CKY algorithm / PCFGs

CS 585, Fall 2019

Introduction to Natural Language Processing
http://people.cs.umass.edu/~miyyer/cs585/

Mohit Iyyer

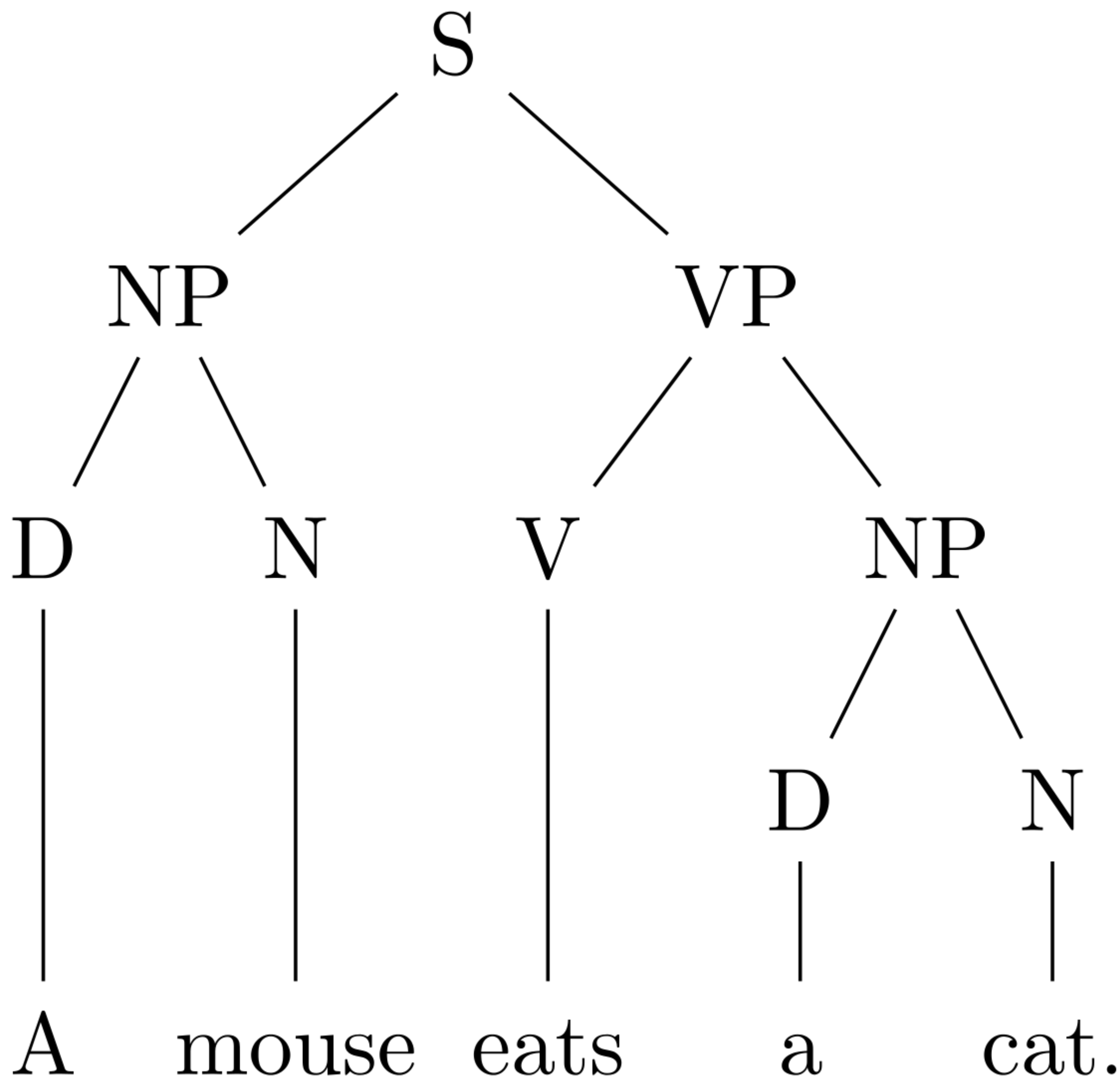
College of Information and Computer Sciences
University of Massachusetts Amherst

some slides from Brendan O'Connor

questions from last time…

- milestone 2 due 11/21
- extra credit due 12/11
- HW3: we'll start it in class on 11/19. you'll have to answer a few short questions after that, will be due after Thanksgiving
-

today we'll be doing *parsing*: given a CFG, how do we use it to parse a sentence?


S
NP
VP
D
N
V
NP
D
N
A
mouse
eats
a
cat.

why parsing?

- historically: good way to obtain features for downstream tasks
- today: can sometimes (not always) use syntax to improve neural models
- always useful for chunking text into phrases
- parsing makes for good *probe* tasks on top of neural models (next class)

Formal Definition of Context-Free Grammar

- A context-free grammar G is defined by four parameters: ***N*, *Σ*, *R*, *S***

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$,
	where A is a non-terminal,
	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$
S	a designated **start symbol** and a member of N

let’s start with a simple CFG

- S > NP VP
- NN > “dog”
- NP > DT JJ NN

first, let's convert this to Chomsky Normal Form (CNF)

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$,
	where A is a non-terminal,
	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$
S	a designated **start symbol** and a member of N

β is either a single terminal from Σ
or a pair of non-terminals from N

converting the simple CFG

- S > NP VP
- NN > “dog”
- NP > DT JJ NN
    - NP > X NN
    - X > DT JJ

we can convert any CFG to a CNF. this is a necessary preprocessing step for the basic CKY alg., produces binary trees!

Parsing!

- Given a sentence and a CNF, we want to **search** through the space of all possible parses for that sentence to find:
    - any valid parse for that sentence
    - all valid parses
    - the most probable parse
- Two approaches
    - **bottom-up**: start from the words and attempt to construct the tree
    - **top-down**: start from START symbol and keep expanding until you can construct the sentence

Pros and cons of each?

Ambiguity in parsing

Syntactic ambiguity is endemic to natural language:[1]

- Attachment ambiguity: we eat sushi with chopsticks, I shot an elephant in my pajamas.
- Modifier scope: southern food store
- Particle versus preposition: The puppy tore up the staircase.
- Complement structure: The tourists objected to the guide that they couldn't hear.
- Coordination scope: "I see," said the blind man, as he picked up the hammer and saw.
- Multiple gap constructions: The chicken is ready to eat

[1] Examples borrowed from Dan Klein

today: CKY algorithm

- Cocke-Kasami-Younger (independently discovered, also known as CYK)
- a *bottom-up* parser for CFGs (and PCFGs)

“I shot an elephant in my pajamas. How he got into my pajamas, I'll never know.”
— Groucho Marx

today: CKY algorithm

- Cocke-Kasami-Younger (independently discovered, also known as CYK)
- a *bottom-up* parser for CFGs (and PCFGs)

“I shot an elephant in my pajamas. How he got into my pajamas, I'll never know.”
— Groucho Marx

CKY is a dynamic programming algorithm.
Where else have we seen such an algorithm?

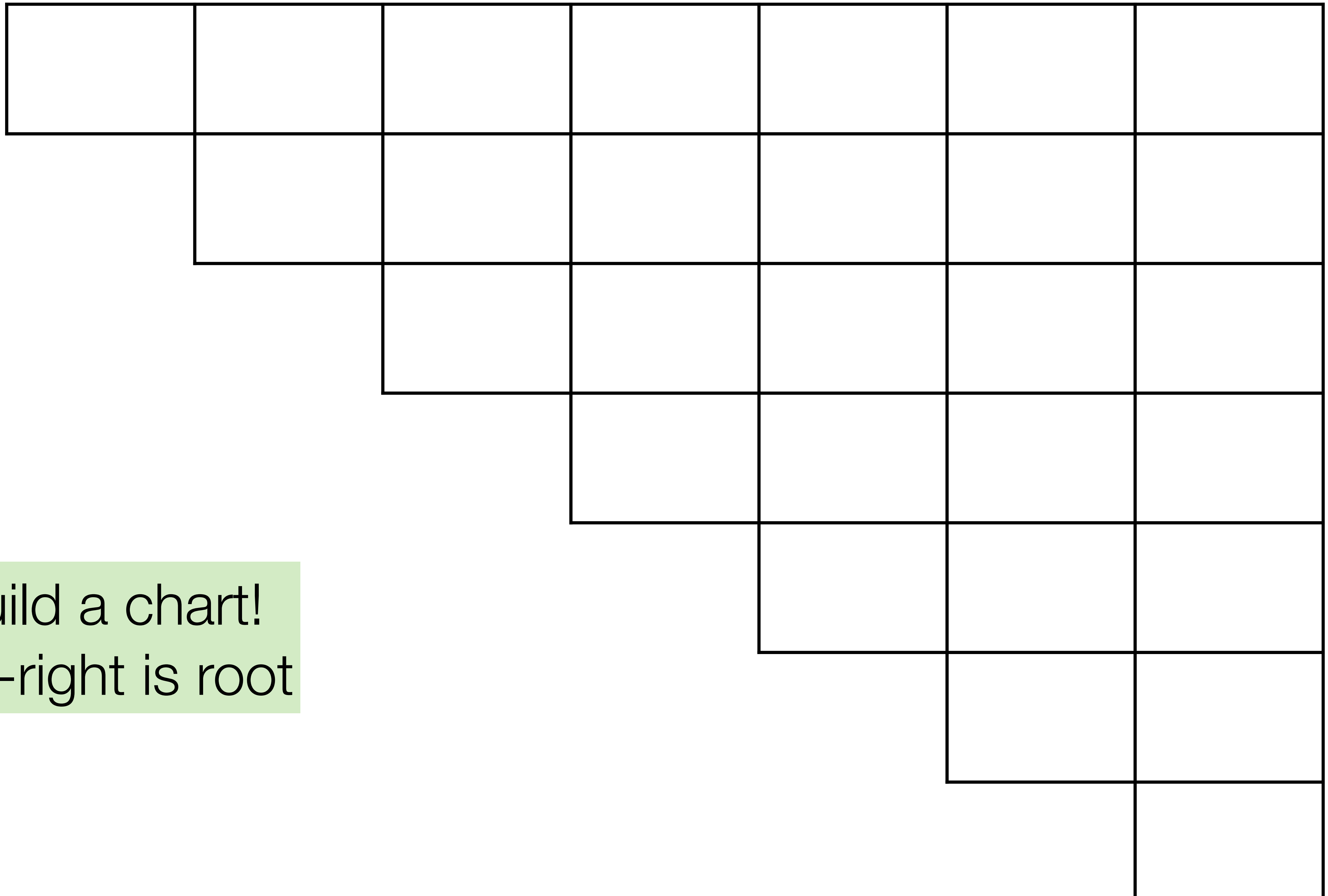
let’s say I have this CNF

- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

- DET ▸ “an”
- VBD ▸ “shot”
- NP ▸ “pajamas”
- NP ▸ “elephant”
- NP ▸ “I”
- PRP ▸ “I”
- IN ▸ “in”
- PRP$▸ “my”

example adapted from David Bamman

I shot an elephant in my pajamas



build a chart!
top-right is root

I	shot	an	elephant	in	my	pajamas
NP / PRP						
	VBD					
		DET				
			NP			
				IN		
					PRP$	
						NP

fill in first level (words)
with possible derivations

I	shot	an	elephant	in	my	pajamas
NP / PRP						
	VBD					
		DET				
			NP			
				IN		
					PRP$	
						NP

onto the second level!

I	shot	an	elephant	in	my	pajamas
NP / PRP						
	VBD					
		DET				
			NP			
				IN		
					PRP$	
						NP

onto the second level!

this cell spans
the phrase “I shot”

I	shot	an	elephant	in	my	pajamas
NP / PRP						
	VBD					
		DET				
			NP			
				IN		
					PRP$	
						NP

onto the second level!

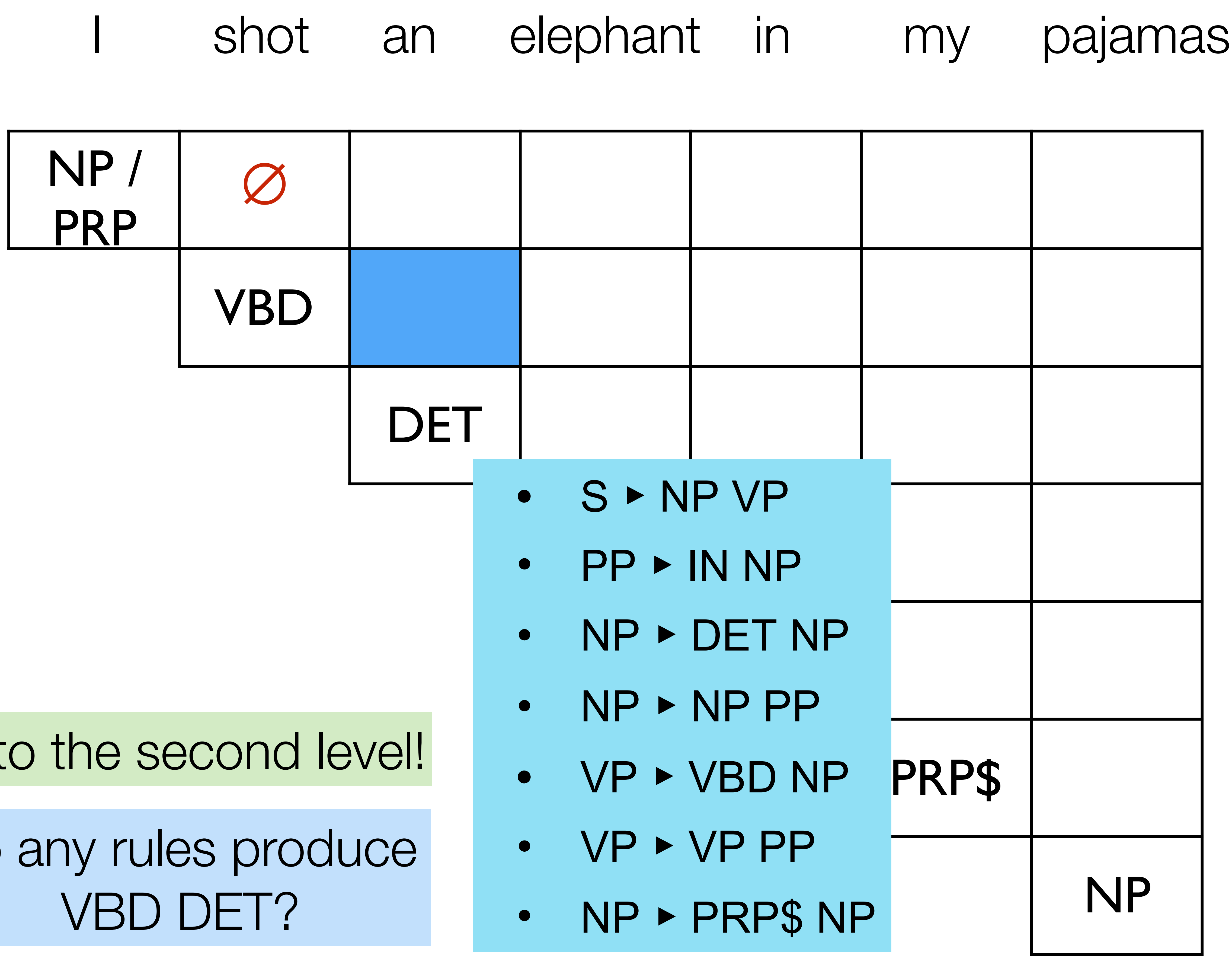
what does this cell span?

I	shot	an	elephant	in	my	pajamas
NP / PRP						
	VBD					
		DET				
					PRP$	
						NP

- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

onto the second level!

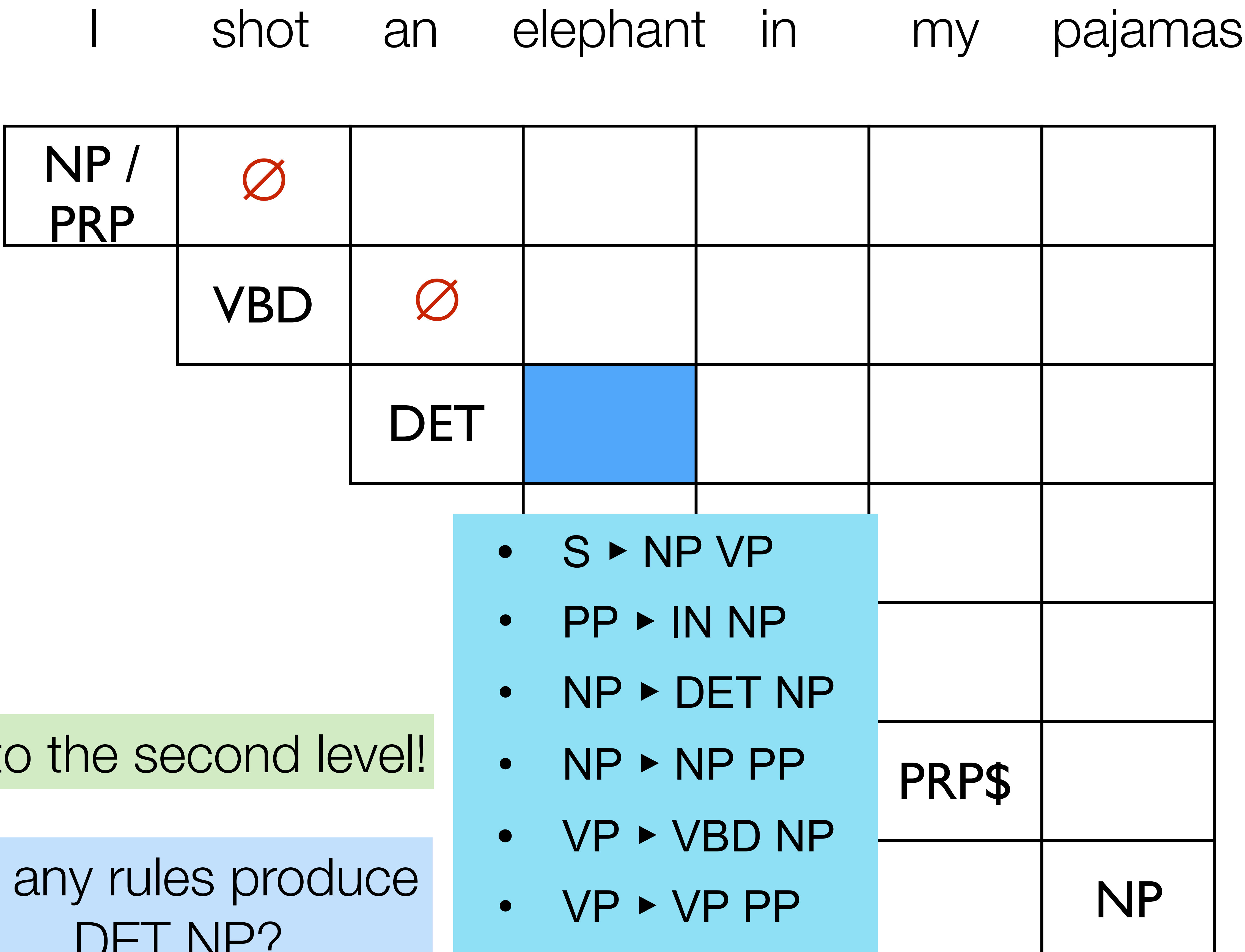
do any rules produce NP VBD or PRP VBD?



- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

onto the second level!

do any rules produce VBD DET?



- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

onto the second level!

do any rules produce DET NP?

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅					
	VBD	∅				
		DET	NP			
					PRP$	
						NP

- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

onto the second level!

do any rules produce DET NP? Yes!

NP ▸ DET NP

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅					
	VBD	∅				
		DET	NP			
			NP	∅		
				IN	∅	
					PRP$	NP
						NP

onto the third level!

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅					
	VBD	∅				
		DET	NP			
			NP	∅		
				IN	∅	
					PRP$	NP
						NP

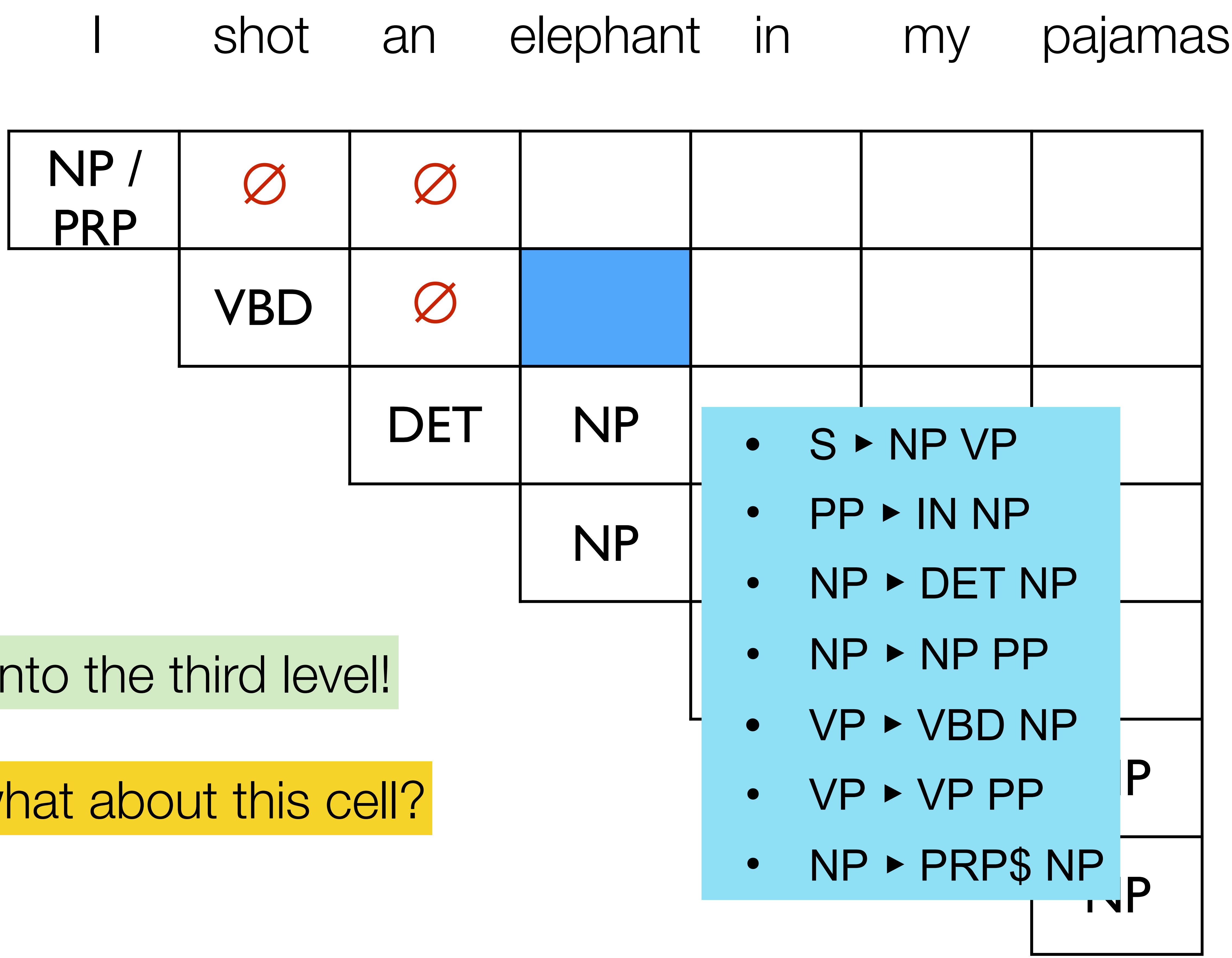
onto the third level!

two ways to form
“I shot an”:
I + shot an

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅					
	VBD	∅				
		DET	NP			
			NP	∅		
				IN	∅	
					PRP$	NP
						NP

onto the third level!

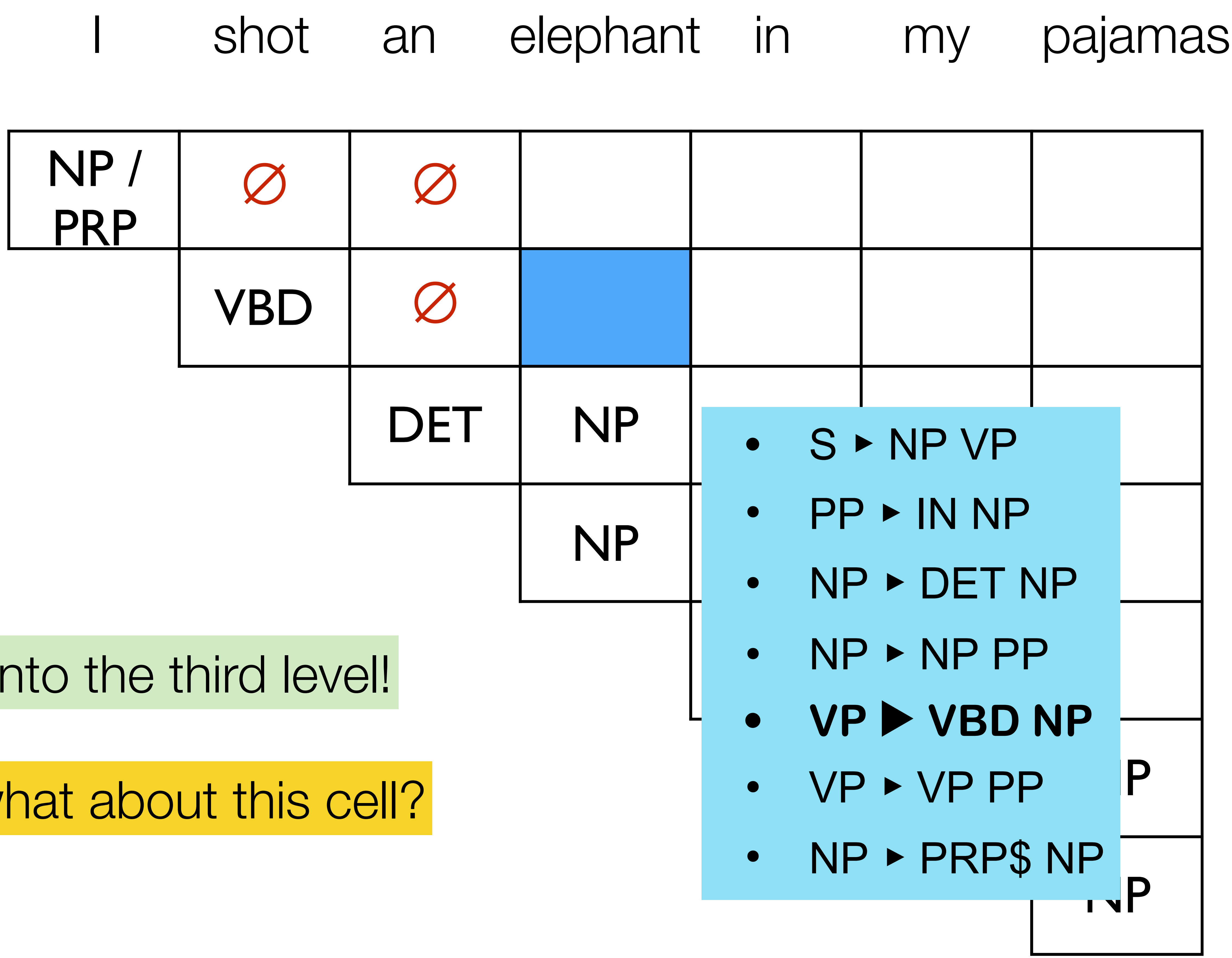
two ways to form
“I shot an”:
I + shot an
I shot + an



- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

onto the third level!

what about this cell?



- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- **VP ▶ VBD NP**
- VP ▸ VP PP
- NP ▸ PRP$ NP

onto the third level!

what about this cell?

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅				
	VBD	∅	VP			
		DET	NP			
			NP	∅		
				IN	∅	
					PRP$	NP
						NP

onto the third level!

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅				
	VBD	∅	VP			
		DET	NP	∅		
			NP	∅	∅	
				IN	∅	PP
					PRP$	NP
						NP

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅				
	VBD	∅	VP			
		DET	NP	∅		
			NP	∅	∅	
				IN	∅	PP
					PRP$	NP
						NP

onto the fourth level!

what are our options here?

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅				
	VBD	∅	VP			
		DET	NP	∅		
			NP			
						PP
						NP
						NP

- **S ▶ NP VP**
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

onto the fourth level!

what are our options here?

NP VP
PRP VP

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅	S			
	VBD	∅	VP	∅		
		DET	NP	∅	∅	
			NP	∅	∅	NP
				IN	∅	PP
					PRP$	NP
						NP

onto the fourth level!

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅	S	∅		
	VBD	∅	VP	∅	∅	
		DET	NP	∅	∅	
			NP	∅	∅	NP
				IN	∅	PP
					PRP$	NP
						NP

- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅	S	∅		
	VBD	∅	VP	∅	∅	
		DET	**NP**	∅	∅	
			NP	∅	∅	**NP**
				IN	∅	**PP**
					PRP$	NP
						NP

- S ▸ NP VP
- PP ▸ IN NP
- **NP ▶ DET NP**
- **NP ▶ NP PP**
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅	S	∅		
	VBD	∅	VP	∅	∅	
		DET	NP	∅	∅	NP_1 / NP_2
			NP	∅	∅	NP
				IN	∅	PP
					PRP$	NP
						NP

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅	S	∅	∅	
	VBD	∅	VP	∅	∅	
		DET	NP	∅	∅	NP_1 / NP_2
			NP	∅	∅	NP
				IN	∅	PP
					PRP$	NP
						NP

- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅	S	∅	∅	
	VBD	∅	VP	∅	∅	
		DET	NP	∅	∅	NP_1 / NP_2
			NP	∅	∅	NP
				IN	∅	PP
					PRP$	NP
						NP

- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- **VP ▶ VBD NP**
- **VP ▶ VP PP**
- NP ▸ PRP$ NP

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅	S	∅	∅	
	VBD	∅	VP	∅	∅	VP_1 / VP_2 / VP_3
		DET	NP	∅	∅	NP_1 / NP_2
			NP	∅	∅	NP
				IN	∅	PP
					PRP$	NP
						NP

- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- **VP ▶ VBD NP**
- **VP ▶ VP PP**
- NP ▸ PRP$ NP

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅	S	∅	∅	
	VBD	∅	VP	∅	∅	VP_1 / VP_2 / VP_3
		DET	NP	∅	∅	NP_1 / NP_2
			NP	∅	∅	NP
				IN	∅	PP
					PRP$	NP
						NP

finally, the root!

- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅	S	∅	∅	
	VBD	∅	VP	∅	∅	**VP_1 / VP_2 / VP_3**
		DET	NP	∅	∅	NP_1 / NP_2
			NP	∅	∅	NP
				IN	∅	PP
					PRP$	NP
						NP

finally, the root!

- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

S > NP VP1
S > NP VP2
S > NP VP3

I	shot	an	elephant	in	my	pajamas
NP / PRP	∅	∅	S	∅	∅	S_1 / S_2 / S_3
	VBD	∅	VP	∅	∅	VP_1 / VP_2 / VP_3
		DET	NP	∅	∅	NP_1 / NP_2
			NP	∅	∅	NP
				IN	∅	PP
					PRP$	NP
						NP

finally, the root!

- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

S > NP VP1
S > NP VP2
S > NP VP3

three valid parses!

how do we recover the full derivation of the valid parses S_1 / S_2 / S_3?

CKY runtime?

function CKY-PARSE(*words, grammar*) **returns** *table*

  for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
    for all $\{A \mid A \rightarrow words[j] \in grammar\}$
      $table[j-1, j] \leftarrow table[j-1, j] \cup A$
    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
      for $k \leftarrow i+1$ **to** $j-1$ **do**
        for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
          $table[i,j] \leftarrow table[i,j] \cup A$

Figure 12.5 The CKY algorithm.

three nested loops, each O(n) where n is # words

$O(n^3)$

how to find best parse?

- use PCFG (*probabilistic* CFG): same as CFG except each rule A > β in the grammar is associated with a probability p(β | A)

- can compute probability of a parse *T* by just multiplying rule probabilities of the rules *r* that make up *T*

$$p(T) = \prod_{r \in T} p(\beta_r | A_r)$$

- S ▸ NP VP, 0.4
- PP ▸ IN NP, 0.1
- NP ▸ DET NP, 0.3
- NP ▸ NP PP, 0.1
- VP ▸ VBD NP, 0.2
- VP ▸ VP PP, 0.3
- NP ▸ PRP$ NP, 0.5

- DET ▸ “an”, 0.9
- VBD ▸ “shot”, 0.3
- NP ▸ “pajamas”, 0.8
- NP ▸ “elephant”, 0.9
- NP ▸ “I”, 0.2
- PRP ▸ “I”, 0.6
- IN ▸ “in”, 0.9
- PRP$▸ “my”, 0.8

I	shot	an	elephant	in	my	pajamas
NP (0.2) / PRP (0.6)						
	VBD (0.3)					
		DET (0.9)				
			NP (0.8)			
				IN (0.9)		
					PRP$ (0.8)	
						NP (0.8)

fill in first level (words) with possible derivations and probabilities

I	shot	an	elephant	in	my	pajamas
NP (0.2) / PRP (0.6)	∅					
	VBD (0.3)	∅				
		DET (0.9)	NP			
			NP (0.8)	∅		
				IN (0.9)	∅	
					PRP$ (0.8)	NP
						NP (0.8)

how do we compute this cell's probability?

I	shot	an	elephant	in	my	pajamas
NP (0.2) / PRP (0.6)	∅					
	VBD (0.3)	∅				
		DET (0.9)	NP (0.22)			
			NP (0.8)	∅		
				IN (0.9)	∅	
					PRP$ (0.8)	NP (0.32)
						NP (0.8)

how do we compute this cell's probability?

$$p(\text{DET NP} \mid \text{NP}) * P(\text{cell}_{\text{DET}}) * P(\text{cell}_{\text{NP}})$$

$$= 0.3 * 0.9 * 0.8$$

$$= 0.22$$

I	shot	an	elephant	in	my	pajamas
NP (-1.6) / PRP (-0.51)	∅	∅	S (-6.8)	∅	∅	
	VBD (-1.2)	∅	VP (-4.3)	∅	∅	
		DET (-0.11)	NP (-1.5)	∅	∅	NP_1 / NP_2
			NP (-0.22)	∅	∅	NP (-6.0)
				IN (-0.11)	∅	PP (-3.5)
					PRP$ (-0.22)	NP (-1.1)
						NP (-0.22)

let's switch to log space and fill out the table some more

I	shot	an	elephant	in	my	pajamas
NP (-1.6) / PRP (-0.51)	∅	∅	S (-6.8)	∅	∅	
	VBD (-1.2)	∅	VP (-4.3)	∅	∅	
		DET (-0.11)	NP (-1.5)	∅	∅	NP_1 / NP_2
			NP (-0.22)	∅	∅	NP (-6.0)
				IN (-0.11)	∅	PP (-3.5)
					PRP$ (-0.22)	NP (-1.1)
						NP (-0.22)

$p(NP_1) = ?$

$p(NP_2) = ?$

I	shot	an	elephant	in	my	pajamas
NP (-1.6) / PRP (-0.51)	∅	∅	S (-6.8)	∅	∅	
	VBD (-1.2)	∅	VP (-4.3)	∅	∅	
		DET (-0.11)	NP (-1.5)	∅	∅	NP_1 (-7.31) / NP_2 (-7.30)
			NP (-0.22)	∅	∅	NP (-6.0)
				IN (-0.11)	∅	PP (-3.5)
					PRP$ (-0.22)	NP (-1.1)
						NP (-0.22)

do we have to store both NPs?

I	shot	an	elephant	in	my	pajamas
NP (-1.6) / PRP (-0.51)	∅	∅	S (-6.8)	∅	∅	
	VBD (-1.2)	∅	VP (-4.3)	∅	∅	VP_1 / VP_2
		DET (-0.11)	NP (-1.5)	∅	∅	NP (-7.3)
			NP (-0.22)	∅	∅	NP (-6.0)
				IN (-0.11)	∅	PP (-3.5)
					PRP$ (-0.22)	NP (-1.1)
						NP (-0.22)

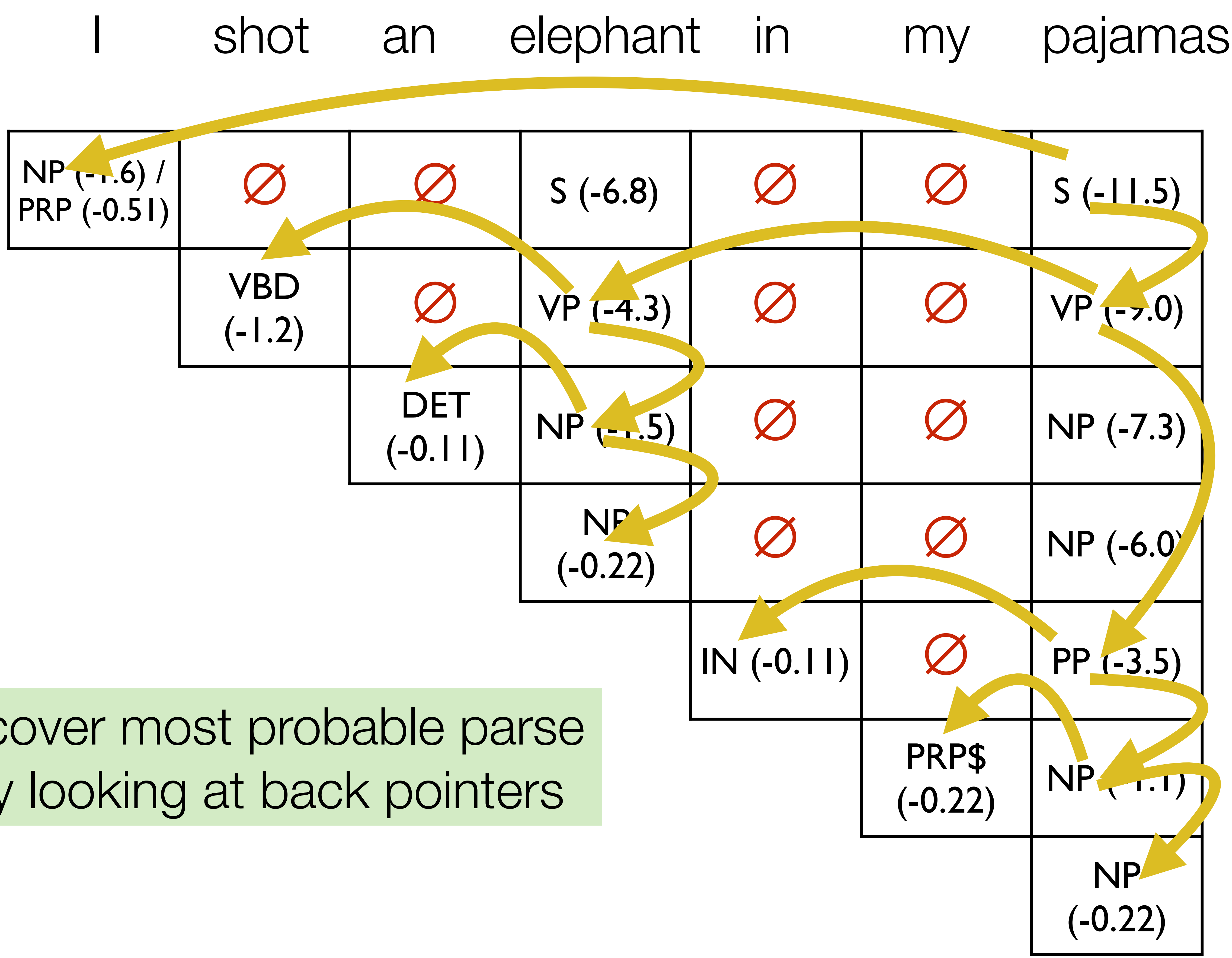
$p(VP_1) = ?$

$p(VP_2) = ?$

I	shot	an	elephant	in	my	pajamas
NP (-1.6) / PRP (-0.51)	∅	∅	S (-6.8)	∅	∅	
	VBD (-1.2)	∅	VP (-4.3)	∅	∅	VP_1 (-10.1) / VP_2 (-9.0)
		DET (-0.11)	NP (-1.5)	∅	∅	NP (-7.3)
			NP (-0.22)	∅	∅	NP (-6.0)
				IN (-0.11)	∅	PP (-3.5)
					PRP$ (-0.22)	NP (-1.1)
						NP (-0.22)

do we need to store both VPs?

I	shot	an	elephant	in	my	pajamas
NP (-1.6) / PRP (-0.51)	∅	∅	S (-6.8)	∅	∅	S (-11.5)
	VBD (-1.2)	∅	VP (-4.3)	∅	∅	VP (-9.0)
		DET (-0.11)	NP (-1.5)	∅	∅	NP (-7.3)
			NP (-0.22)	∅	∅	NP (-6.0)
				IN (-0.11)	∅	PP (-3.5)
					PRP$ (-0.22)	NP (-1.1)
						NP (-0.22)



recover most probable parse by looking at back pointers

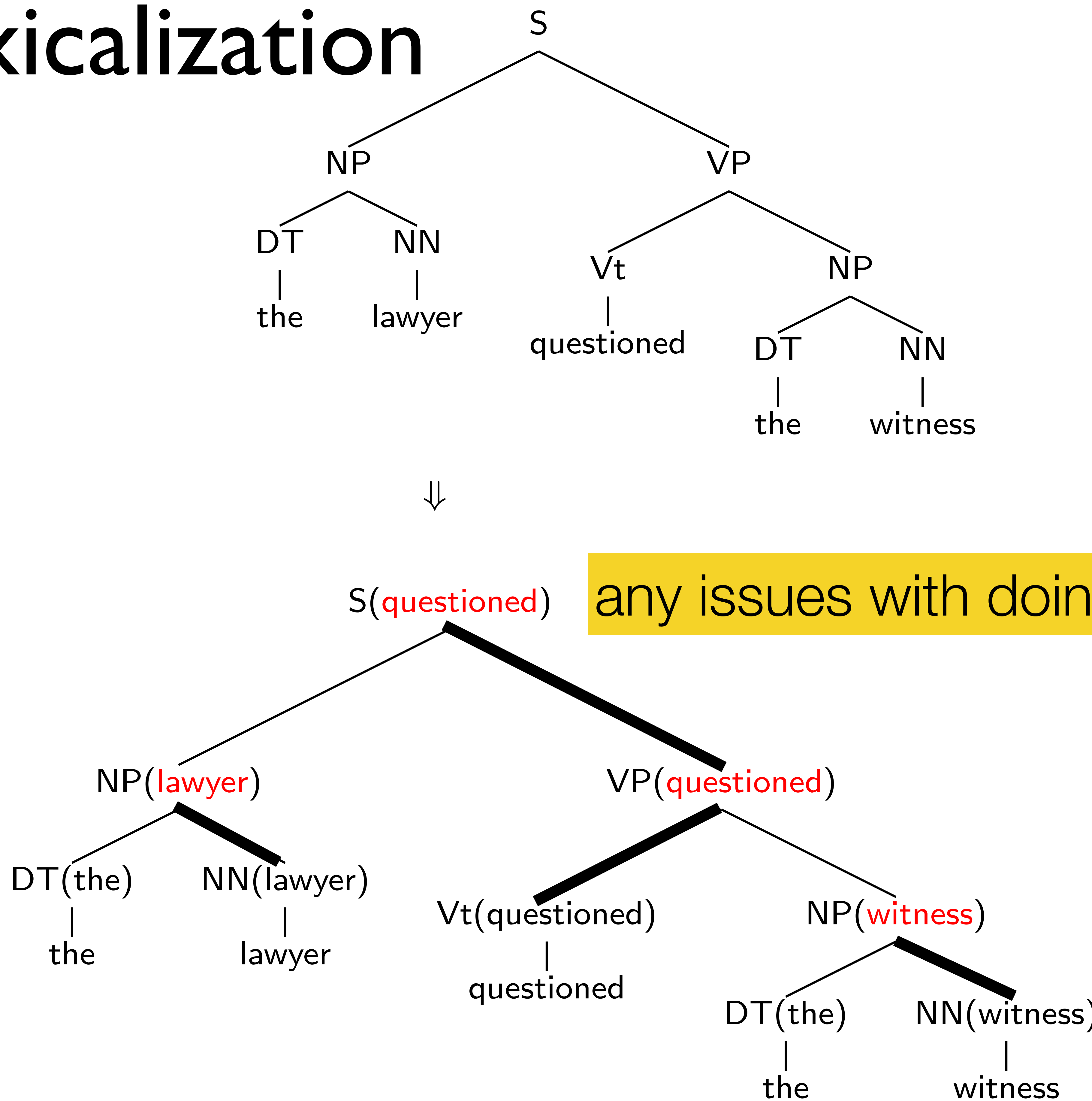
issues w/ PCFGs

- independence assumption: each rule's probability is independent of the rest of the tree!!!
- doesn't take into account location in the tree or what words are involved (for A>BC)
  - John saw the man with the hat
  - John saw the moon with the telescope

add more info to PCFG!

- **How to make good attachment decisions?**
  - Enrich PCFG with
    - parent information: what's above me?
    - lexical information via head rules
      - VP[fight]: a VP headed by "fight"
  - (or better, word/phrase embedding-based generalizations: e.g. recurrent neural network grammars (RNNGs))

Lexicalization



any issues with doing this?

where do we get the PCFG probabilities?

- given a treebank, we can just compute the MLE estimate by counting and normalizing

$$P(\alpha \to \beta|\alpha) = \frac{\text{Count}(\alpha \to \beta)}{\sum_\gamma \text{Count}(\alpha \to \gamma)} = \frac{\text{Count}(\alpha \to \beta)}{\text{Count}(\alpha)}$$

- without a treebank, we can use the *inside-outside algorithm* to estimate probabilities by
  1. randomly initializing probabilities
  2. computing parses
  3. computing expected counts for rules
  4. re-estimate probabilities
  5. repeat!