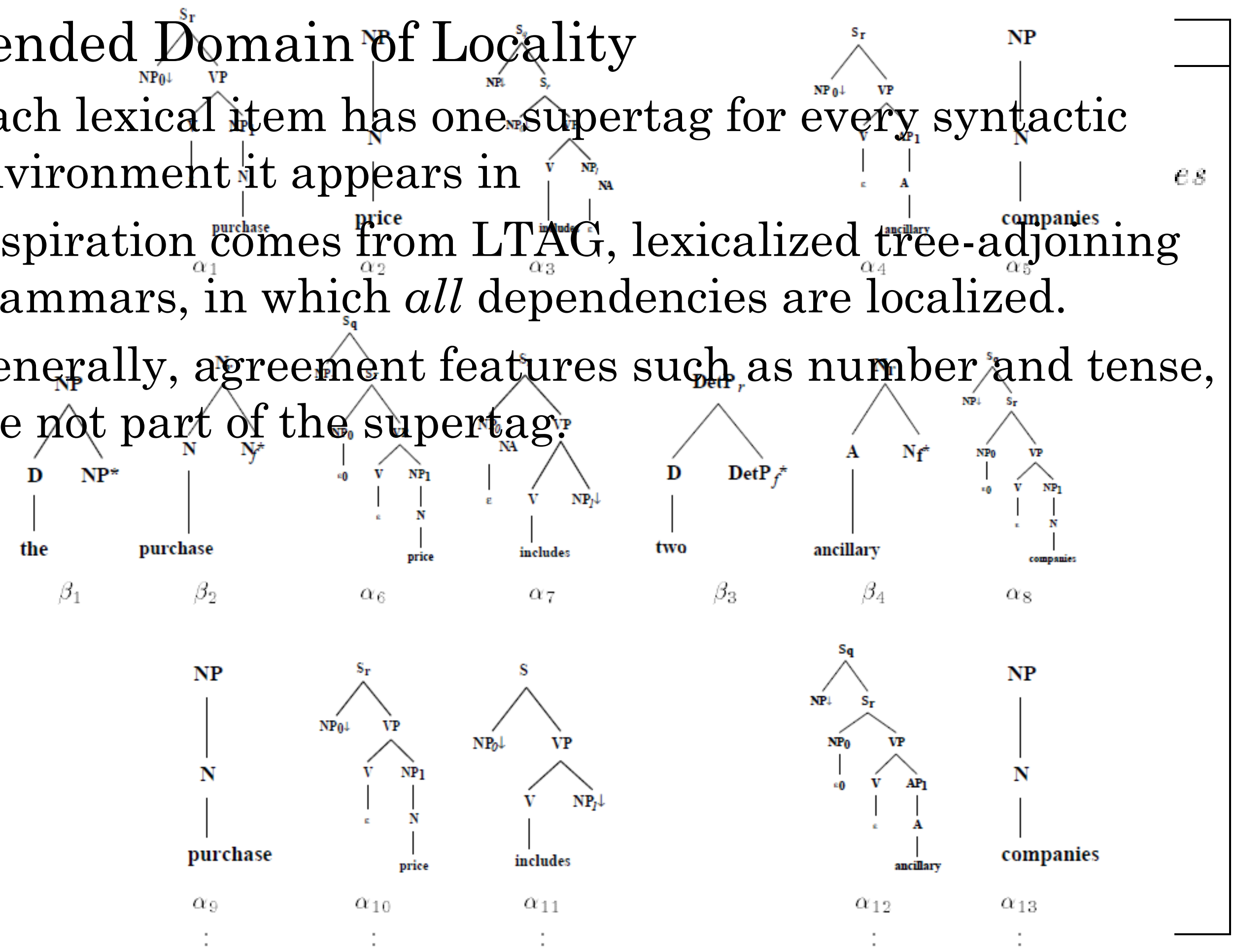
- Straightforward to compile parse from accurate supertagging
  - But impossible if there are any supertag errors
    - We can account for *some* supertag errors
    - Don't always want a full parse anyway

What is supertagging?

- Systematic assignment of supertags
- Supertags are:
  - Statistically selected
    - Robust
    - Tends to work
  - Linguistically motivated
    - This makes sense

WHAT IS SUPERTAGGING?

- Many supertags for each word
  - Extended Domain of Locality
    - Each lexical item has one supertag for every syntactic environment it appears in
    - Inspiration comes from LTAG, lexicalized tree-adjoining grammars, in which *all* dependencies are localized.
    - Generally, agreement features such as number and tense, are not part of the supertag.



How to supertag

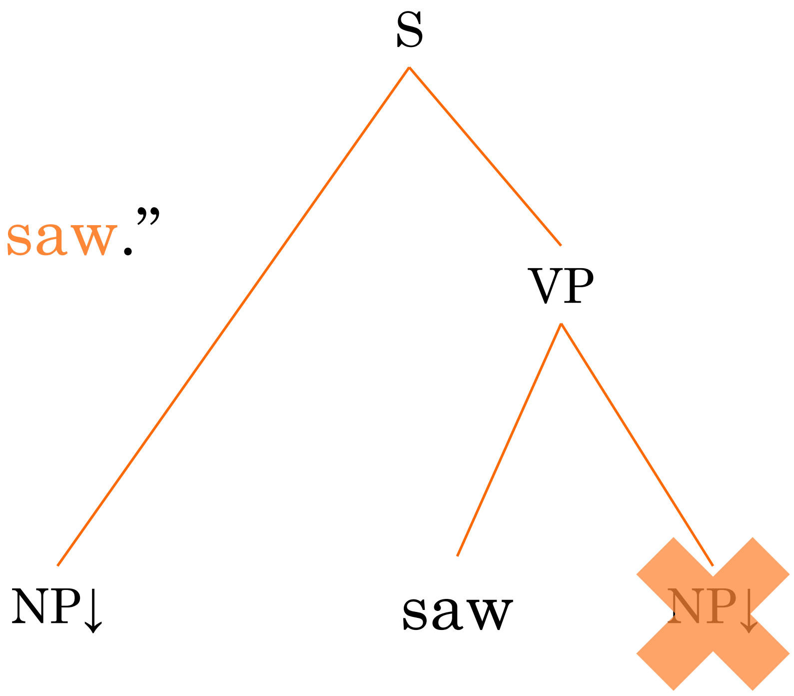
"Alice opened her eyes and saw."

- Supertags:
  - Verb
    - Transitive verb
    - Intransitive verb
    - Infinitive verb
    - …
  - Noun
    - Noun phrase (subject)
    - Nominal predicative
    - Nominal modifier
    - Nominal predicative subject extraction
    - …

HOW TO SUPERTAG

"Alice opened her eyes and saw."

- Supertags:
  - Verb
    - ~~Transitive verb~~
    - Intransitive verb
    - Infinitive verb
    - …
  - Noun
    - Noun phrase (subject)
    - Nominal predicative
    - Nominal modifier
    - Nominal predicative subject extraction
    - …



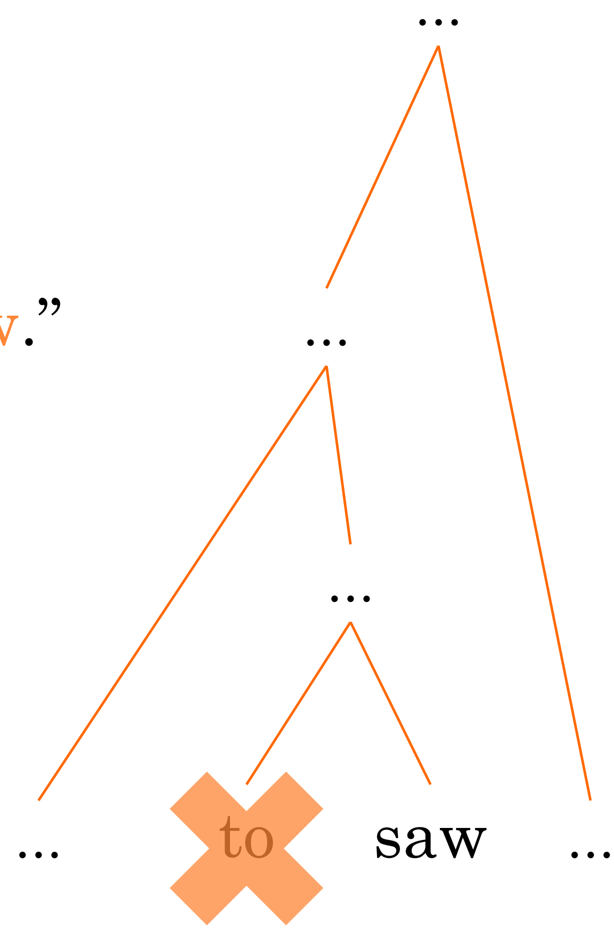
How to supertag

- A supertag can be ruled out for a given word in a given input string…
  - Left and/or right context is too long/short for the input
  - If the supertag contains other terminals not found in the input

How to supertag

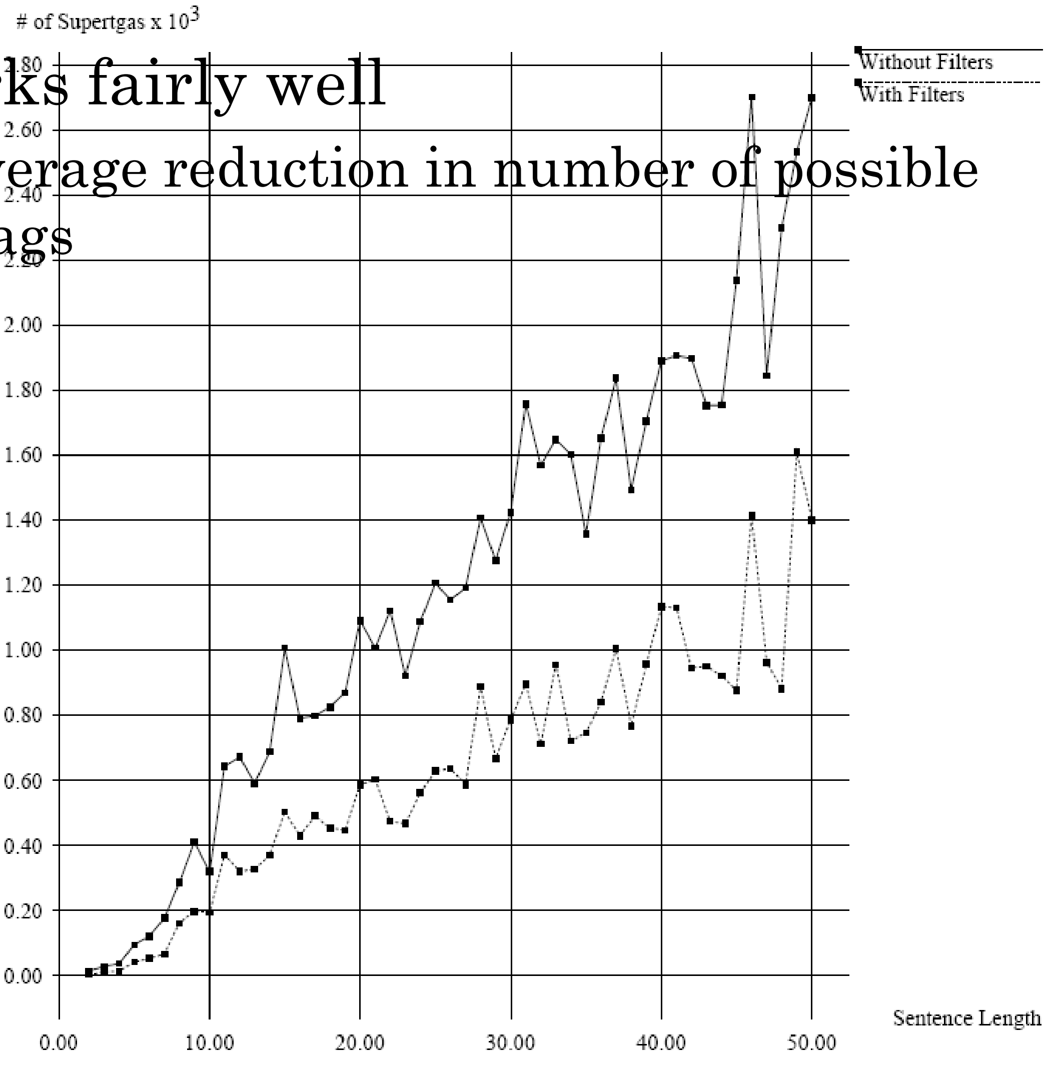
“Alice opened her eyes and saw.”

- Supertags:
  - Verb
    - ~~Transitive verb~~
    - Intransitive verb
    - ~~Infinitive verb~~
    - …
  - Noun
    - Noun phrase (subject)
    - Nominal predicative
    - Nominal modifier
    - Nominal predicative subject extraction
    - …



How to supertag

- This works fairly well
  - 50% average reduction in number of possible supertags



How to supertag

- …but there's more to be done
  - Good: average number of possible supertags per word reduced from 47 to 25
  - Bad: average of 25 possible supertags per word

How to supertag

- Disambiguation by unigrams?
  - Give each word its most frequent supertag after PoS tagging
    - ~75% accurate
      - Better results than one might expect given large number of possible supertags
      - Common words (determiners, etc.) usually correct
        - This helps accuracy
      - Back off to PoS for unknown words
        - Also usually correct

How to supertag

- Disambiguation by n-grams?

$$T = \underset{T}{\operatorname{argmax}} \Pr(T_1, T_2, \dots, T_N) * \Pr(W_1, W_2, \dots, W_N | T_1, T_2, \dots, T_N)$$

  - We assume that subsequent words are independent

$$\Pr(W_1, W_2, \dots, W_N | T_1, T_2, \dots, T_N) \approx \prod_{i=1}^{N} \Pr(W_i | T_i)$$

  - Trigrams plus Good-Turing smoothing
    - Accuracy around 90%
      - Versus 75% from unigrams
    - Contextual information more important than lexical
      - Reversal of trend for PoS tagging

However…

- Correctly supertagged text yields a 30X parsing speedup
  - But even one mistake can cause parsing to fail completely
    - This is rather likely

- Solution: n-best supertags?
  - When n=3, we get up to 96% accuracy…
    - Not bad at all for such a simple method
    - 425 lexical categories (PTB-CFG: ~50)
    - 12 combinatory rules (PTB-CFG: > 500,000)