Computational Semantics

Introduction to Natural Language Processing
Computer Science 585—Fall 2009
University of Massachusetts Amherst

David Smith
with slides from Dan Klein, Stephen Clark & Eva Banik

Overview

- Last time: What is semantics?
  - First order logic and lambda calculus for compositional semantics
- Today: How do we infer semantics?
  - Minimalist approach
    - Semantic role labeling
  - Semantically informed grammar
    - Combinatory categorial grammar (CCG)
    - Tree adjoining grammar (TAG)

Semantic Role Labeling

- Characterize predicates (e.g., verbs, nouns, adjectives) as *relations* with *roles* (slots)

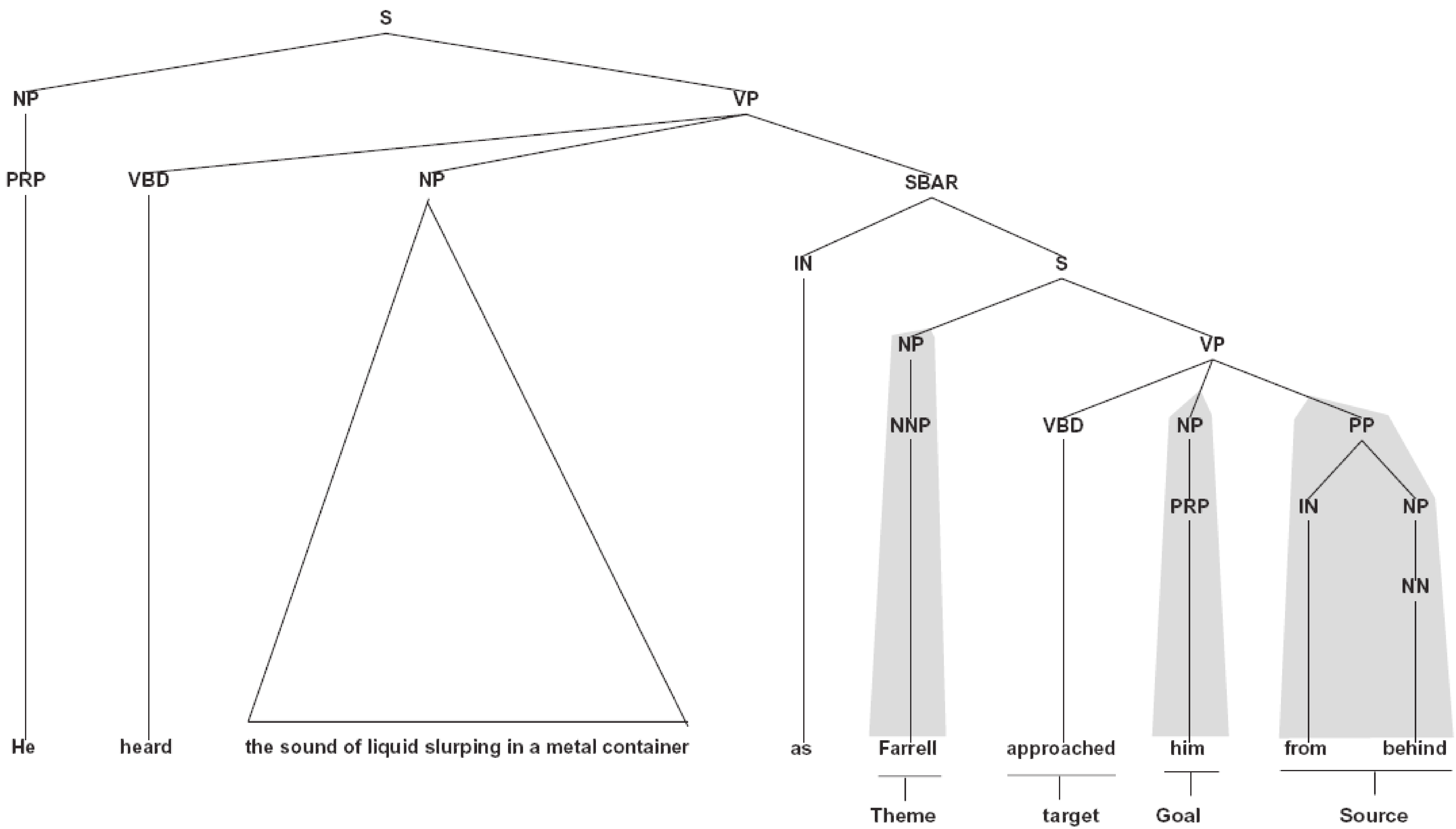
  [$_{Judge}$ She] **blames** [$_{Evaluee}$ the Government] [$_{Reason}$ for failing to do enough to help] .

  Holman would characterize this as **blaming** [$_{Evaluee}$ the poor] .

  The letter quotes Black as saying that [$_{Judge}$ white and Navajo ranchers] misrepresent their livestock losses and **blame** [$_{Reason}$ everything] [$_{Evaluee}$ on coyotes] .

- We want a bit more than which NP is the subject (but not much more):
  - Relations like subject are syntactic, relations like agent or experiencer are semantic (think of passive verbs)
- Typically, SRL is performed in a pipeline on top of constituency or dependency parsing and is much easier than parsing.

SRL Example


S
NP
VP
PRP
VBD
NP
SBAR
IN
S
NP
VP
NNP
VBD
NP
PP
PRP
IN
NP
NN
He
heard
the sound of liquid slurping in a metal container
as
Farrell
approached
him
from
behind
Theme
target
Goal
Source

PropBank Example

fall.01		sense: move downward	
	roles:	Arg1:	thing falling
		Arg2:	extent, distance fallen
		Arg3:	start point
		Arg4:	end point

Sales fell to $251.2 million from $278.7 million.

arg1:	Sales
rel:	fell
arg4:	to $251.2 million
arg3:	from $278.7 million

PropBank Example

rotate.02 sense: shift from one thing to another

roles:	Arg0:	causer of shift
	Arg1:	thing being changed
	Arg2:	old thing
	Arg3:	new thing

Many of Wednesday's winners were losers yesterday as investors quickly took profits and rotated their buying to other issues, traders said. (wsj_1723)

arg0:	investors
rel:	rotated
arg1:	their buying
arg3:	to other issues

PropBank Example

aim.01 sense: intend, plan
roles: Arg0: aimer, planner
Arg1: plan, intent

The Central Council of Church Bell Ringers aims *trace* to improve relations with vicars. (wsj_0089)

arg0: The Central Council of Church Bell Ringers
rel: aims
arg1: *trace* to improve relations with vicars

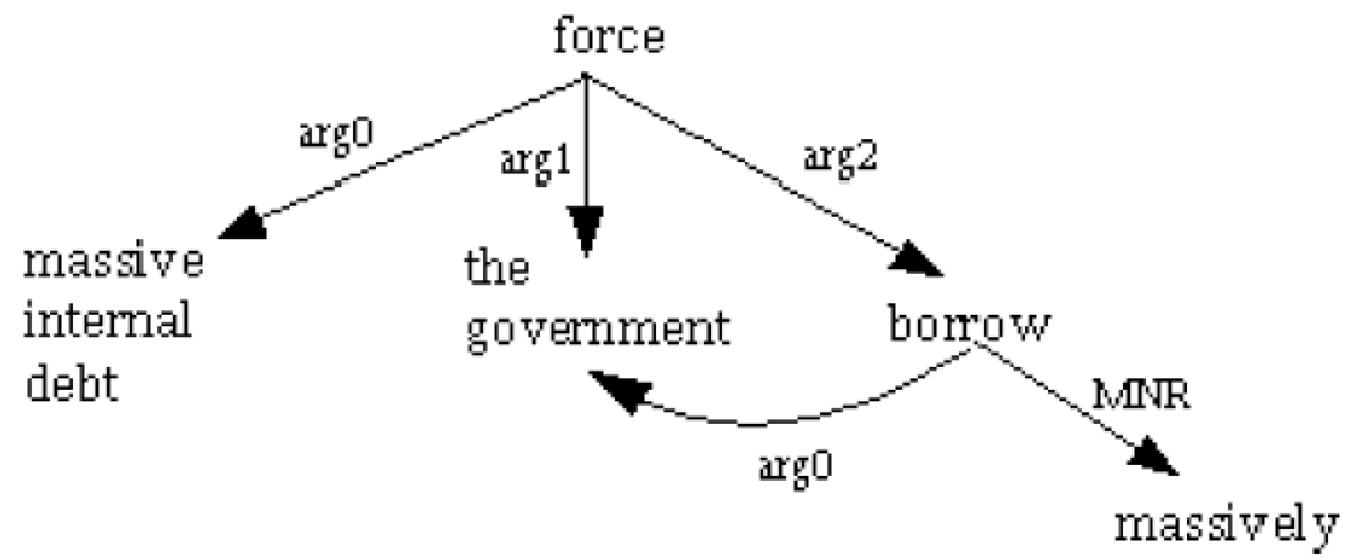
aim.02 sense: point (weapon) at
roles: Arg0: aimer
Arg1: weapon, etc.
Arg2: target

Banks have been aiming packages at the elderly.

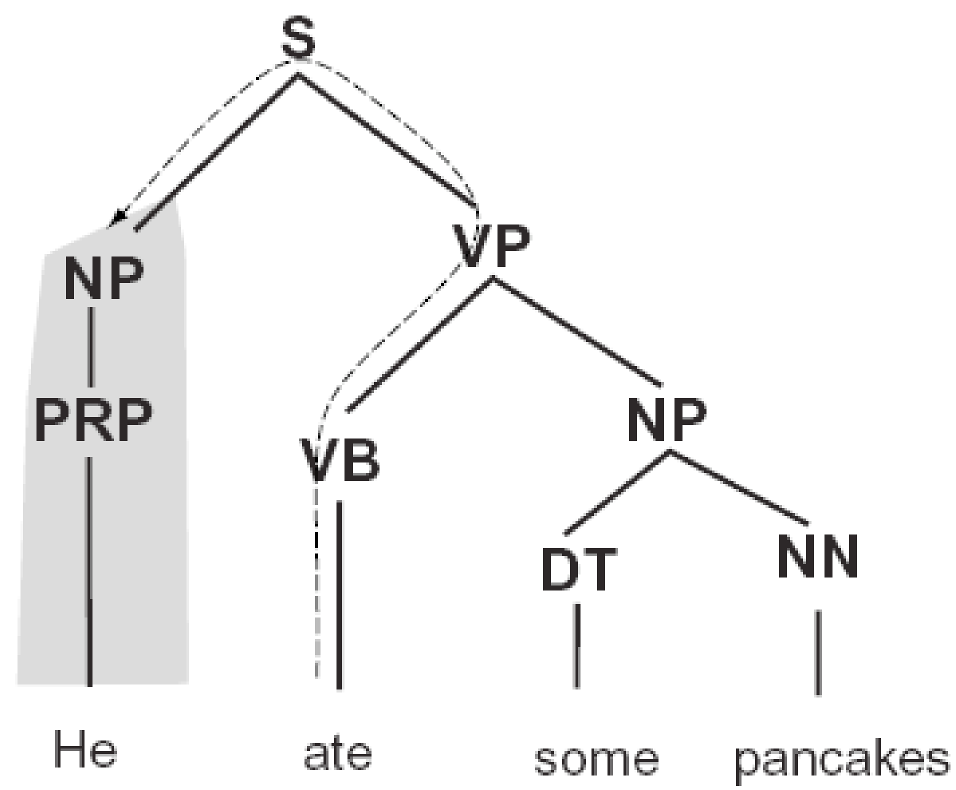
arg0: Banks
rel: aiming
arg1: packages
arg2: at the elderly

Shared Arguments

```
(NP-SBJ (JJ massive) (JJ internal) (NN debt) )
  (VP (VBZ has)
   (VP (VBN forced)
    (S
     (NP-SBJ-1 (DT the) (NN government) )
     (VP
      (VP (TO to)
       (VP (VB borrow)
        (ADVP-MNR (RB massively) )...
```



Path Features



Path	*Description*
VB↑VP↓PP	PP argument/adjunct
VB↑VP↑S↓NP	subject
VB↑VP↓NP	object
VB↑VP↑VP↑S↓NP	subject (embedded VP)
VB↑VP↓ADVP	adverbial adjunct
NN↑NP↑NP↓PP	prepositional complement of noun

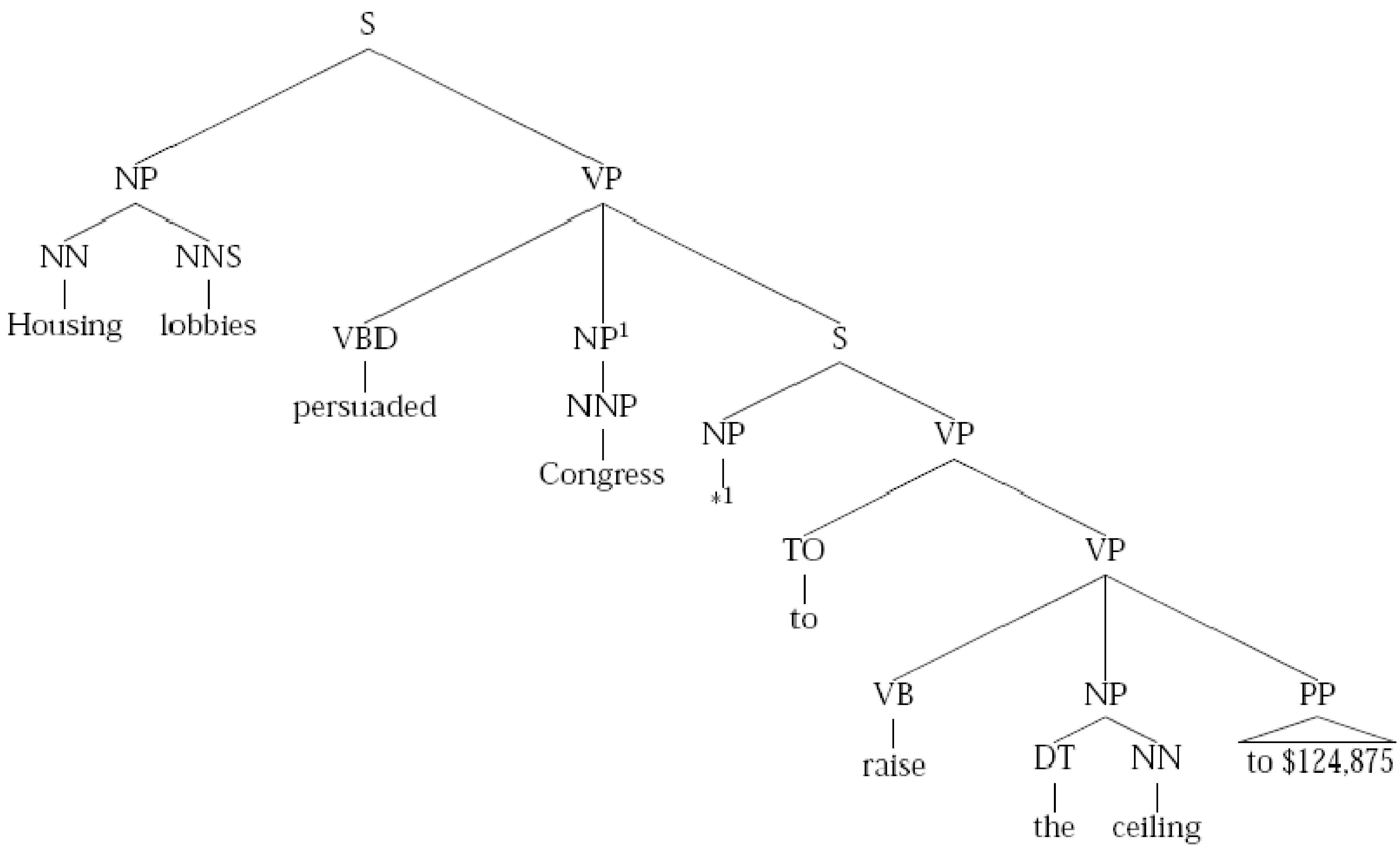
SRL Accuracy

- Features
  - Path from target to role-filler
  - Filler's syntactic type, headword, case
  - Target's identity
  - Sentence voice, etc.
  - Lots of other second-order features
- Gold vs. parsed source trees
  - SRL is fairly easy on gold trees
  - Harder on automatic parses
  - Joint inference of syntax and semantics not a helpful as expected

CORE		ARGM	
F1	Acc.	F1	Acc.
92.2	80.7	89.9	71.8

CORE		ARGM	
F1	Acc.	F1	Acc.
84.1	66.5	81.4	55.6

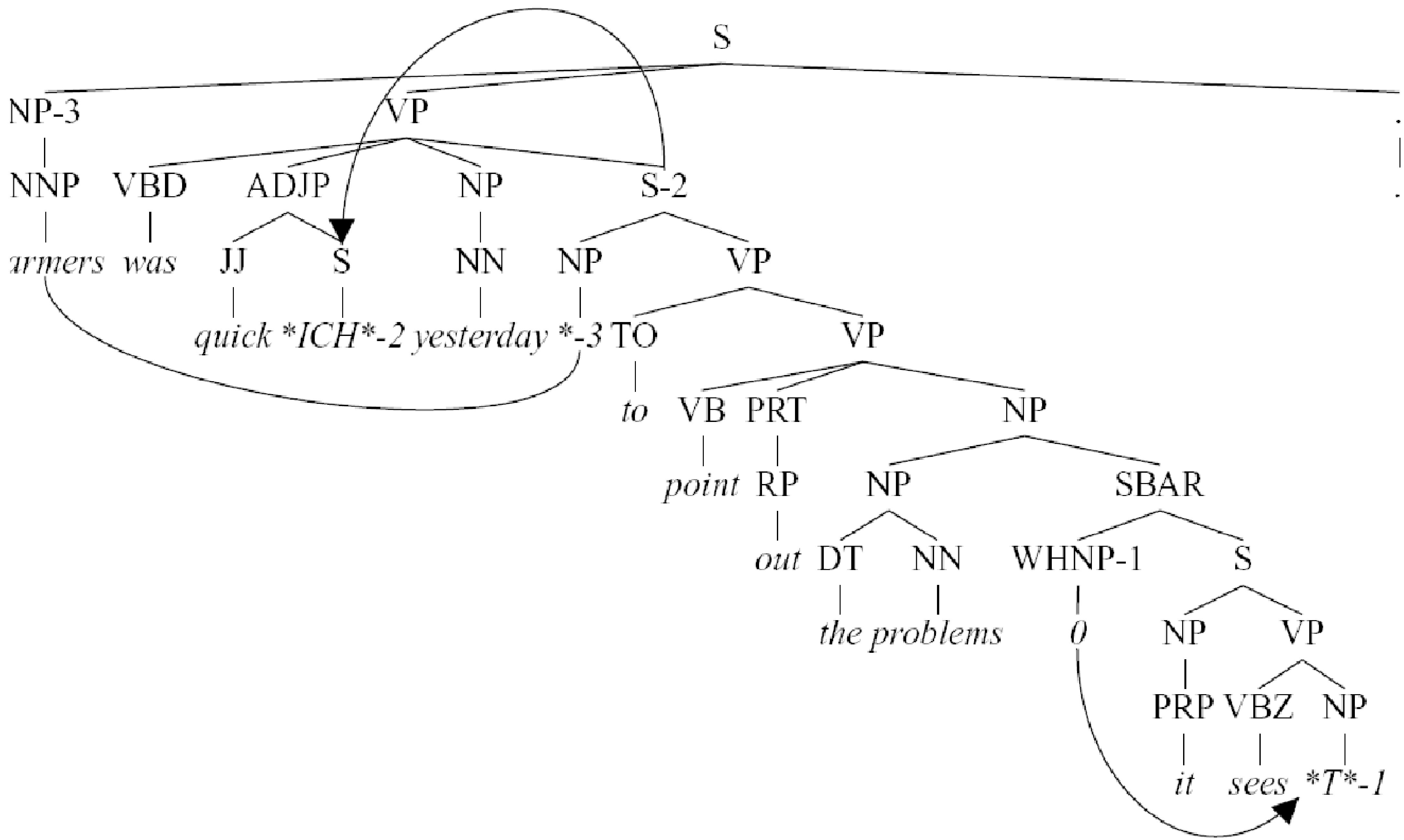
Interaction with Empty Elements



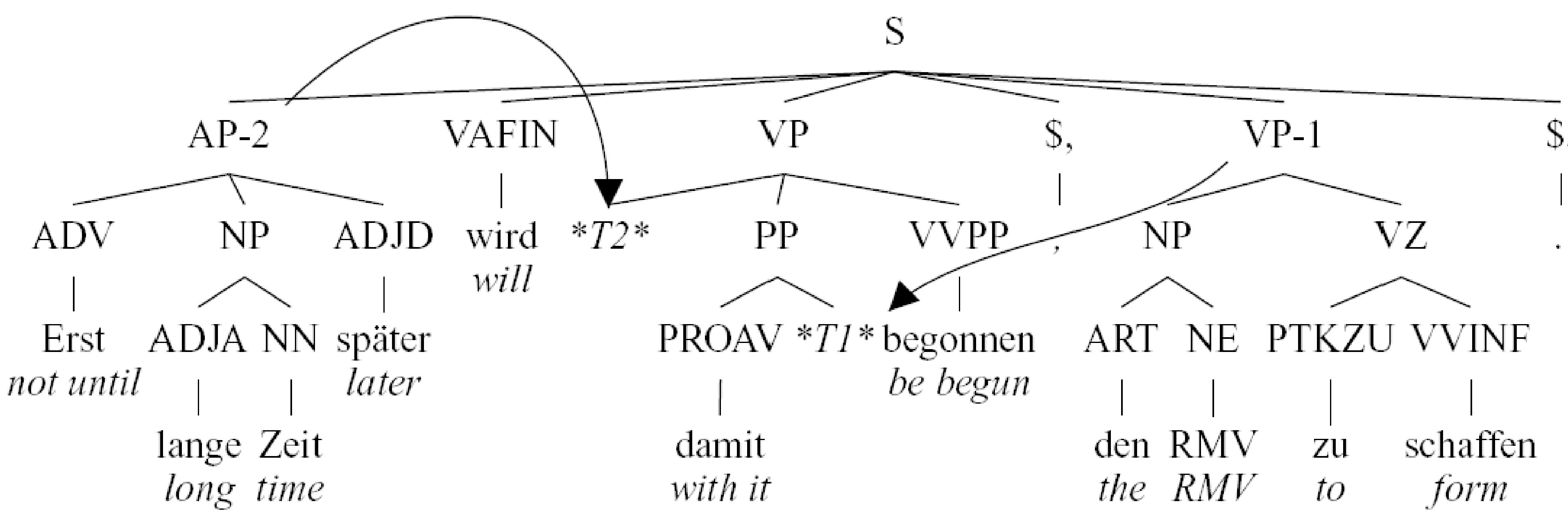
Empty Elements

- In Penn Treebank, 3 kinds of empty elem.
  - Null items
  - Movement traces (WH, topicalization, relative clause and heavy NP extraposition)
  - Control (raising, passives, control, shared arguments)
- Semantic interpretation needs to reconstruct these and resolve indices

English Example



German Example



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; prodody 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

$$
\begin{array}{ccc}
\mathit{interleukin} - \mathit{10} & \mathit{inhibits} & \mathit{production} \\
\hline
NP & (S\backslash NP)/NP & NP \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP \quad >} \\
\hline
\multicolumn{3}{c}{S \quad <}
\end{array}
$$

$>$ forward application

$<$ backward application

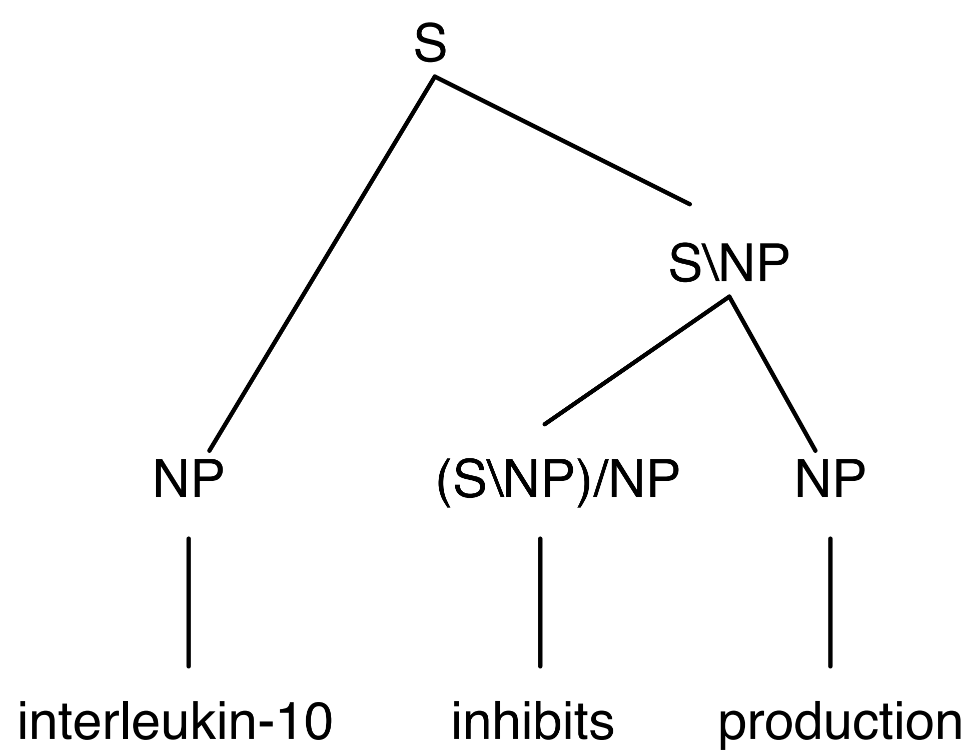
Function Application Schemata

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

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

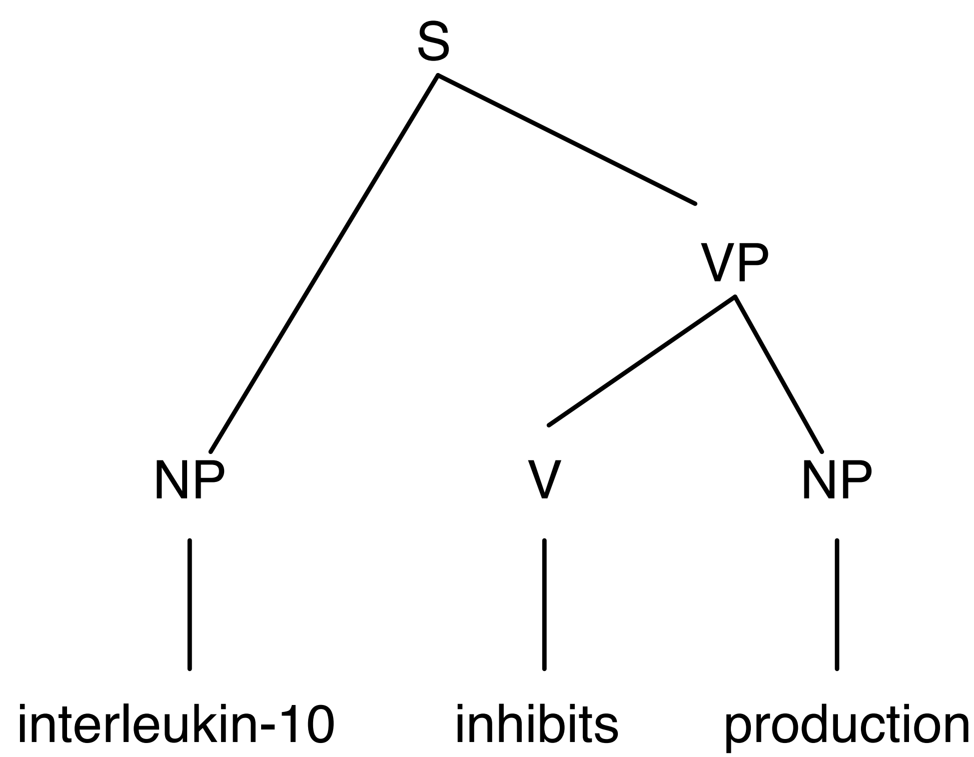
Classical Categorial Grammar

- ‘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

$$\dfrac{The}{NP/N} \quad \dfrac{company}{N} \quad \dfrac{which}{(NP\backslash NP)/(S/NP)} \quad \cfrac{\dfrac{Microsoft}{NP}}{S/(S\backslash NP)}{>}\mathbf{T} \quad \dfrac{bought}{(S\backslash NP)/NP}$$

$>\mathbf{T}$ type-raising

Extraction out of a Relative Clause

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

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

Extraction out of a Relative Clause

$$
\begin{array}{ccccc}
\textit{The} & \textit{company} & \textit{which} & \textit{Microsoft} & \textit{bought} \\
\hline
NP/N & N & (NP\backslash NP)/(S/NP) & NP & (S\backslash NP)/NP \\
 & & & \overline{S/(S\backslash NP)}^{>\mathbf{T}} & \\
 & & & \multicolumn{2}{c}{\overline{S/NP}^{>\mathbf{B}}} \\
 & & \multicolumn{3}{c}{\overline{NP\backslash NP}^{>}}
\end{array}
$$

Extraction out of a Relative Clause

$$
\begin{array}{ccccc}
\mathit{The} & \mathit{company} & \mathit{which} & \mathit{Microsoft} & \mathit{bought} \\
\hline
NP/N & N & (NP\backslash NP)/(S/NP) & NP & (S\backslash NP)/NP \\
\cline{1-2} \cline{4-4}
\multicolumn{2}{c}{{\color{blue}NP} \quad >} & & S/(S\backslash NP) \quad >\mathbf{T} & \\
\cline{4-5}
 & & & \multicolumn{2}{c}{S/NP \quad >\mathbf{B}} \\
\cline{3-5}
 & & \multicolumn{3}{c}{NP\backslash {\color{blue}NP} \quad >} \\
\cline{1-5}
\multicolumn{5}{c}{NP \quad <}
\end{array}
$$

Forward Composition and Type-Raising

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

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

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

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

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

“Non-constituents” in CCG – Right Node Raising

$$\dfrac{\dfrac{Google}{NP}}{S/(S\backslash NP)}{>}\mathbf{T} \quad \dfrac{sells}{(S\backslash NP)/NP} \quad \dfrac{but}{conj} \quad \dfrac{\dfrac{Microsoft}{NP}}{S/(S\backslash NP)}{>}\mathbf{T} \quad \dfrac{buys}{(S\backslash NP)/NP} \quad \dfrac{shares}{NP}$$

$> \mathbf{T}$ type-raising

"Non-constituents" in CCG – Right Node Raising

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

$>$ **T** type-raising

$>$ **B** forward composition

"Non-constituents" in CCG – Right Node Raising

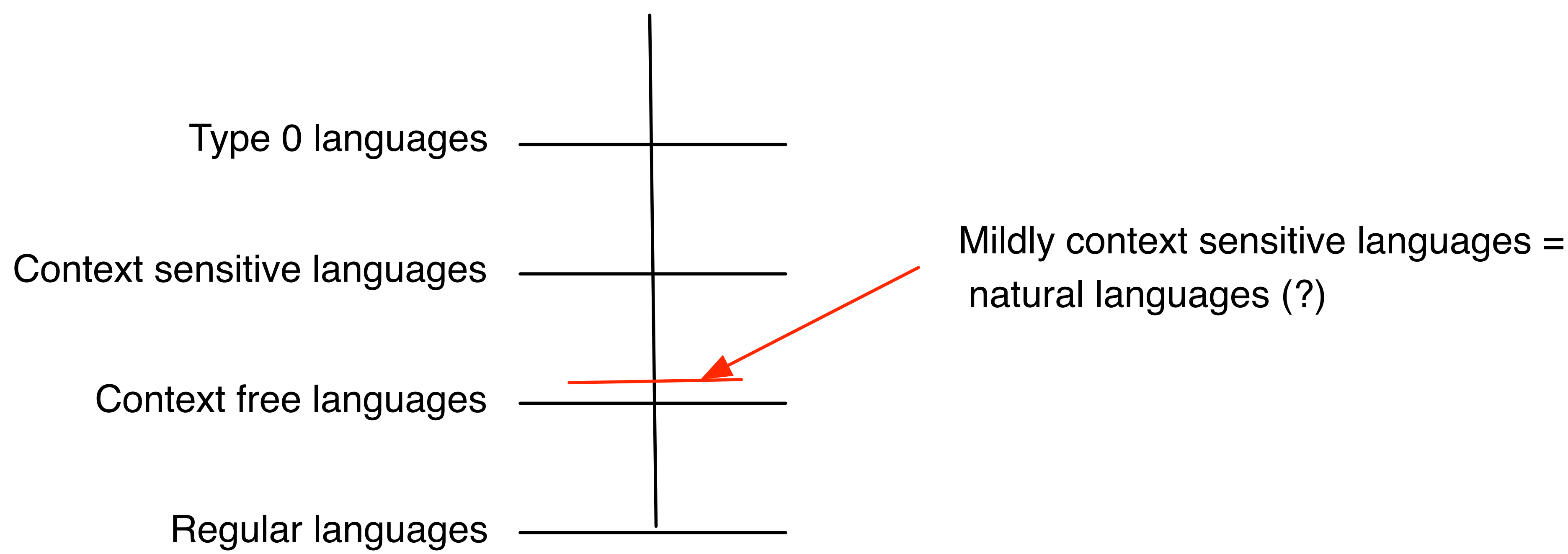
$$
\begin{array}{cccccc}
\textit{Google} & \textit{sells} & \textit{but} & \textit{Microsoft} & \textit{buys} & \textit{shares} \\
\hline
NP & (S\backslash NP)/NP & conj & NP & (S\backslash NP)/NP & NP \\
\hline
S/(S\backslash NP)\ {>}\mathbf{T} & & & S/(S\backslash NP)\ {>}\mathbf{T} & & \\
\hline
\multicolumn{2}{c}{S/NP\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

“Non-constituents” in CCG – Right Node Raising

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

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

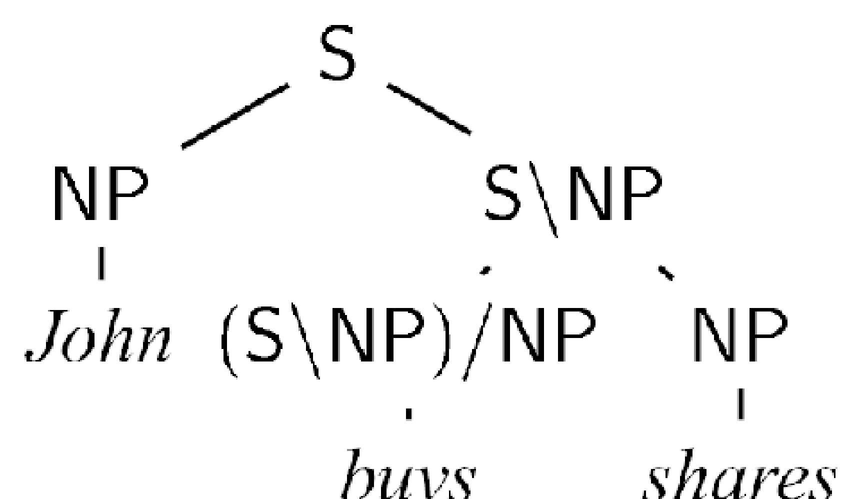
$John \vdash \mathsf{NP} : john'$

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

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

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

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



CCG Semantics

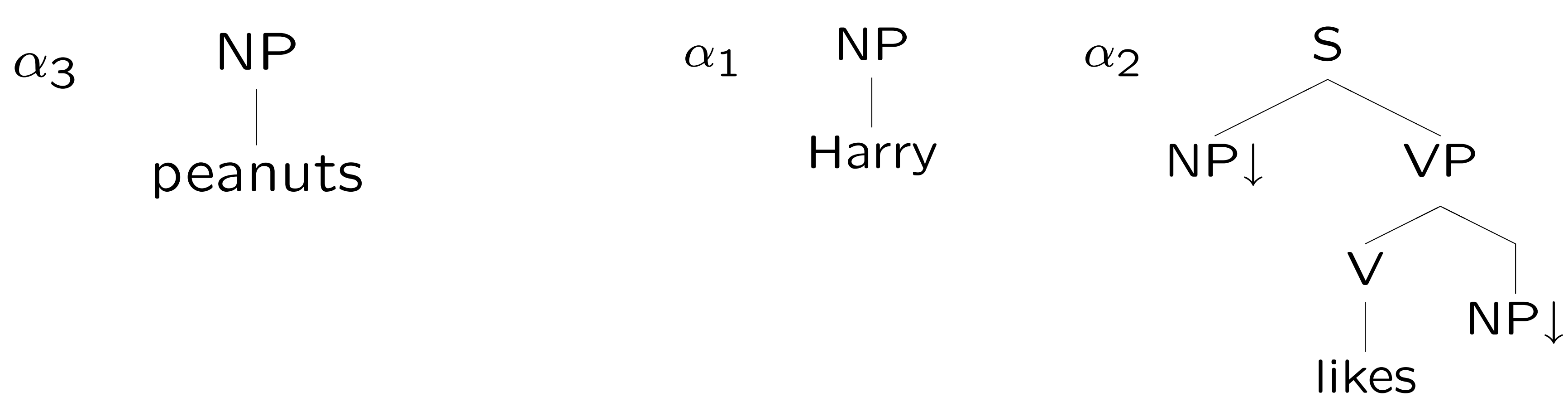
Left arg.	Right arg.	Operation	Result
X/Y : f	Y : a	Forward application	X : f(a)
Y : a	X\Y : f	Backward application	X : f(a)
X/Y : f	Y/Z : g	Forward composition	X/Z : λx.f(g(x))
X : a		Type raising	T/(T\X) : λf.f(a)

etc.

Tree Adjoining Grammar

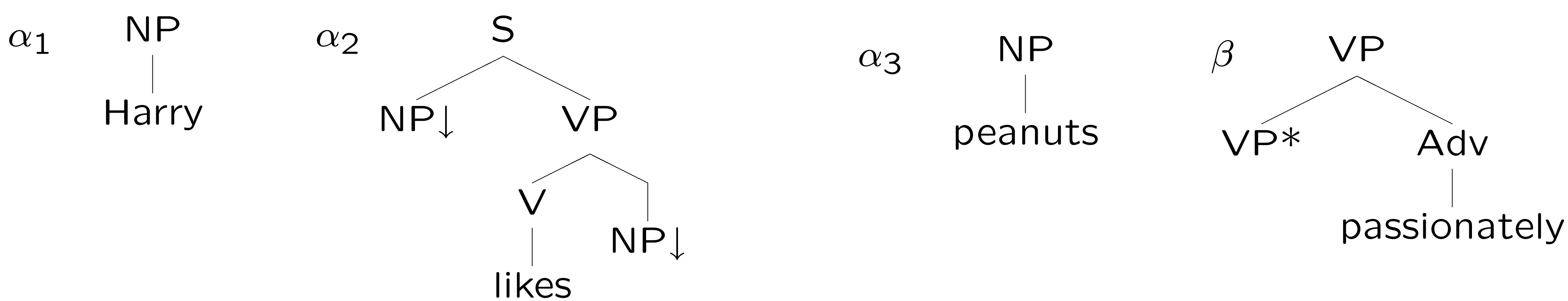
TAG Building Blocks

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

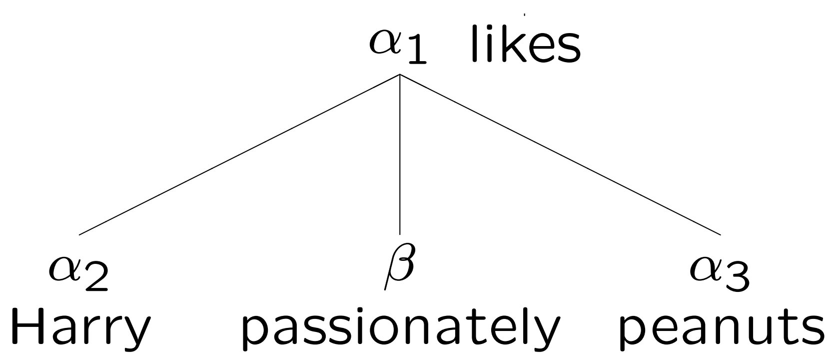


TAG Building Blocks

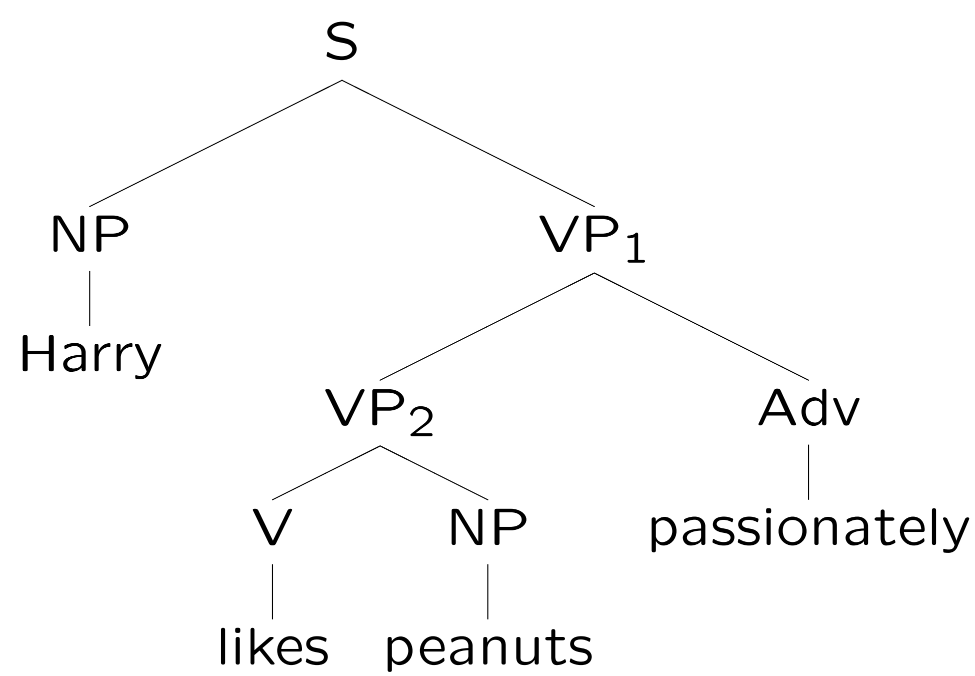
- Auxiliary trees for *adjunction*
- Adds extra power beyond CFG



Derivation Tree



Derived Tree



Semantics

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

4

Semantic representation - derived or derivation tree?

Derived tree

- not monotonic (e.g. immediate domination)
- contains nodes that are not needed for semantics

Derivation tree in TAG shows

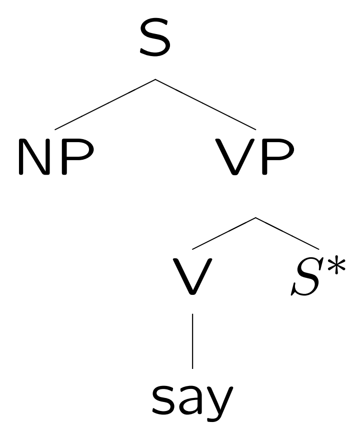
- what elementary and auxiliary trees were used
- how the trees were combined
- where the trees were adjoined / substituted

⇒ Derivation tree provides a natural representation for compositional semantics

Elementary Semantic Representations

- description of meaning (conjunction of formulas)
- list of argument variables

β_{say}



$say(e_1, x, e_2)$
arg: $< x, 00 >, < e_2, 011 >$

Composition of Semantic Representations

- sensitive to way of composition indicated in the derivation tree
- sensitive to order of traversal

Substitution: a new argument is inserted in $\sigma(\alpha)$

- unify the variable corresponding to the argument node (e.g. x in $thought(e, x)$) with the variable in the substituted tree (e.g. NP: $Peter(x_5)$)
- semantic representations are merged

Adjoining: $\sigma(\beta)$ applied to $\sigma(\alpha)$

- predicate: semantic representation of adjoined auxiliary tree
- argument: a variable in the 'host' tree

Harry likes peanuts passionately.

$Harry(x)$
arg: -

$likes(e, x, y)$
arg: $< x, 00 >, < y, 011 >$

$peanuts(y)$
arg: -

$passionately(e)$
arg: e

Result:

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

Extensions and Multi-Component LTAG

To what extent can we obtain a compositional semantics by using derivation trees?

Problem: Representation of Scope

Every boy saw a girl.

(suppose there are 5 boys in the world, how many girls have to exist for the sentence to be true?)

Quantifiers have two parts:

- predicate-argument structure
- scope information

The two parts don't necessarily stay together in the final semantic representation.

Multi-Component Lexicalized Tree Adjoining Grammar

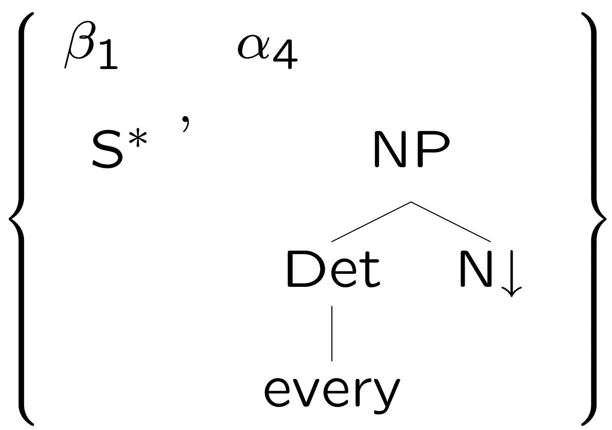
- Building blocks are sets of trees (roughly corresponding to split-up LTAG elementary trees)

- Locality constraint: a multi-component elementary tree has to be combined with only one elementary tree (tree locality; Tree local MC-TAG is as powerful as LTAG)

- We use at most two components in each set

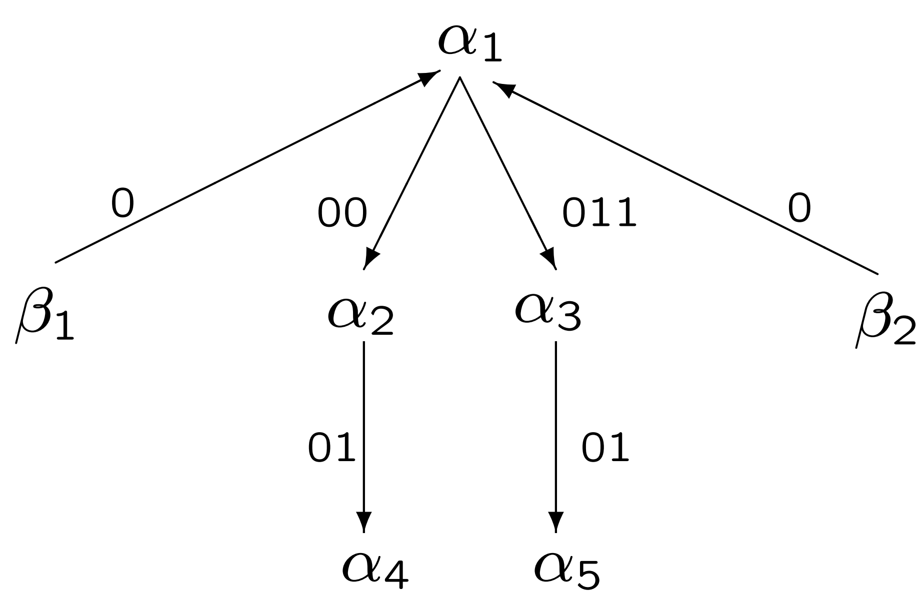
- Constraint on multiple adjunction

Representation of Quantifiers in MC-TAG



Derivation Tree with Two Quantifiers - underspecified scope

Some student loves every course.



CCG & TAG

- Lexicon is encoded as combinators or trees
- *Extended domain of locality*: information is localized in the lexicon and "spread out" during derivation
- Greater than context-free power; polynomial-time parsing; $O(n^5)$ and up
- Spurious ambiguity: multiple derivations for a single derived tree