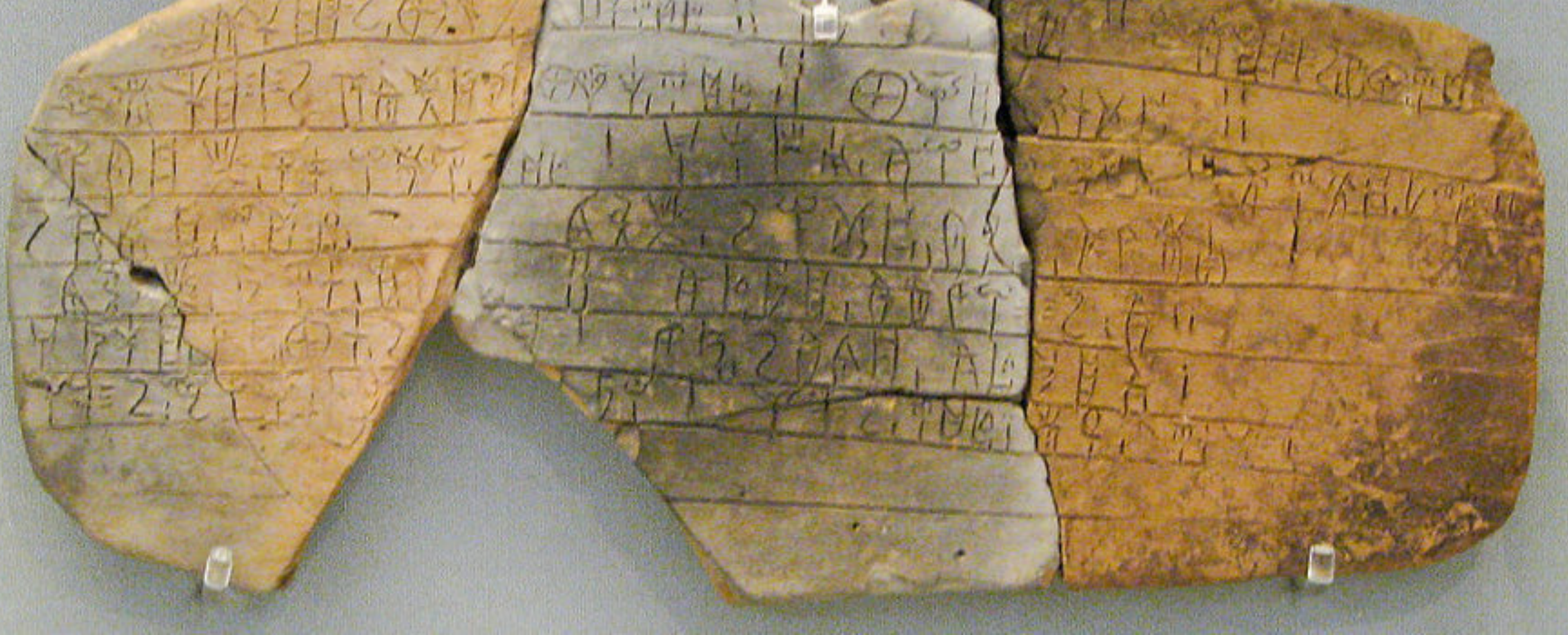




Natural Language Processing

Info 159/259

Lecture 19: Semantic parsing (Oct. 31, 2017)

David Bamman, UC Berkeley

Announcements

- 259 final project presentations: 3:30-5pm Tuesday, Dec. 5 (RRR week), 202 South Hall

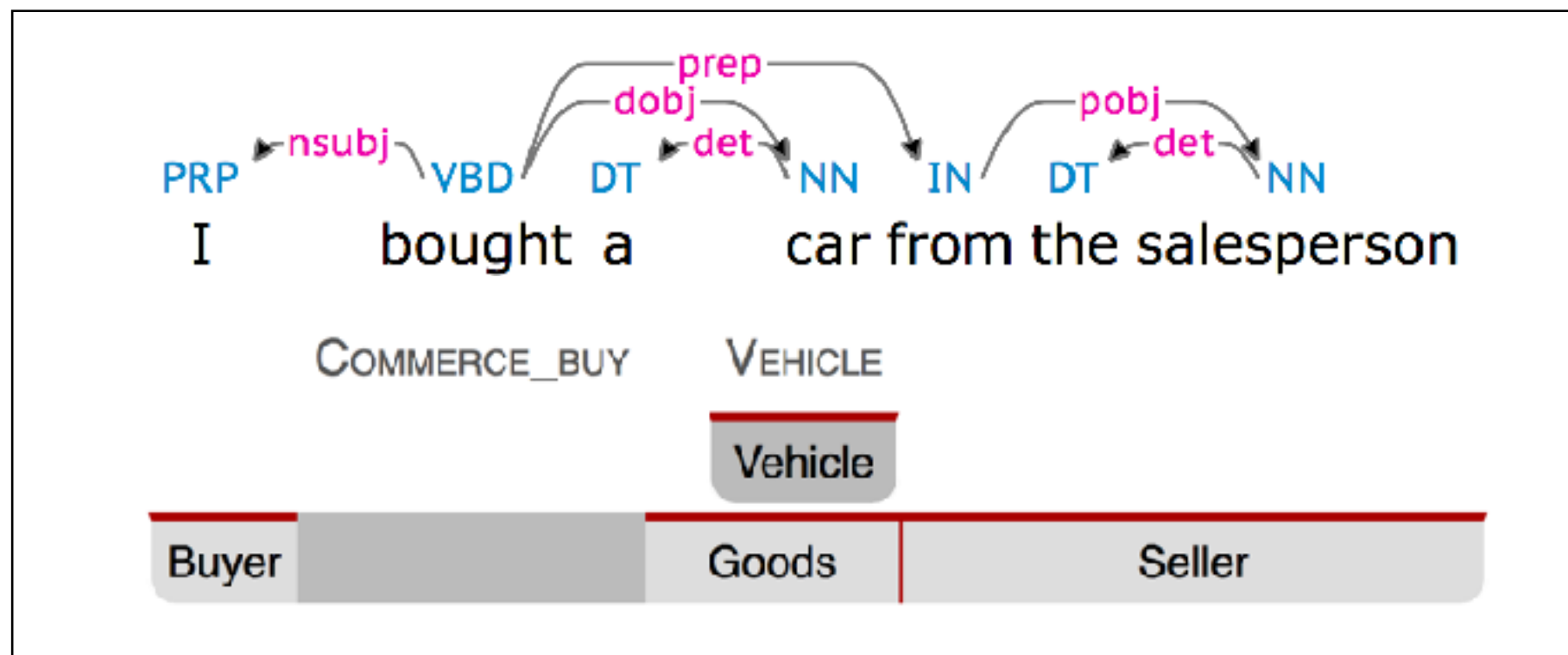
Semantics

- The meaning of a sentence is the set of possible worlds it picks out
- Collections of possible worlds can be compactly represented with logical forms
- Pieces of logical forms correspond to pieces of language
- Types in logic correspond to grammatical categories in language

From last time (Andreas)

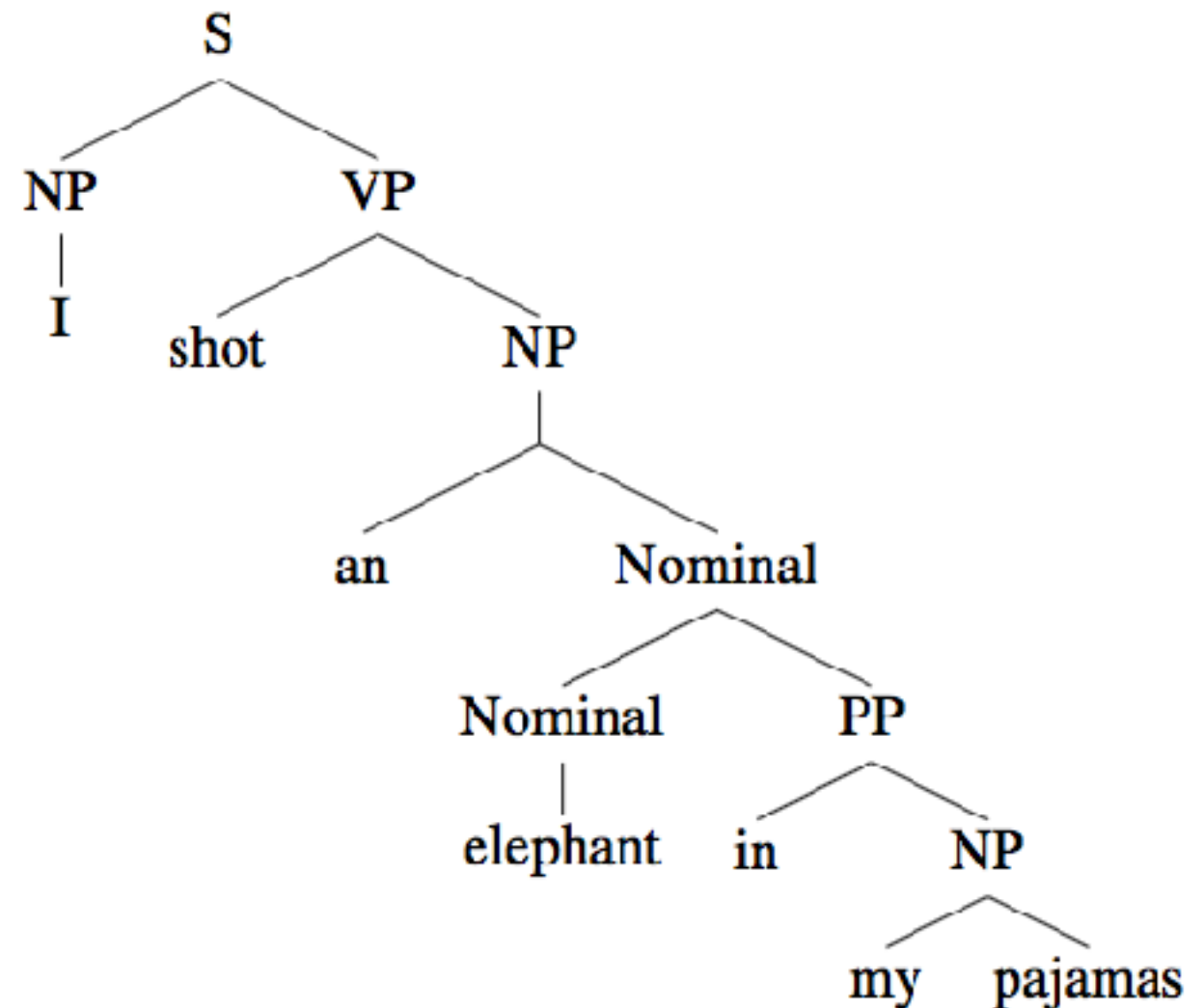
Why is syntax important?

- Foundation for **semantic analysis** (on many levels of representation: semantic roles, compositional semantics, frame semantics)



Why is syntax insufficient?

- Syntax encodes the structure of language but doesn't directly address **meaning**.
- Even if we have a reference model for each word in a sentence, syntax doesn't tell us how those referents change as a function of their compositionally.



Representation of meaning

- **Constants** name individual entities in the world
- **Relations** are sets of entities
- **Variables** refer to entities that have not yet been specified
- **Quantifiers** bind variables.
 - $\exists$ (existential quantifier)
 - $\forall$ (universal quantifier)

Representation of meaning

Pat likes Sal

- Constants: Pat, Sal
- Relations: likes(x,y)
- The denotation $\llbracket \text{likes} \rrbracket$ = the ordered set of entities for whom the relation is true
- $\text{likes}(\text{Pat}, \text{Sal}) = \text{true}$
- $\llbracket \text{likes} \rrbracket = \{(\text{Pat}, \text{Sal}), (\dots, \dots)\}$

Representation of meaning

- Quantifiers bind variables.
 - $\exists$ (existential quantifier)
 - $\forall$ (universal quantifier)
- Order matters!
 - $\forall x \exists y \text{ speaks}(x,y)$
 - $\exists y \forall x \text{ speaks}(x,y)$

Representation of meaning

- Relations: likes(x,y) is scoped over two variables
- We can represent the partial representation of meaning with lambda expressions:

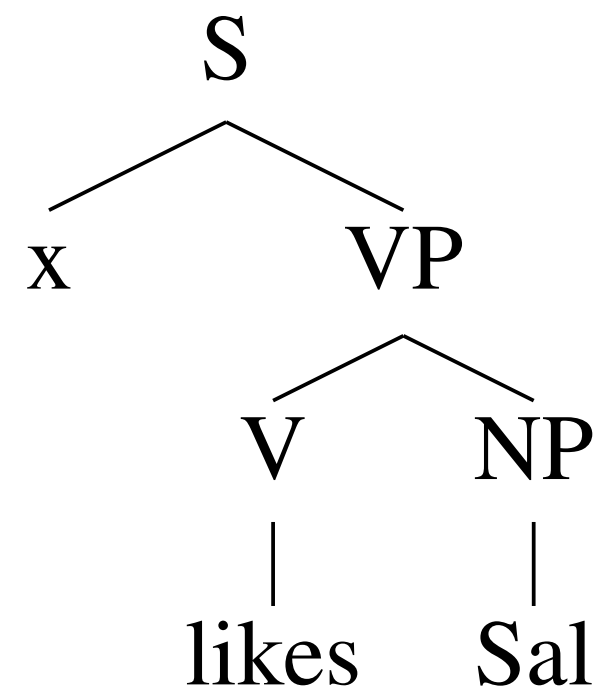
$\lambda x.\text{likes}(x,\text{Sal})$

Expect one other argument to complete the meaning of this relation

Representation of meaning

$\lambda x.\text{likes}(x, \text{Sal})$

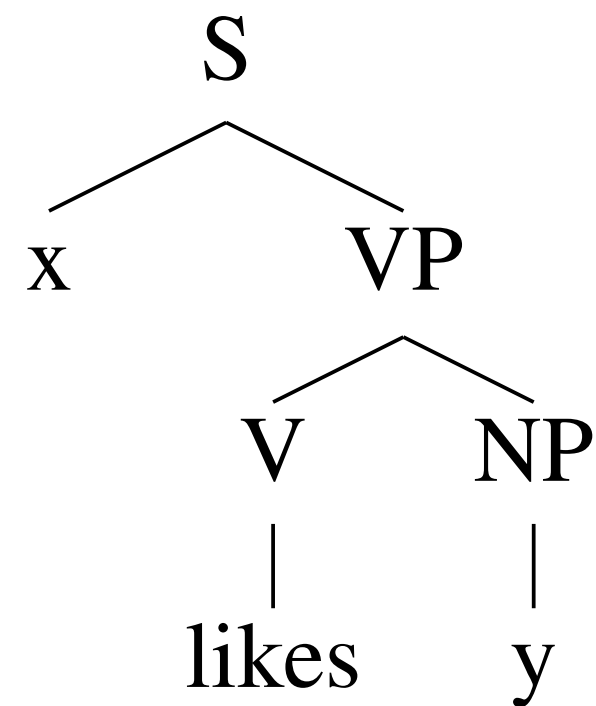
Lambda expressions
let us tie semantics
explicitly to phrases
(subtrees in syntax)



Representation of meaning

$\lambda y.\lambda x.\text{likes}(x,y)$

Lambda expressions
let us tie semantics
explicitly to phrases
(subtrees in syntax)

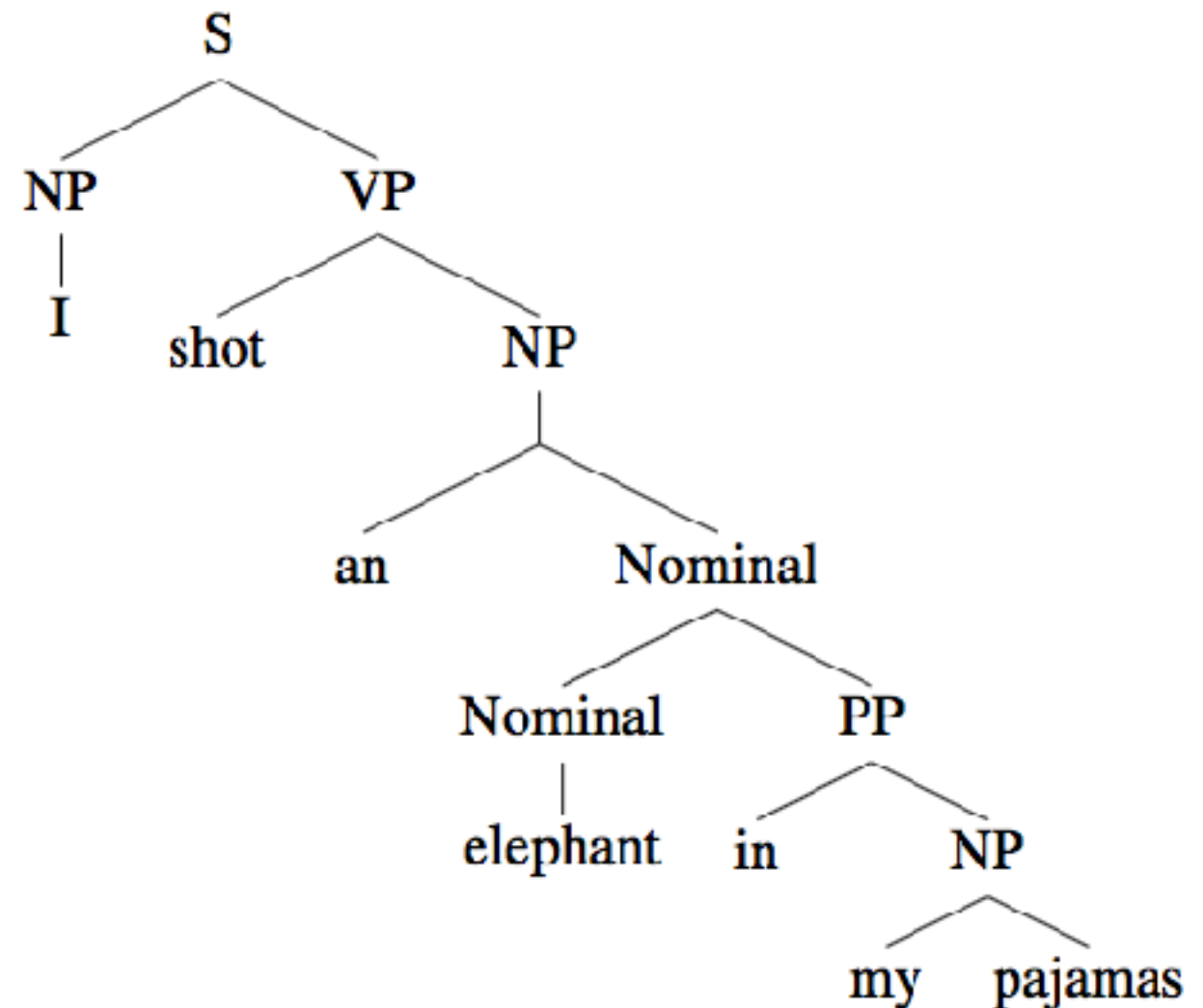


Compositional semantics is driven by syntax

Principle of compositionality

Syntax

- We could represent the relationship between syntax and semantics in a CFG.
- But what we want is fine-grained control over the mapping between **words** and semantic primitives.



CCG

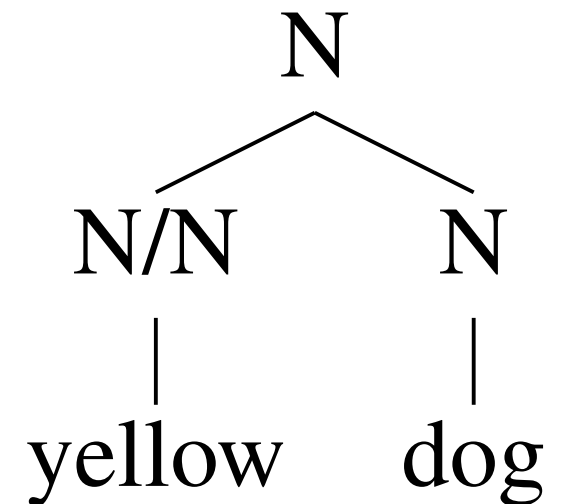
- Infinitely large set of structured categories (*types*).
- Primitives:
 - S, NP, PP, N
 - Complex types:
 - S/NP (S, except missing NP to right)
 - S\NP (S, except missing NP to left)

CCG

- CFG has a large set of productions (e.g., $S \rightarrow NP VP$)
- CCG has a very small set of **combinators** that us how to put the types together.

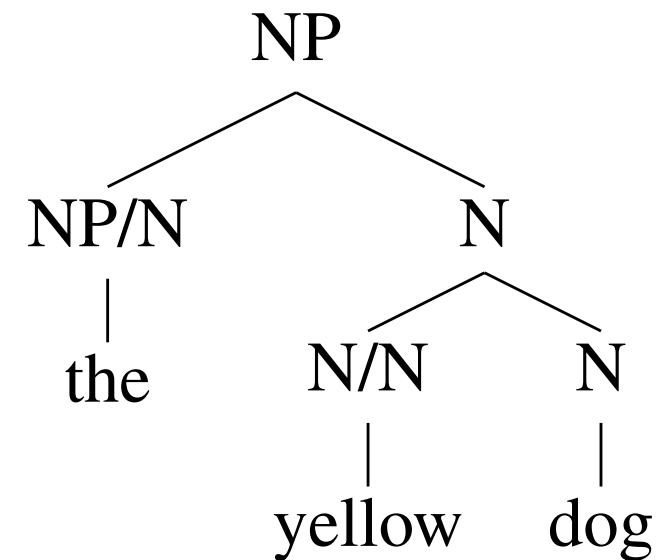
CCG Combinators

- Forward application combinator ($X/Y \rightarrow X$)
- $N/N \rightarrow N$



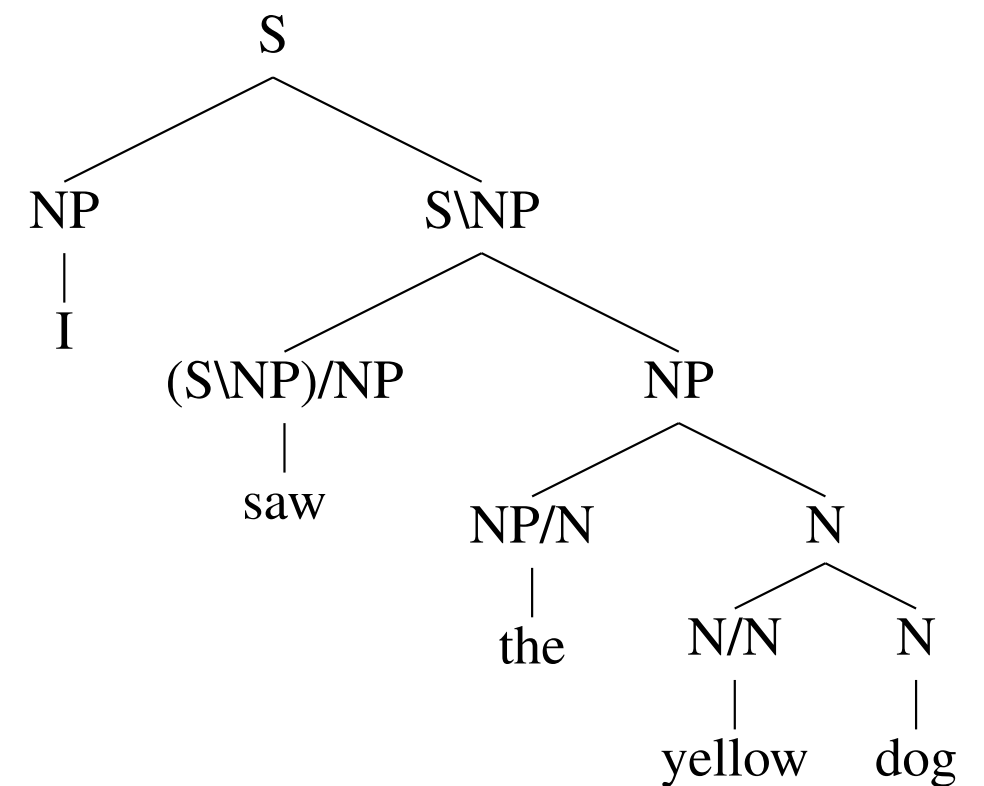
CCG Combinators

- Forward application combinator ($X/Y \rightarrow X$)
 - $N/N \rightarrow N$
 - $NP/N \rightarrow NP$



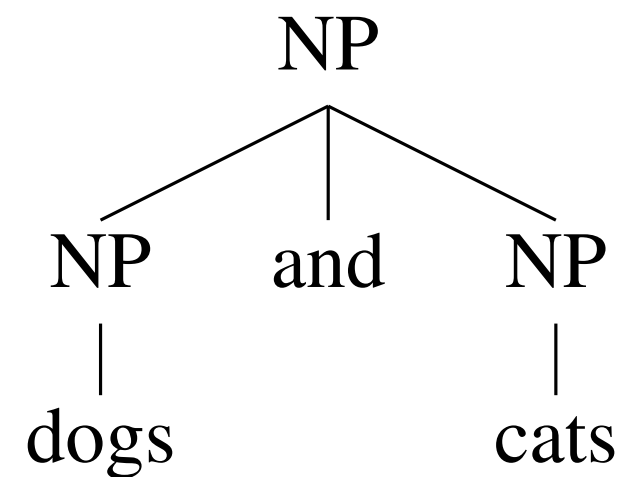
CCG Combinators

- Backward application combinator ($X \backslash Y \rightarrow X$)
- $S \backslash NP \rightarrow S$



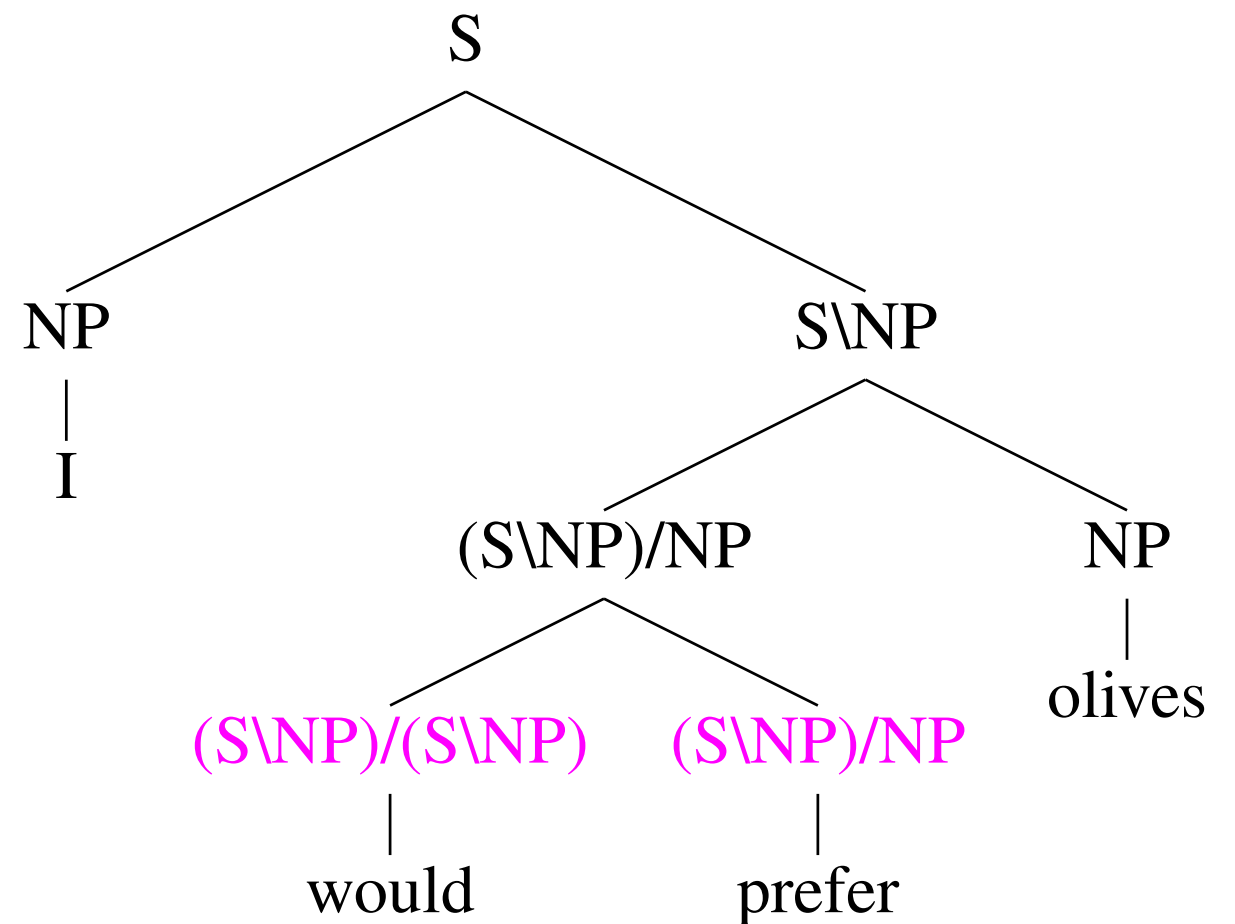
CCG Combinators

- Conjunction combinator ($X \text{ and } X \rightarrow X$)
- $\text{NP and NP} \rightarrow \text{NP}$



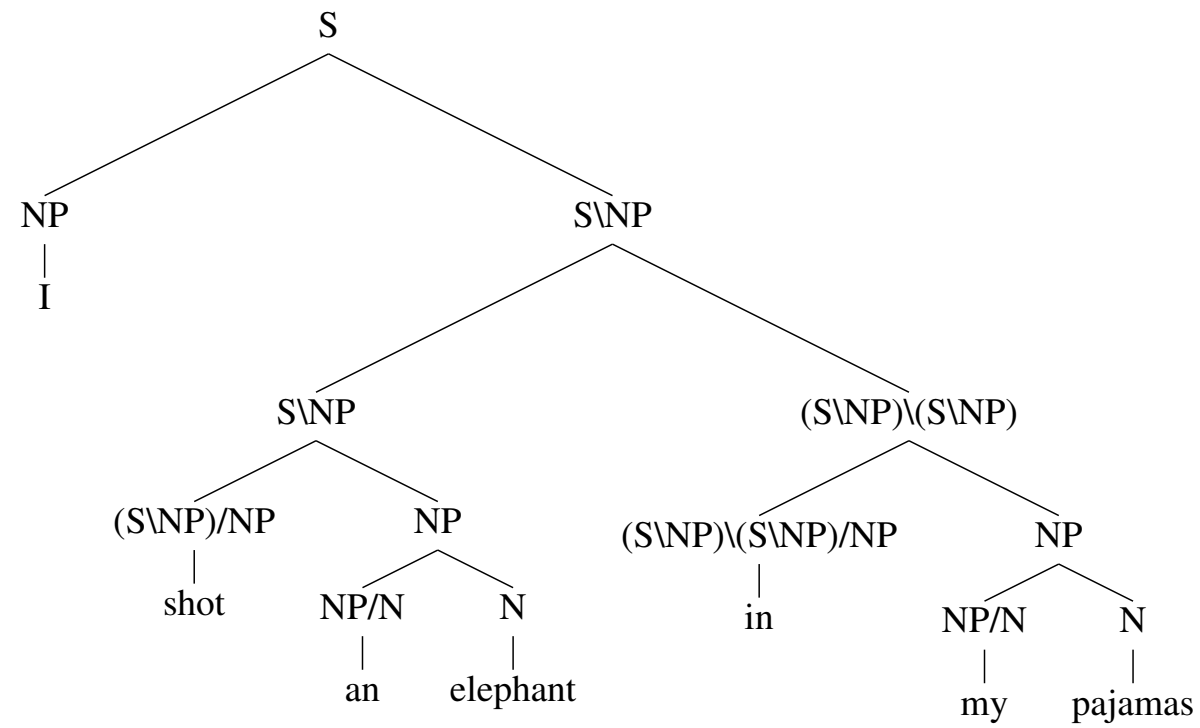
CCG Combinators

- Forward composition ($X/Y \ Y/Z \rightarrow X/Z$) and backward composition ($Y/Z \ X/Y \rightarrow X/Z$)



24085	N/N	adjective
22875	N	noun
2583	(S[dcI]\NP)/NP	transitive verb (declarative)
2107	S[adj]\NP	predicative adjectives (man is old)
1679	(S[b]\NP)/NP	transitive verb (bare infinitive)
1628	(N/N)/(N/N)	adjective adjective pairs
1431	S[pss]\NP	intransitive verb (past participles)
1385	(S[ng]\NP)/NP	transitive verb (present participle)
1308	N[num]	numerals
1227	S[dcI]\NP	intransitive verb (present participle)
1112	(S\NP)\(S\NP)	adverbs

Most frequent types in CCGBank lexicon



S

S\NP

S\NP

(S\NP)\(S\NP)

NP

NP

NP

(S\NP)/NP

NP/N

N

(S\NP)\(S\NP)/NP

NP/N

N

I

shot

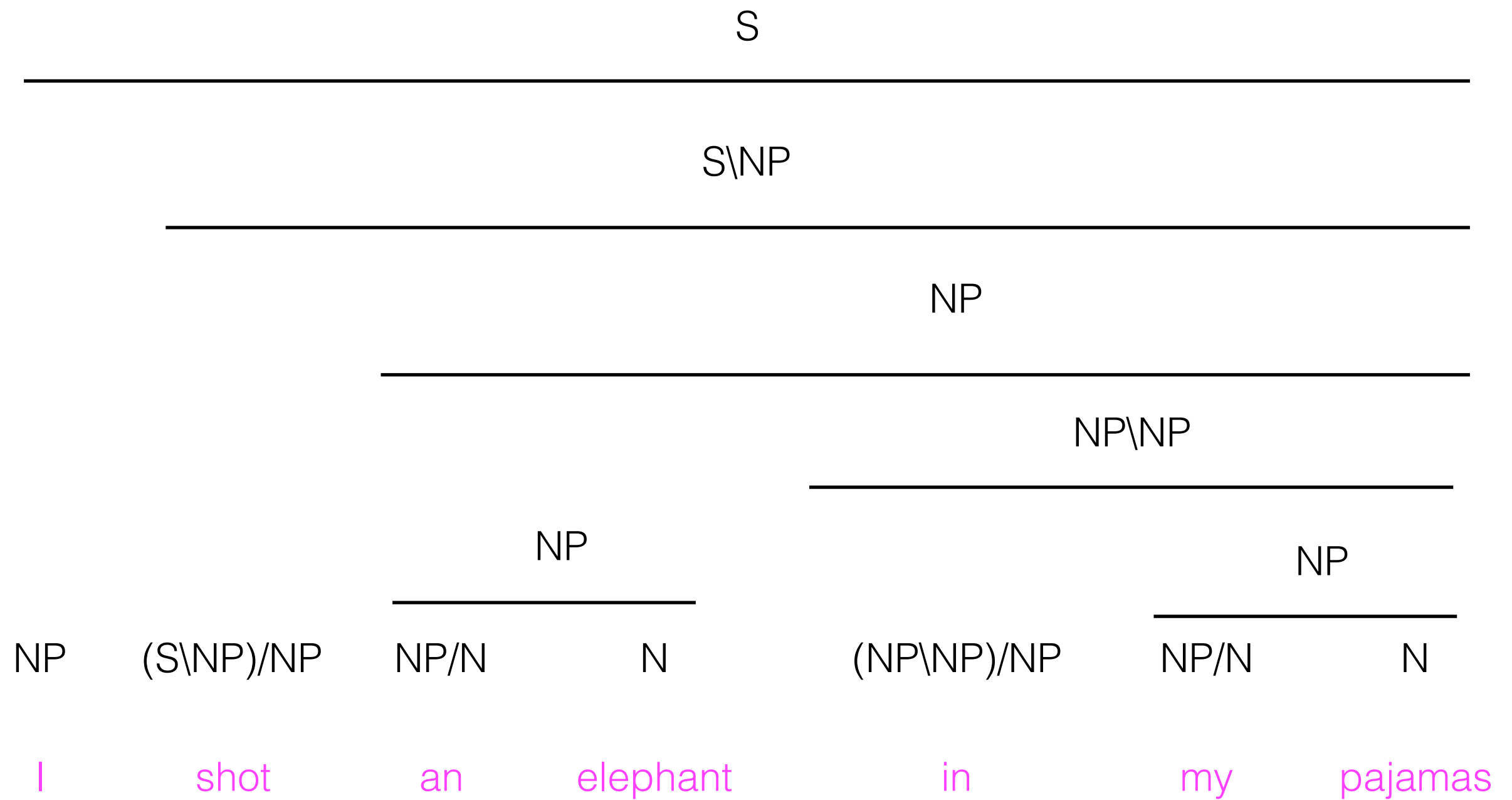
an

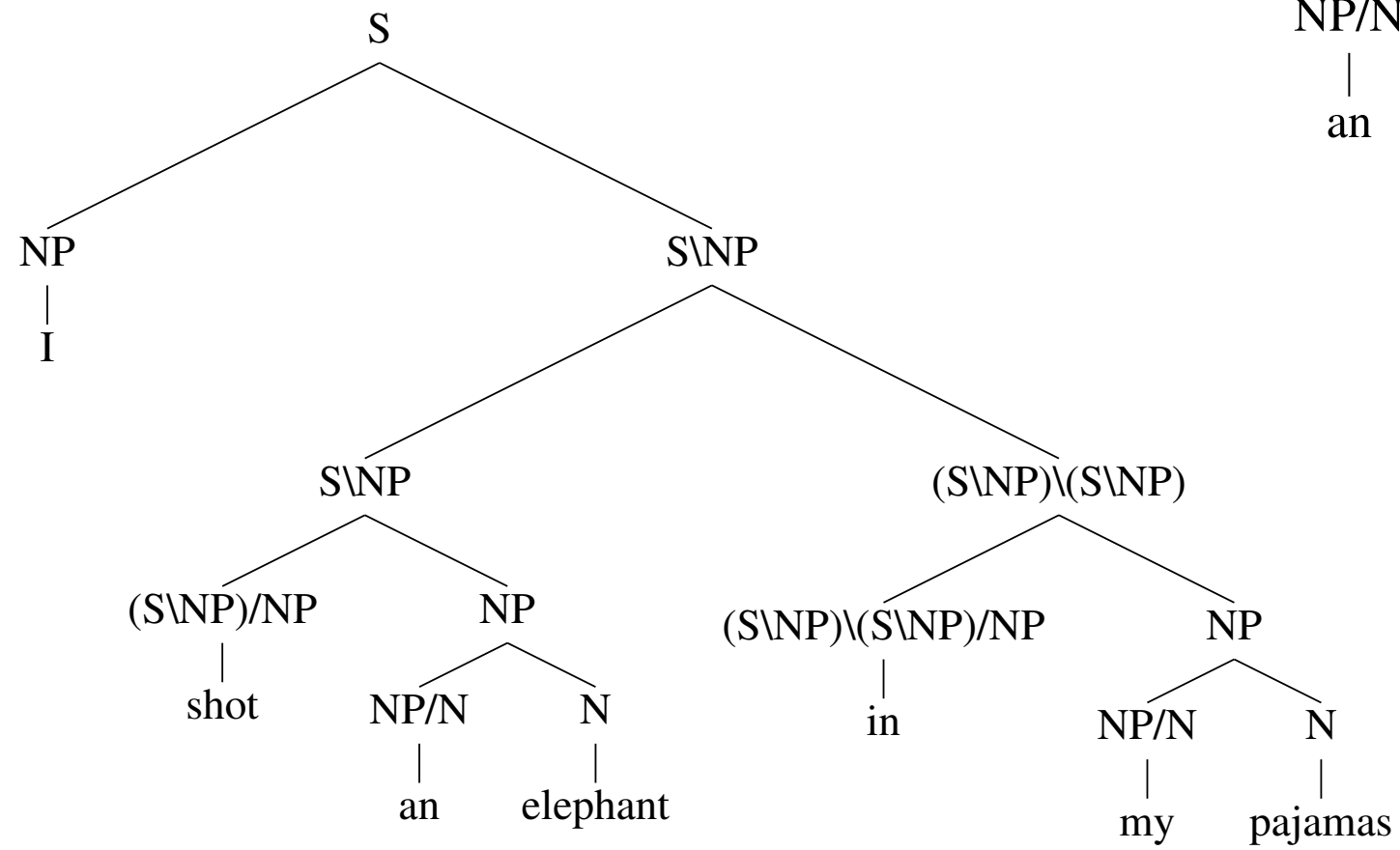
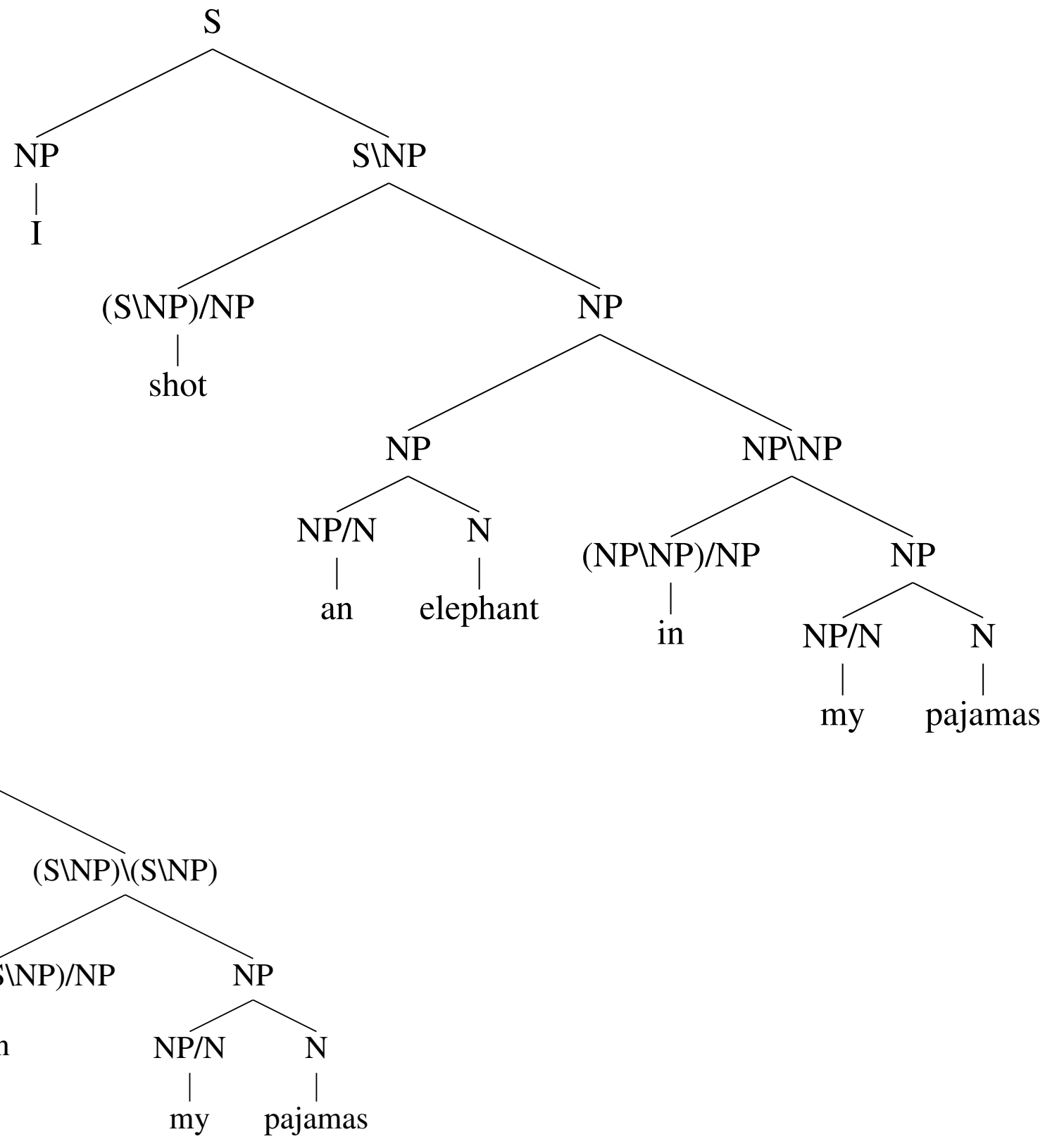
elephant

in

my

pajamas





I	shot	an	elephant	in	my	pajamas
NP [0,1]						
	(S\NP)/NP [1,2]					
		NP/N [2,3]				
			N [3,4]			
				(NP\NP)/NP (S\NP)\(S\NP)/ NP		
					NP/N [5,6]	
						N [6,7]

I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP					
	(S\NP)/NP [1,2]					
		NP/N [2,3]				
			N [3,4]			
				(NP\NP)/NP (S\NP)\(S\NP)/ NP		
					NP/N [5,6]	
						N [6,7]

I	shot	an	elephant	in	my	pajamas
---	------	----	----------	----	----	---------

NP [0,1]	S/NP	∅				
	(S\NP)/NP [1,2]	∅				
		NP/N [2,3]				
			N [3,4]			
				(NP\NP)/NP (S\NP)\(S\NP)/ NP		
					NP/N [5,6]	
						N [6,7]

(Leaving out forward composition for clarity)

I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP	∅				
	(S\NP)/NP [1,2]	∅				
		NP/N [2,3]	NP			
			N [3,4]			
				(NP\NP)/NP (S\NP)\(S\NP)/ NP		
					NP/N [5,6]	
						N [6,7]

I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP	∅				
	(S\NP)/NP [1,2]	∅	S\NP			
		NP/N [2,3]	NP			
			N [3,4]			
				(NP\NP)/NP (S\NP)\(S\NP)/ NP		
					NP/N [5,6]	
						N [6,7]

I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP	∅	S			
	(S\NP)/NP [1,2]	∅	S\NP			
		NP/N [2,3]	NP			
			N [3,4]			
				(NP\NP)/NP (S\NP)\(S\NP)/ NP		
					NP/N [5,6]	
						N [6,7]

I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP	∅	S	∅	∅	
	(S\NP)/NP [1,2]	∅	S\NP	∅	∅	
		NP/N [2,3]	NP	∅	∅	
			N [3,4]	∅	∅	
				(NP\NP)/NP (S\NP)\(S\NP)/ NP	∅	
					NP/N [5,6]	
						N [6,7]

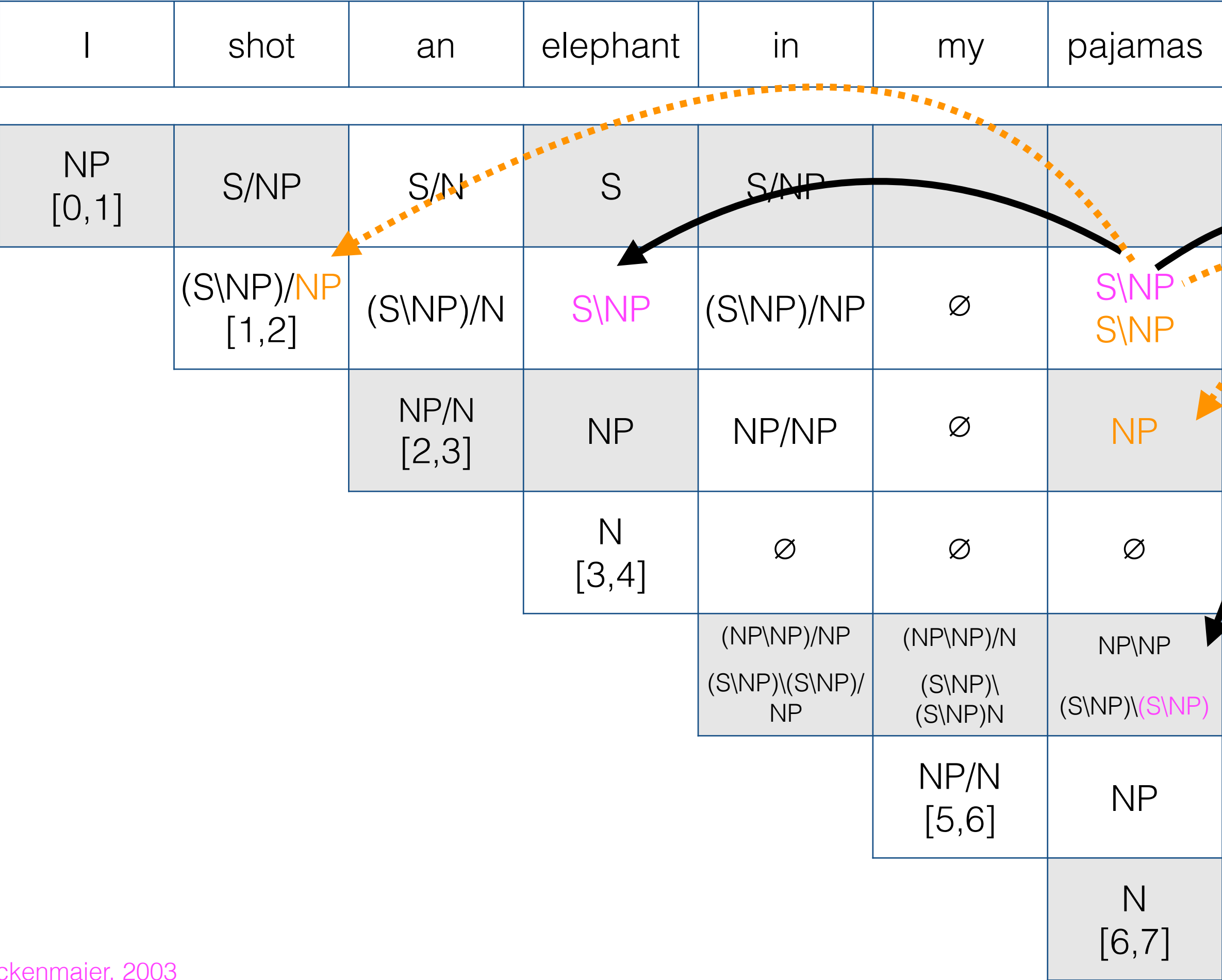
I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP	∅	S	∅	∅	
	(S\NP)/NP [1,2]	∅	S\NP	∅	∅	
		NP/N [2,3]	NP	∅	∅	
			N [3,4]	∅	∅	
				(NP\NP)/NP (S\NP)\(S\NP)/ NP	∅	
					NP/N [5,6]	NP
						N [6,7]

I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP	∅	S	∅	∅	
	(S\NP)/NP [1,2]	∅	S\NP	∅	∅	
		NP/N [2,3]	NP	∅	∅	
			N [3,4]	∅	∅	
				(NP\NP)/NP (S\NP)\(S\NP)/ NP	∅	NP\NP (S\NP)\(S\NP)
					NP/N [5,6]	NP
						N [6,7]

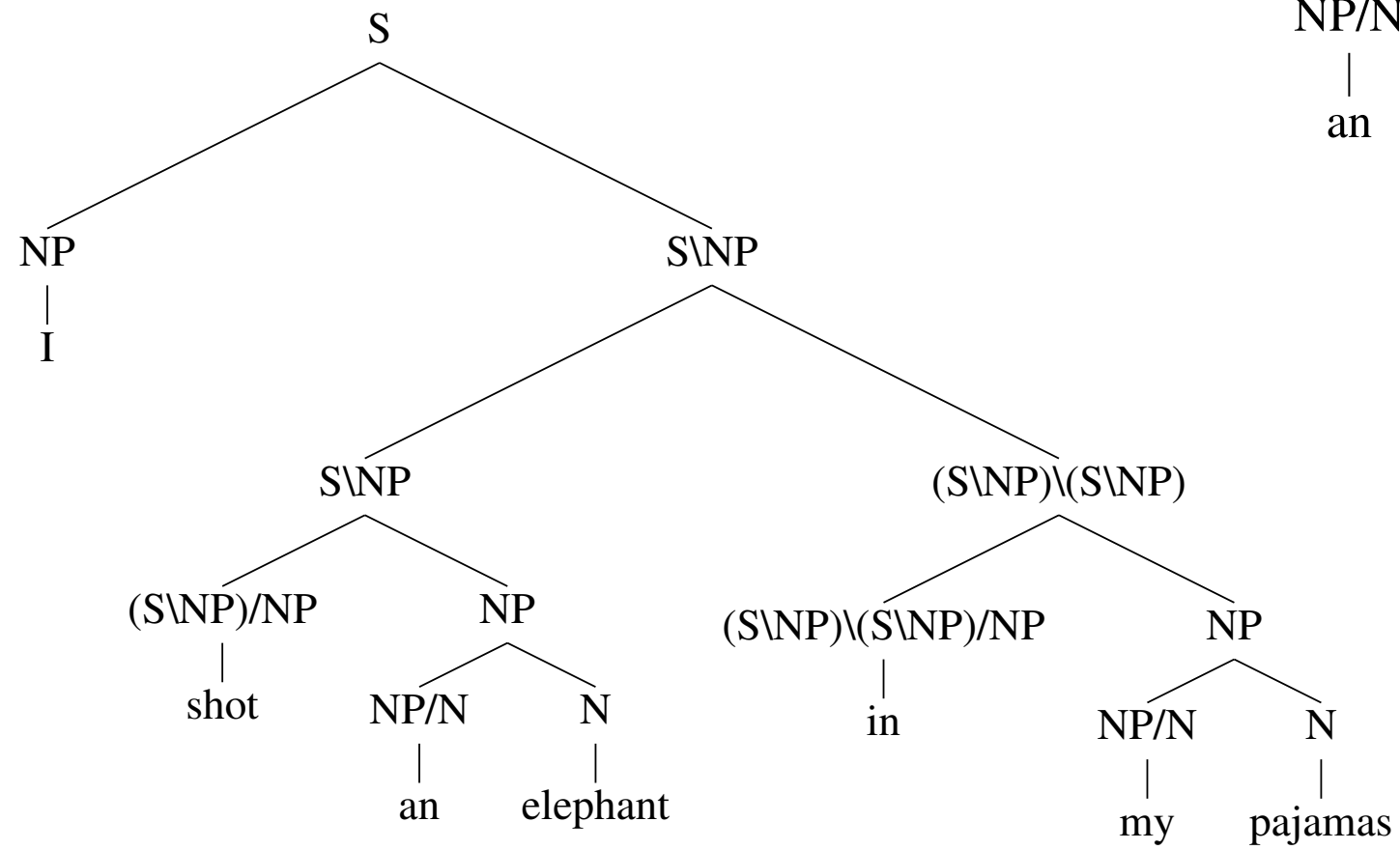
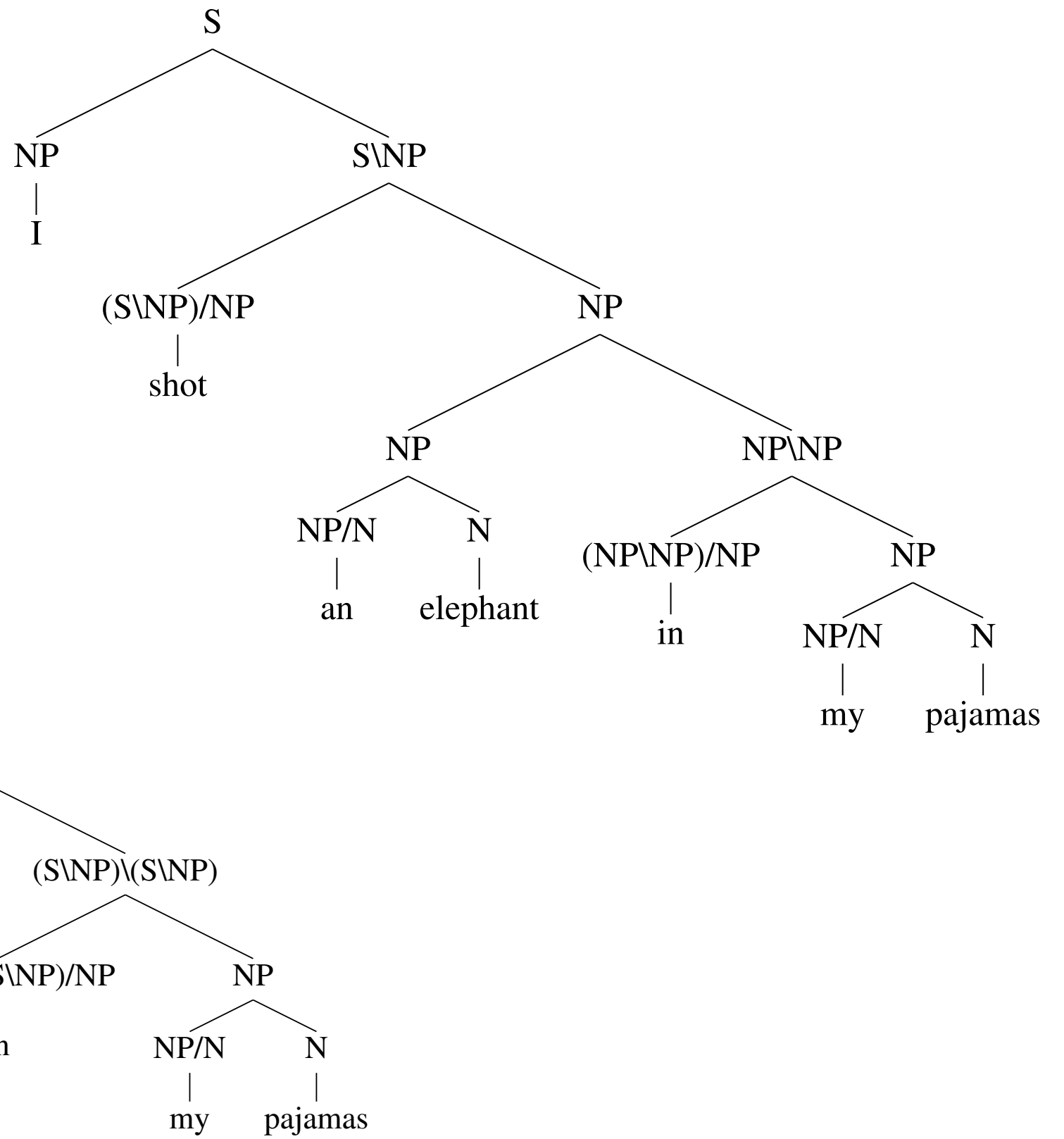
I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP	∅	S	∅	∅	
	(S\NP)/NP [1,2]	∅	S\NP	∅	∅	
		NP/N [2,3]	NP	∅	∅	
			N [3,4]	∅	∅	∅
				(NP\NP)/NP (S\NP)\(S\NP)/ NP	∅	NP\NP (S\NP)\(S\NP)
					NP/N [5,6]	NP
						N [6,7]

I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP	∅	S	∅	∅	
	(S\NP)/NP [1,2]	∅	S\NP	∅	∅	
		NP/N [2,3]	NP	∅	∅	NP
			N [3,4]	∅	∅	∅
				(NP\NP)/NP (S\NP)\(S\NP)/ NP	∅	NP\NP (S\NP)\(S\NP)
					NP/N [5,6]	NP
						N [6,7]

I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP	∅	S	∅	∅	
	(S\NP)/NP [1,2]	∅	S\NP	∅	∅	S\NP S\NP
		NP/N [2,3]	NP	∅	∅	NP
			N [3,4]	∅	∅	∅
				(NP\NP)/NP (S\NP)\(S\NP)/ NP	∅	NP\NP (S\NP)\(S\NP)
					NP/N [5,6]	NP
						N [6,7]



I	shot	an	elephant	in	my	pajamas
NP [0,1]	S/NP	S/N	S	S/NP		S S
	(S\NP)/NP [1,2]	(S\NP)/N	S\NP	(S\NP)/NP	∅	S\NP S\NP
		NP/N [2,3]	NP	NP/NP	∅	NP
			N [3,4]	∅	∅	∅
				(NP\NP)/NP (S\NP)\(S\NP)/ NP	(NP\NP)/N (S\NP)\ (S\NP)N	NP\NP (S\NP)\(S\NP)
					NP/N [5,6]	NP
						N [6,7]



CCG Lexicon

- Ambiguity in CCG comes from the **lexicon**, not grammar

CCG Lexicon

I	NP	1438
I	N	22
I	N/N	4
I	N\N	3

shot	N	25
shot	(S[dcI]\NP)/NP	8
shot	S[dcI]\NP	7
shot	S[pss]\NP	5
shot	(S[pss]\NP)/NP	1

elephant	N	5
elephant	N/N	1

an	NP[nb]/N	3685
an	(NP\NP)/N	76
an	((S\NP)\(S\NP))/N	16
an	((((S\NP)\(S\NP))\((S\NP)\(S\NP))))/N	8
an	N/N	3
an	NP/NP	3
an	((S\NP)\(S\NP))/((S\NP)\(S\NP))	2
an	(N/N)/(N/N)	2
an	(S[qem]/(S[dcI]/NP))/N	2
an	((S/S)\(S/S))/N	1
an	(S\S)/N	1
an	,	1
an	NP	1

CCG Lexicon

in	(NP\NP)/NP	8013
in	((S\NP)\(S\NP))/NP	7035
in	PP/NP	1644
in	(S/S)/NP	374
in	(S\NP)\(S\NP)	279
in	((S\NP)/(S\NP))/NP	241
in	((S\NP)\(S\NP))/(S[ng]\NP)	155
in	((((S\NP)\(S\NP))\((S\NP)\(S\NP))))/NP	125
in	(NP\NP)/(S[ng]\NP)	121
in	((S[adj]\NP)\(S[adj]\NP))/NP	110
in	PP/(S[ng]\NP)	106

in	(NP\NP)/NP
in	((S\NP)\(S\NP))/NP
in	PP/NP
in	(S/S)/NP
in	(S\NP)\(S\NP)
in	((S\NP)/(S\NP))/NP
in	((S\NP)\(S\NP))/(S[ng]\NP)
in	((((S\NP)\(S\NP))\((S\NP)\(S\NP)))\NP
in	(NP\NP)/(S[ng]\NP)
in	((S[adj]\NP)\(S[adj]\NP))/NP
in	PP/(S[ng]\NP)
in	(S\S)/NP
in	(PP\PP)/NP
in	((((S\NP)\(S\NP))/(S[to]\NP))/N
in	S[adj]\NP
in	((NP\NP)\(NP\NP))/NP
in	(NP\NP)/(S[adj]\NP)
in	(N\N)/NP
in	((((S\NP)\(S\NP))\((S\NP)\(S\NP)\(S\NP)))\NP
in	(NP/NP)/NP
in	((S\NP)\(S\NP))/(S[adj]\NP)
in	(PP/(S[ng]\NP))/NP
in	((S\NP)/(S\NP))/(S[ng]\NP)
in	((S\NP)/(S\NP))/(S[adj]\NP)
in	(S/S)/(S[ng]\NP)

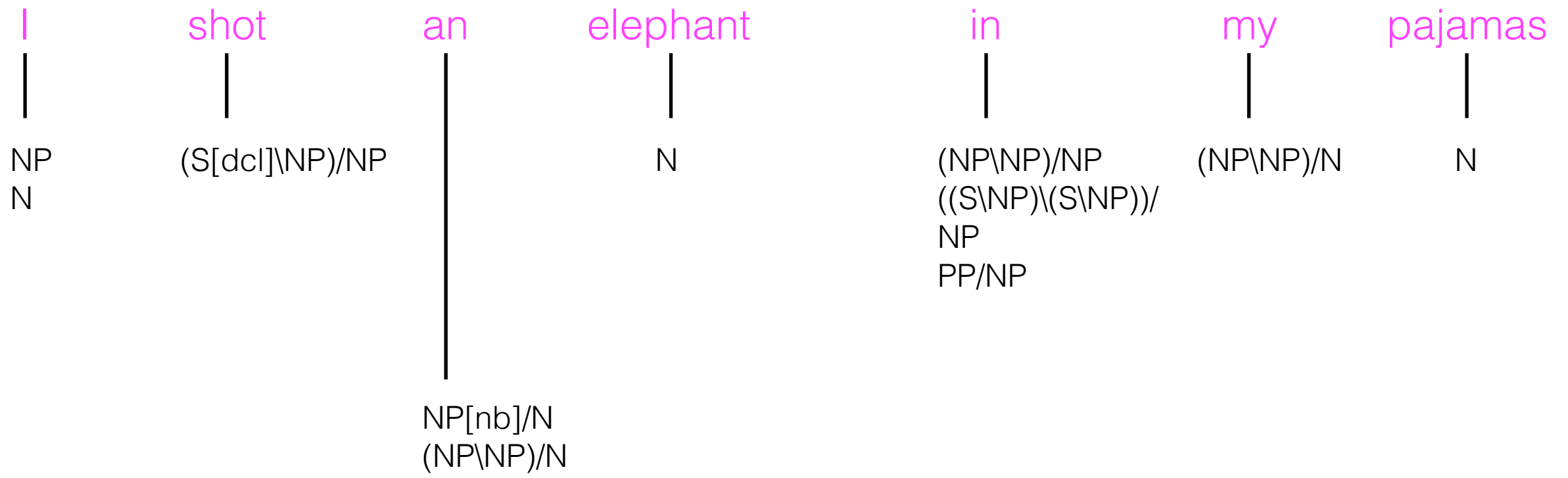
in	(S\NP)/NP
in	((((S\NP)/(S\NP))\((S\NP)/(S\NP)))/NP
in	((((S\NP)\(S\NP))/(S[to]\NP))/NP
in	((((S\NP)\(S\NP))/S[dcI])/N
in	((S[adj]\NP)\(S[adj]\NP))/(S[ng]\NP)
in	((S\NP)\(S\NP))/PP
in	(NP\NP)/PP
in	((N/N)\(N/N))/(N/N)
in	((NP\NP)/(NP\NP))/NP
in	((S\NP)\(S\NP))/S[em]
in	(NP\NP)/S[qem]
in	NP\NP
in	PP/(S[adj]\NP)
in	PP/PP
in	((((S\NP)\(S\NP))\NP)/NP
in	((S\NP)\(S\NP))/((S\NP)\(S\NP))
in	(N\N)/N
in	(S/(S[to]\NP))/N
in	(S\S)/(S[ng]\NP)
in	N\N
in	((((S\NP)\(S\NP))\((S\NP)\(S\NP)))\((S\NP)\(S\NP))/((S\NP)\(S\NP)))\NP
in	((N/N)\(N/N))/NP
in	(PP/PP)/NP
in	(S[adj]\NP)\(S[adj]\NP)
in	((((S\NP)\(S\NP))/NP)/NP

in	((PP/PP)\(PP/PP))/NP
in	((S[adj]\NP)/(S[adj]\NP))/NP
in	((S\NP)\(S\NP))/N
in	((S\NP)\(S\NP))/S[qem]
in	((S\S)\(S\S))/NP
in	(PP\NP)/NP
in	N/N
in	PP
in	PP/S[qem]
in	((((S\NP)\(S\NP))\((S\NP)\(S\NP)))\((S\NP)\(S\NP))\((S\NP)\(S\NP))\NP
in	((((S\NP)\(S\NP))\((S\NP)\(S\NP)))\((S\NP)\(S\NP))\((S\NP)\(S\NP))\NP
in	((S/S)\(S/S))\((S/S)\(S/S))/NP
in	((((S\NP)/(S\NP))/(S[to]\NP))/N
in	((((S\NP)\(S\NP))/(S[ng]\NP))/NP
in	((N/N)/(N/N))/NP
in	((NP\NP)/(S[ng]\NP))/NP
in	((NP\NP)/NP)/NP
in	((S/S)/(S/S))/NP
in	((S[dcI]\NP)/PP)/NP
in	((S\NP)\(S\NP))\((S\NP)\(S\NP))
in	((S\S)/(S[to]\NP))/N
in	((S\S)/NP)/NP
in	(N/N)/(N/N)
in	(N/N)/NP
in	(NP\NP)/(NP\NP)

Supertagging

- The CCG lexicon + very small set of combinators dictates the overall parse for a sentence.
- The base categories in CKY for syntactic parsing are POS tags (~45 in PTB)
- The base categories in CKY for CCG parsing are the lexical rules (~1,363 in CCGBank)
- This blows up the complexity of parsing; **supertagging** reduces the lexical categories to a much smaller set by predicting the likeliest tags **in context**.

I	shot	an	elephant	in	my	pajamas
NP	N		N	(NP\NP)/NP	(NP\NP)/N	N
N	(S[dcI]\NP)/NP		N/N	((S\NP)\(S\NP))/	(S/S)\(S/S)	
N/N	S[dcI]\NP			NP	N/N	
N\N	S[pss]\NP			PP/NP	NP[nb]/N	
	(S[pss]\NP)/NP			(S/S)/NP	NP\NP	
				(S\NP)\(S\NP)		
				((S\NP)/(S\NP))/		
				NP		
				((S\NP)\(S\NP))/		
				(S[ng]\NP)		
				((S\NP)\(S\NP))\		
				((S\NP)\(S\NP)))\		
				NP		
				(NP\NP)/(S[ng]		
				\NP)		
				((S[adj]\NP)\		
				(S[adj]\NP))/NP		
				PP/(S[ng]\NP)		
				(S\S)/NP		
				(PP\PP)/NP		
				((S\NP)\(S\NP))/		
				(S[to]\NP))/N		
				S[adj]\NP		
				((NP\NP)\		
				(NP\NP))/NP		
				(NP\NP)/(S[adj]		
				\NP)		
				...		



MEMM

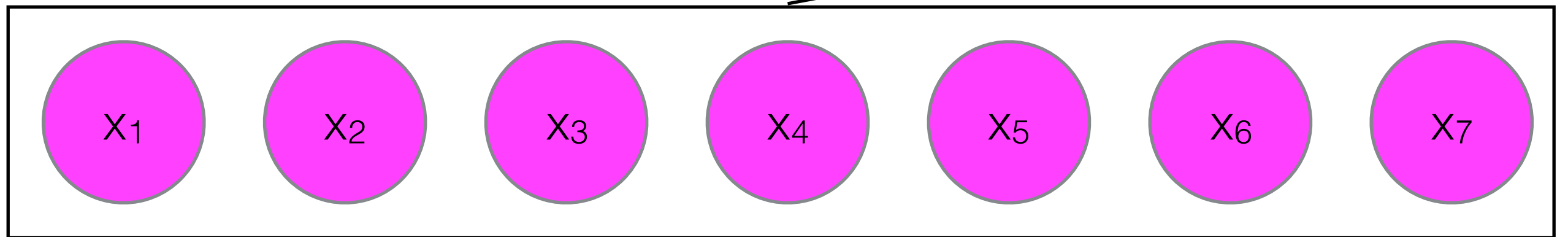
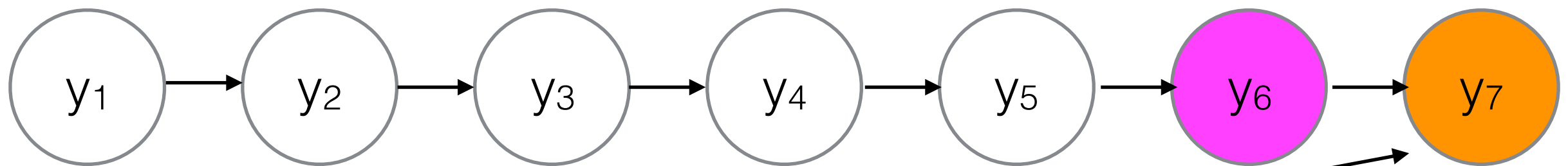
General maxent form

$$\arg \max_y P(y \mid x, \beta)$$

Maxent with first-order Markov
assumption: Maximum Entropy
Markov Model

$$\arg \max_y \prod_{i=1}^n P(y_i \mid y_{i-1}, x)$$

MEMM



Features

in: (NP\NP)/NP
((S\NP)\(S\NP))/NP

$$f(t_i, t_{i-1}; x_1, \dots, x_n)$$

Features are scoped over
the previous predicted
tag and the entire
observed input

feature	example
$x_i = \text{man}$	1
$t_{i-1} = \text{JJ}$	1
$i=n$ (last word of sentence)	1
x_i ends in -ly	0

Viterbi decoding

Viterbi for MEMM: max conditional probability

$$P(y \mid x)$$

$$v_t(y) = \max_{u \in \mathcal{Y}} [v_{t-1}(u) \times P(y_t = y \mid y_{t-1} = u, x, \beta)]$$

Supertagging

- The single best sequence is often still too errorful to be the input for CCG parsing.
- Rather than predicting the single best sequence, we can identify the top k tags for each word that have the highest probability.
- Note this is not $P(y_i \mid y_{i-1}, x, \beta)$ but rather $P(y_i \mid x, \beta)$ we can calculate using the forward-backward algorithm.

Semantics

- Semantic parsing with CCG is simply syntactic parsing, assuming mapping from syntactic primitives to logical forms.

CCG Lexicon

Utah	NP	utah
Idaho	NP	idaho
borders	$(S \backslash NP) / NP$	$\lambda x. \lambda y(\text{borders}(y, x))$
adjoins	$(S \backslash NP) / NP$	$\lambda x. \lambda y(\text{adjoins}(y, x))$
abuts	$(S \backslash NP) / NP$	$\lambda x. \lambda y(\text{abuts}(y, x))$

CCG Combinators

- Each combinator tells us what to do with the corresponding semantics
- Forward application:

$X/Y : f$	$Y : g$	$\rightarrow$	$X f(g)$	
$(S \backslash NP) / NP :$ $\lambda x . \lambda y (\text{borders}(y, x))$	$NP :$ idaho	$\rightarrow$	$S \backslash NP :$ $\lambda x . \lambda y (\text{borders}(y, x)(\text{idaho}))$	$S \backslash NP :$ $\lambda y (\text{borders}(y, \text{idaho}))$

Semantics

S

borders(utah,idaho)

S\NP

$\lambda y(\text{borders}(y,\text{idaho}))$

NP

utah

Utah

(S\NP)/NP

$\lambda x.\lambda y(\text{borders}(y,x))$

borders

NP

idaho

Idaho

Semantics

Utah	NP	utah
Idaho	NP	idaho
borders	(S\NP)/NP	$\lambda x.\lambda y(\text{borders}(y,x))$
adjoins	(S\NP)/NP	$\lambda x.\lambda y(\text{adjoins}(y,x))$
abuts	(S\NP)/NP	$\lambda x.\lambda y(\text{abuts}(y,x))$

Semantics

Utah	NP	utah
Idaho	NP	idaho
borders	(S\NP)/NP	$\lambda x.\lambda y(\text{borders}(y,x))$
adjoins	(S\NP)/NP	$\lambda x.\lambda y(\text{borders}(y,x))$
abuts	(S\NP)/NP	$\lambda x.\lambda y(\text{borders}(y,x))$

Semantics

Utah	NP	utah
Idaho	NP	idaho
borders	$(S \backslash NP) / NP$	$\lambda x. \lambda y(\text{borders}(y, x))$
adjoins	$(S \backslash NP) / NP$	$\lambda x. \lambda y(\text{borders}(y, x))$
abuts	$(S \backslash NP) / NP$	$\lambda x. \lambda y(\text{borders}(y, x))$

- Does Utah border **California**?

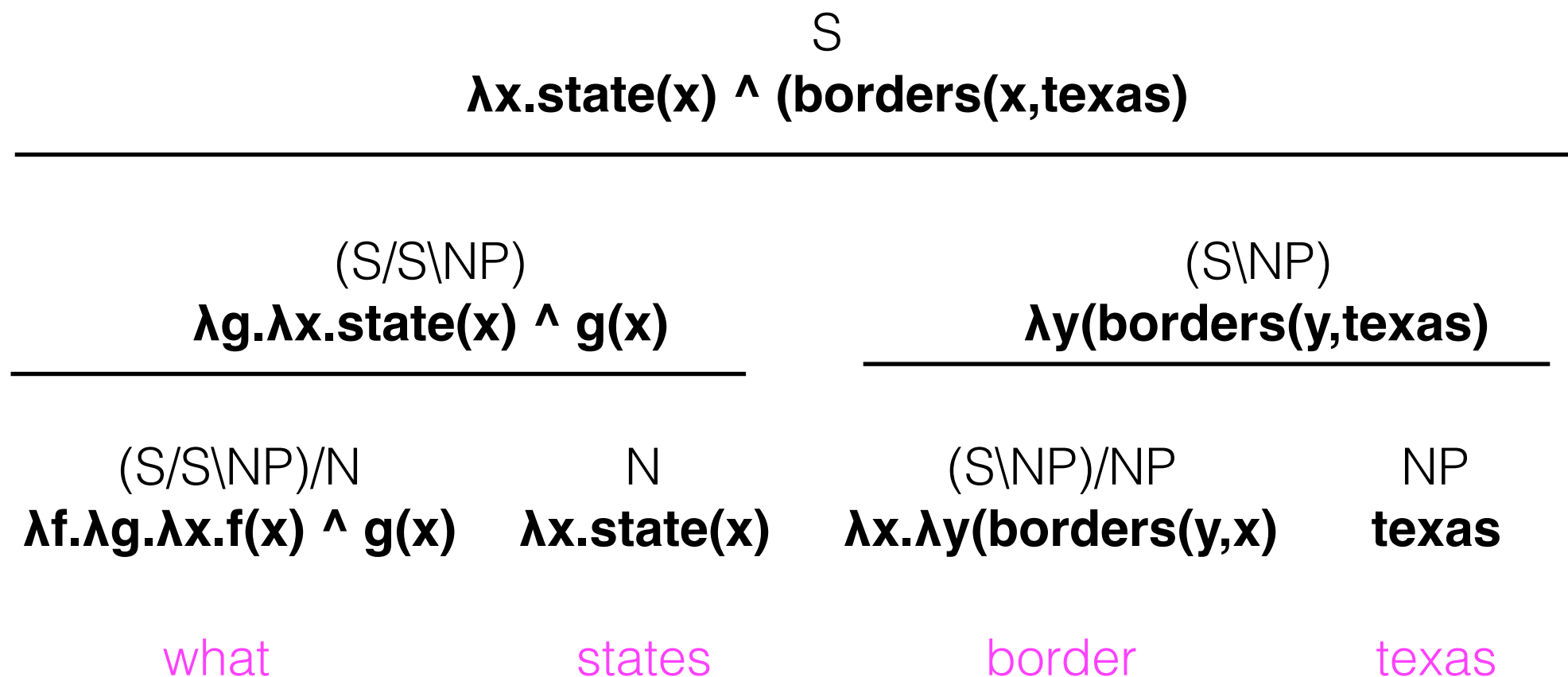
Semantics

- Semantic parsing with CCG is simply syntactic parsing, assuming mapping from syntactic primitives to logical forms.
- But this encounters two problems:
 - We don't have those manual mappings (task-specific).
 - We can't parse anything not in our lexicon.

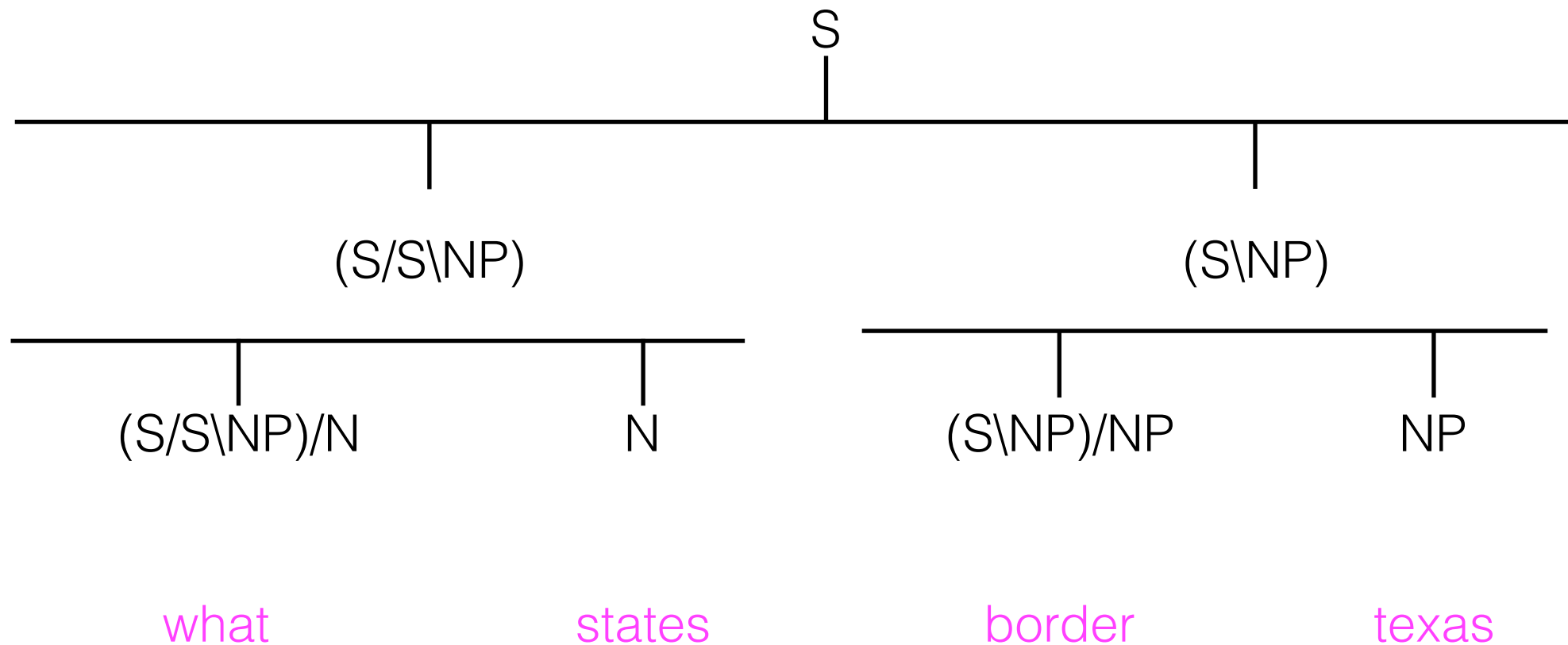
Learning from logical forms

- We can train a semantic parser in a number of ways:
 - Full derivational trees (CCGBank)
 - Logical forms (Zettlemoyer and Collins 2005)
 - Denotations (Berant et al. 2013)

Learning from trees



Learning from trees



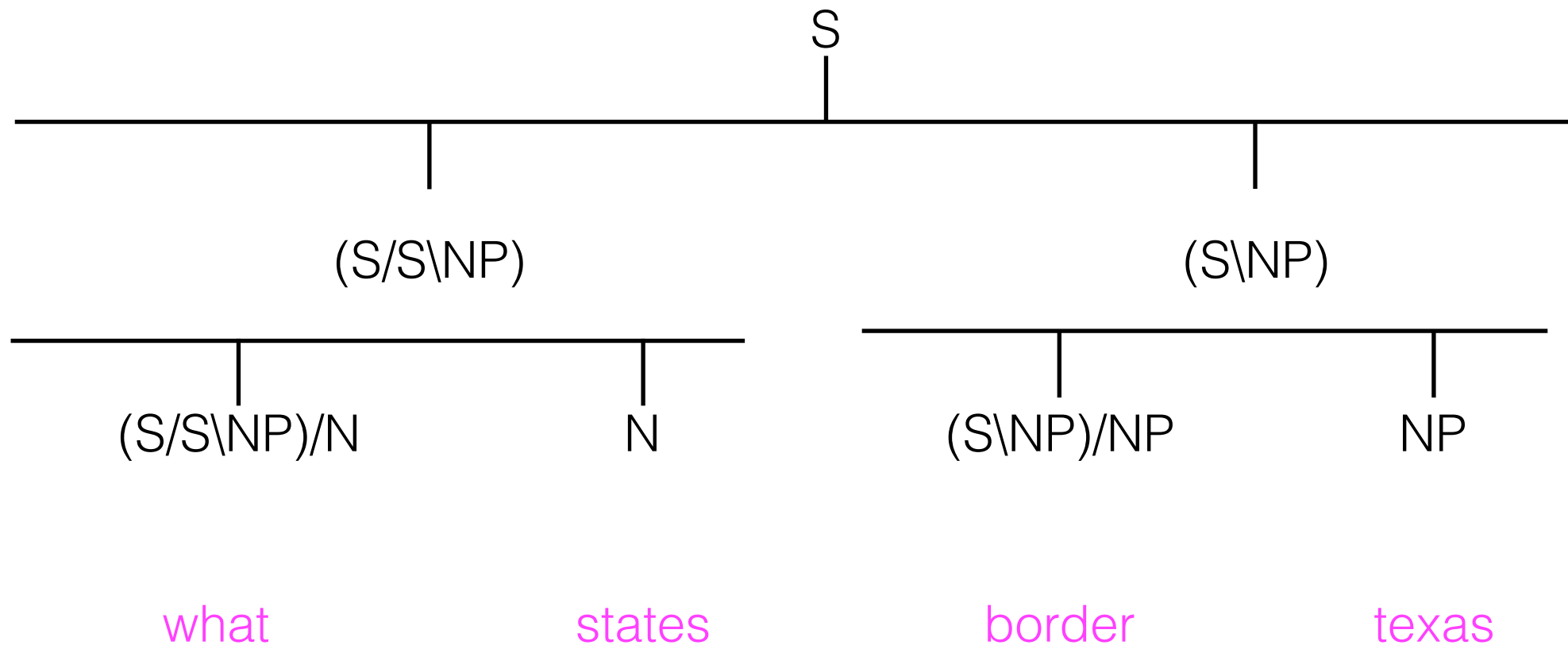
Learning from logical forms

sentence	what states border texas
logical form	$\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{texas})$

Two core ideas:

- We'll learn the lexicon (including the lambda expressions)
- We'll learn CCG parser from that lexicon, and treat the true tree as a **latent variable**

Learning from trees



We'll treat the tree (derivation) as a latent variable

Learning the lexicon

sentence	Utah borders Idaho
logical form	borders(utah,idaho)

- For a given sentence and logical form, return the set of lexicon entries that could have generated the logical form.

Utah	NP	utah
Idaho	NP	idaho
borders	$(S \backslash NP) / NP$	$\lambda x. \lambda y(\text{borders}(y, x))$

Learning the lexicon

sentence	Utah borders Idaho
logical form	borders(utah,idaho)

- For a given sentence and logical form, return the set of lexicon entries that could have generated the logical form.

All
subsequences
of x

$$\text{GENLEX}(S, L) = \{x := y \mid x \in W(S), y \in C(L)\}$$

All categories found
in logical form

Rules	
Input Trigger	Output Category
constant c	$NP : c$
arity one predicate p_1	$N : \lambda x.p_1(x)$
arity one predicate p_1	$S \setminus NP : \lambda x.p_1(x)$
arity two predicate p_2	$(S \setminus NP) / NP : \lambda x.\lambda y.p_2(y, x)$
arity two predicate p_2	$(S \setminus NP) / NP : \lambda x.\lambda y.p_2(x, y)$
arity one predicate p_1	$N / N : \lambda g.\lambda x.p_1(x) \wedge g(x)$
literal with arity two predicate p_2 and constant second argument c	$N / N : \lambda g.\lambda x.p_2(x, c) \wedge g(x)$
arity two predicate p_2	$(N \setminus N) / NP : \lambda x.\lambda g.\lambda y.p_2(x, y) \wedge g(x)$
an arg max / min with second argument arity one function f	$NP / N : \lambda g. \arg \max / \min(g, \lambda x.f(x))$
an arity one numeric-ranged function f	$S / NP : \lambda x.f(x)$

C(L)

Categories produced from logical form $\arg \max(\lambda x.state(x) \wedge borders(x, texas), \lambda x.size(x))$	
$NP : texas$	
$N : \lambda x.state(x)$	
$S \setminus NP : \lambda x.state(x)$	
$(S \setminus NP) / NP : \lambda x.\lambda y.borders(y, x)$	
$(S \setminus NP) / NP : \lambda x.\lambda y.borders(x, y)$	
$N / N : \lambda g.\lambda x.state(x) \wedge g(x)$	
$N / N : \lambda g.\lambda x.borders(x, texas) \wedge g(x)$	
$(N \setminus N) / NP : \lambda g.\lambda x.\lambda y.borders(x, y) \wedge g(x)$	
$NP / N : \lambda g. \arg \max(g, \lambda x.size(x))$	
$S / NP : \lambda x.size(x)$	

Rules	
Input Trigger	Output Category
constant c	$NP : c$
arity one predicate p_1	$N : \lambda x.p_1(x)$
arity one predicate p_1	$S \backslash NP : \lambda x.p_1(x)$
arity two predicate p_2	$(S \backslash NP) / NP : \lambda x.\lambda y.p_2(y, x)$
arity two predicate p_2	$(S \backslash NP) / NP : \lambda x.\lambda y.p_2(x, y)$
arity one predicate p_1	$N / N : \lambda g.\lambda x.p_1(x) \wedge g(x)$
literal with arity two predicate p_2 and constant second argument c	$N / N : \lambda g.\lambda x.p_2(x, c) \wedge g(x)$
arity two predicate p_2	$(N \backslash N) / NP : \lambda x.\lambda g.\lambda y.p_2(x, y) \wedge g(x)$
an arg max / min with second argument arity one function f	$NP / N : \lambda g.\text{arg max / min}(g, \lambda x.f(x))$
an arity one numeric-ranged function f	$S / NP : \lambda x.f(x)$

logical form	borders(utah,idaho)
utah	NP : utah
idaho	NP : idaho
borders	$(S \backslash NP) / NP : \lambda x.\lambda y.borders(y, x)$
borders	$(S \backslash NP) / NP : \lambda x.\lambda y.borders(x, y)$

Learning the lexicon

All
subsequences
of x

All categories found
in logical form

$$\text{GENLEX}(S, L) = \{x := y \mid x \in W(S), y \in C(L)\}$$

Utah	NP	utah
Idaho	NP	idaho
borders	NP	idaho
borders utah	(S\NP)/NP	$\lambda x.\lambda y(\text{borders}(y,x)$
borders	(S\NP)/NP	$\lambda x.\lambda y(\text{borders}(y,x)$
...	...	...

Learning from logical forms

- If we create a lexicon $\lambda_i = \text{initial lexicon } \Lambda_0 + \text{lexicon entries identified by GENLEX}$, we can find **many** parses for the sentence.

sentence	Utah borders Idaho
----------	--------------------

logical form	borders(utah,idaho)
utah	NP : utah
idaho	NP : idaho
idaho	NP : utah
utah	NP : idaho
borders	(S\NP)/NP : $\lambda x.\lambda y.\text{borders}(y,x)$
borders	(S\NP)/NP : $\lambda x.\lambda y.\text{borders}(x,y)$

Learning from logical forms

- Calculate the joint probability of a logical form L and derivation T for sentence S as:

$$P(L, T \mid S; \theta) = \frac{\exp(f(L, T, S)^\top \theta)}{\sum_{L, T} \exp(f(L, T, S)^\top \theta)}$$

sums over all valid trees/logical forms for the sentence

feature
Utah := NP : utah
Utah := NP : idaho
borders := (S\NP)/NP : $\lambda x. \lambda y. \text{borders}(y, x)$
borders := (S\NP)/NP : $\lambda x. \lambda y. \text{borders}(x, y)$

$f(L, T, S)$

Learning from logical forms

- For all <sentence, logical form> pairs in training data, maximize the probability of the logical form by marginalizing over the joint probability:

$$P(L \mid S; \theta) = \sum_T P(L, T \mid S; \theta)$$

Start with random values for θ ;
update with SGD

- Where

$$P(L, T \mid S; \theta) = \frac{\exp(f(L, T, S)^\top \theta)}{\sum_{L, T} \exp(f(L, T, S)^\top \theta)}$$

Learning from logical forms

- Learning from logical forms is means we don't need training data in the form of **full CCG derivations** + semantically enriched **lexicon**.
- But we do still need training data in the form of logical forms.

Utah borders Idaho	borders(utah,idaho)
number of dramas starring tom cruise	???

Learning from denotations

sentence	what states border texas
logical form	$\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{texas})$
denotation	new_mexico, oklahoma, arkansas, louisiana

sentence	number of dramas starring tom cruise
logical form	$\text{count}(\lambda x.\text{genre}(x, \text{drama}) \wedge \exists y.\text{performance}(x, y) \wedge \text{actor}(y, \text{tom_cruise}))$
denotation	28

Learning from denotations

sentence	what states border texas
logical form	$\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{texas})$
denotation	new_mexico, oklahoma, arkansas, louisiana

sentence	number of dramas starring tom cruise
logical form	$\text{count}(\lambda x.\text{genre}(x, \text{drama}) \wedge \exists y.\text{performance}(x, y) \wedge \text{actor}(y, \text{tom_cruise}))$
denotation	28

Learning from denotations

- How could we use the principles of **learning from logical forms** to learn from denotations?
- The meaning of a sentence is the set of possible worlds consistent with that statement.

Utah borders Idaho	TRUE
number of dramas starring tom cruise	28

Learning from denotations

- Basic idea: maximize the probability of the tree T / logical form z that, when executed against a knowledge base $\mathcal{K}$, yield the correct denotation y

$$\sum_{i=1}^N \log \sum_{T: \llbracket T.z \rrbracket_{\mathcal{K}} = y_i} P(T \mid S_i, \theta)$$

objective function

Why do we need CCG (or a syntactic representation)
at all?