CKY algorithm / PCFGs

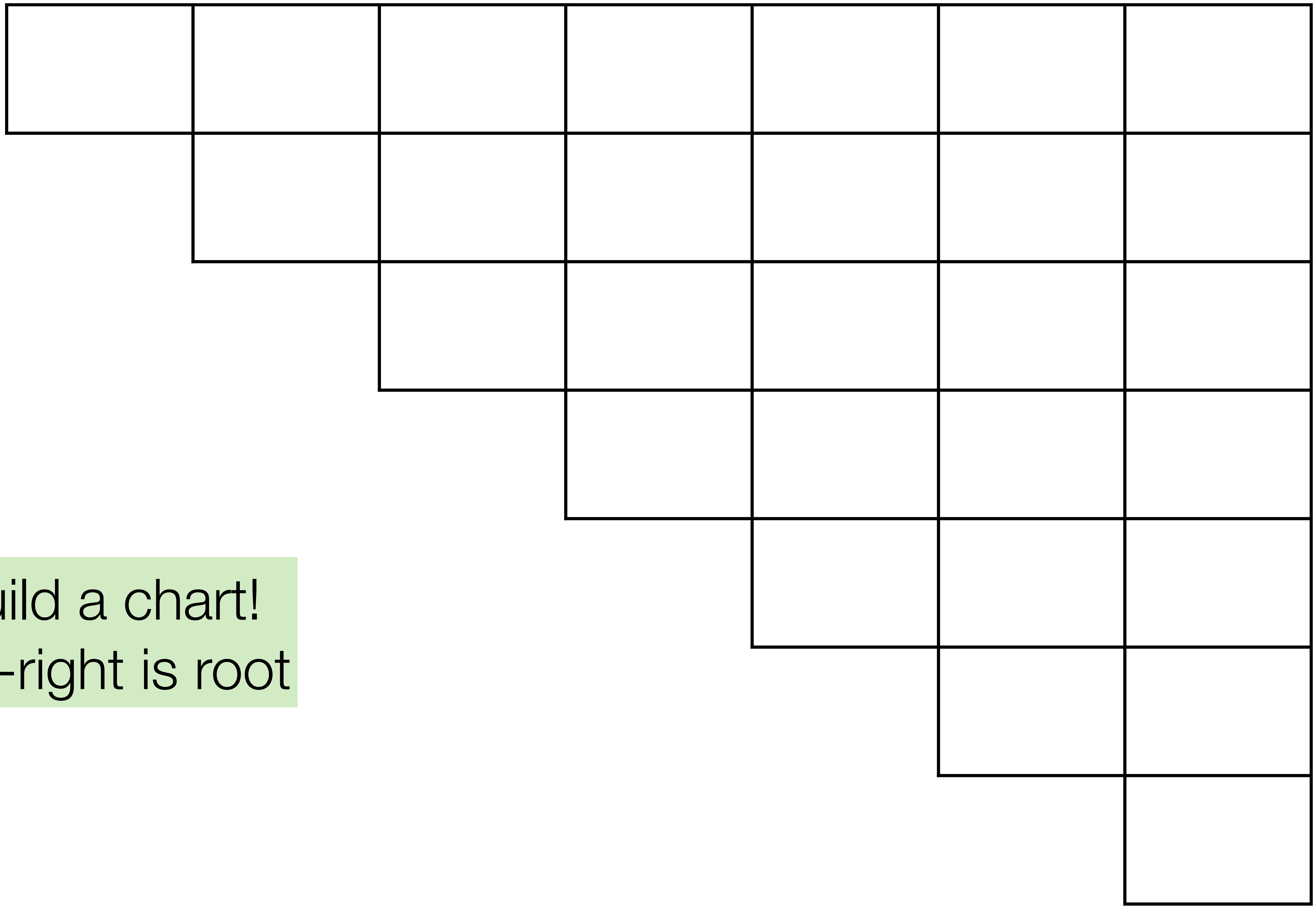
let’s say I have this CNF

- 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!

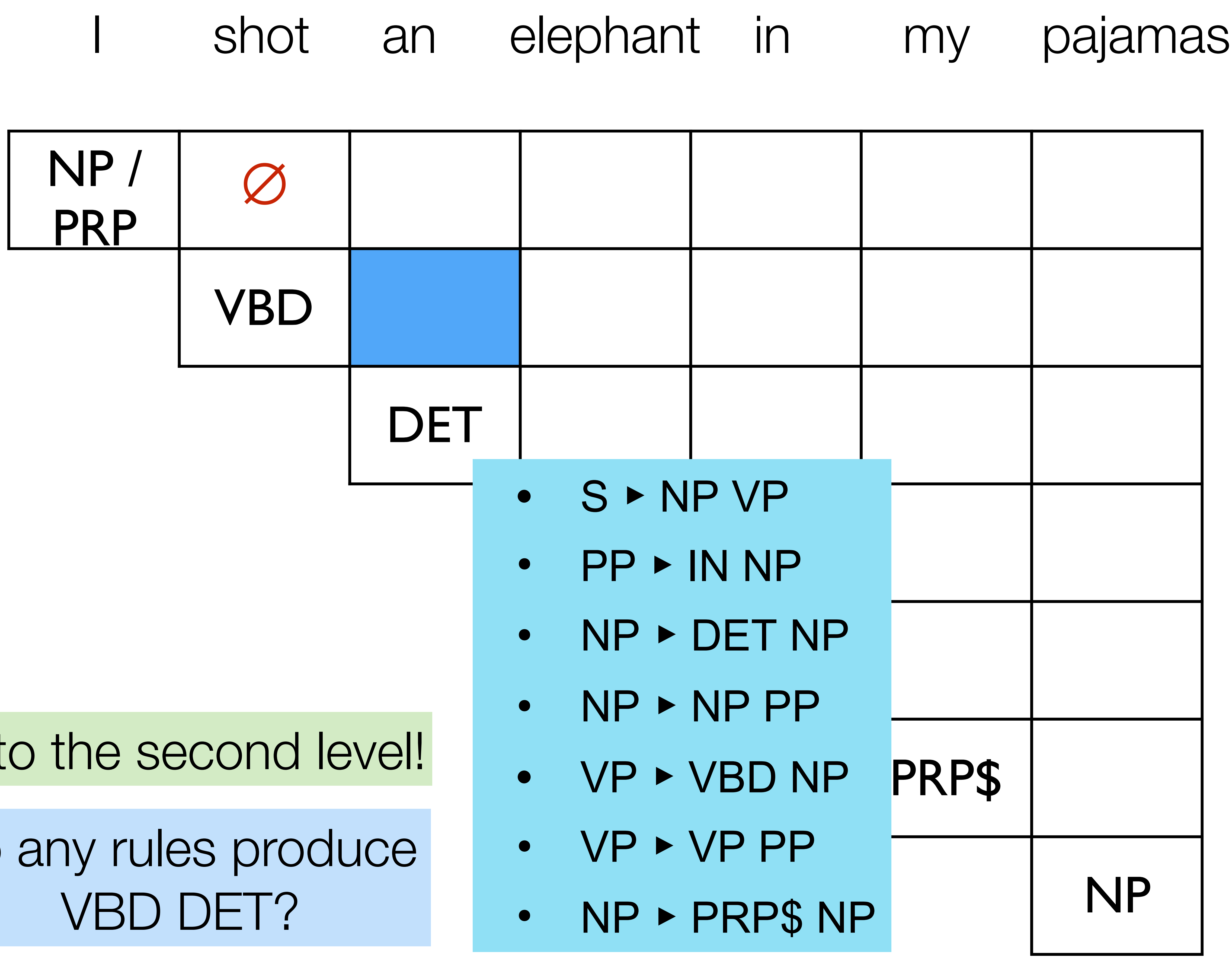
what does this cell span?

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!

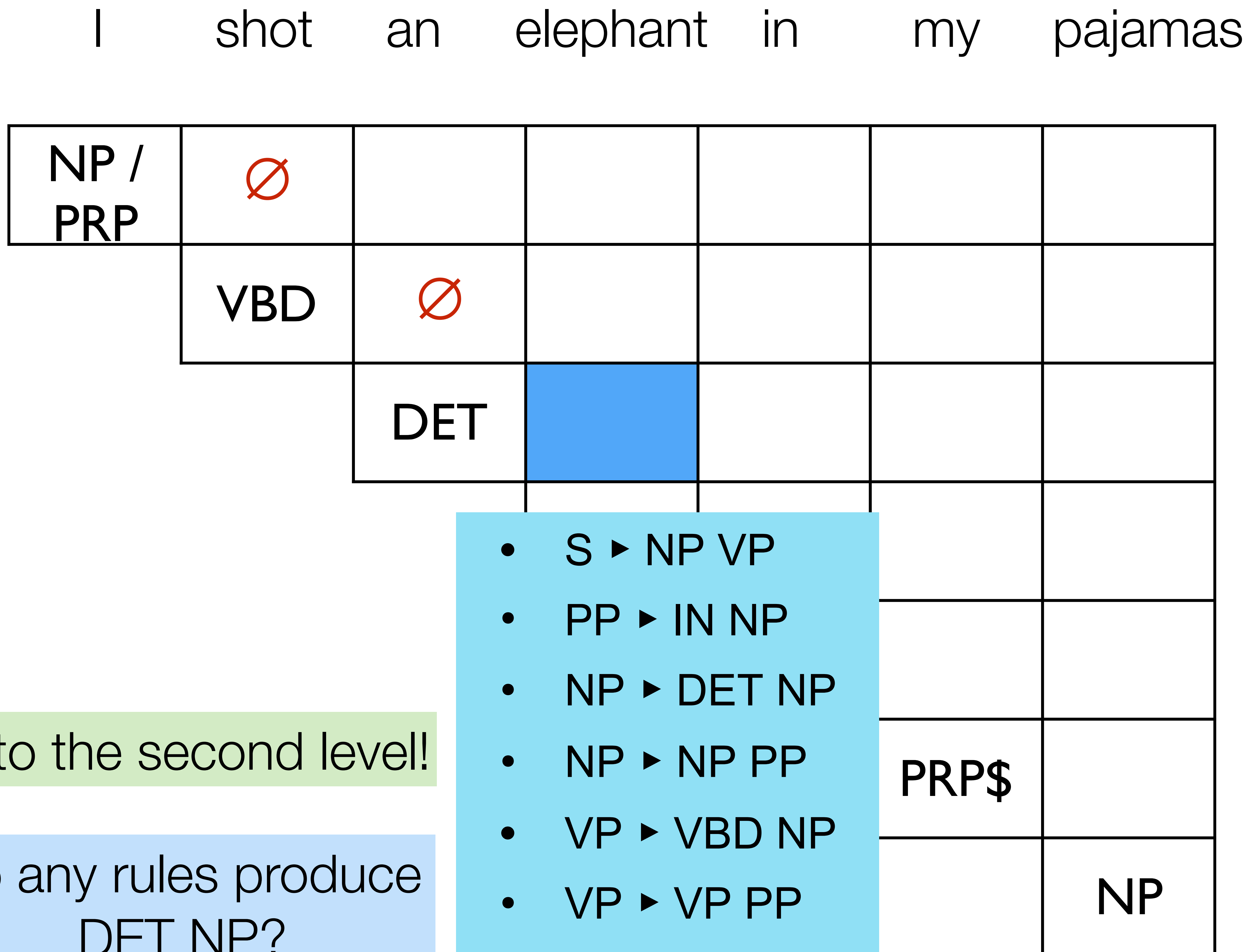
do any rules produce NP VBD or PRP VBD?



- 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

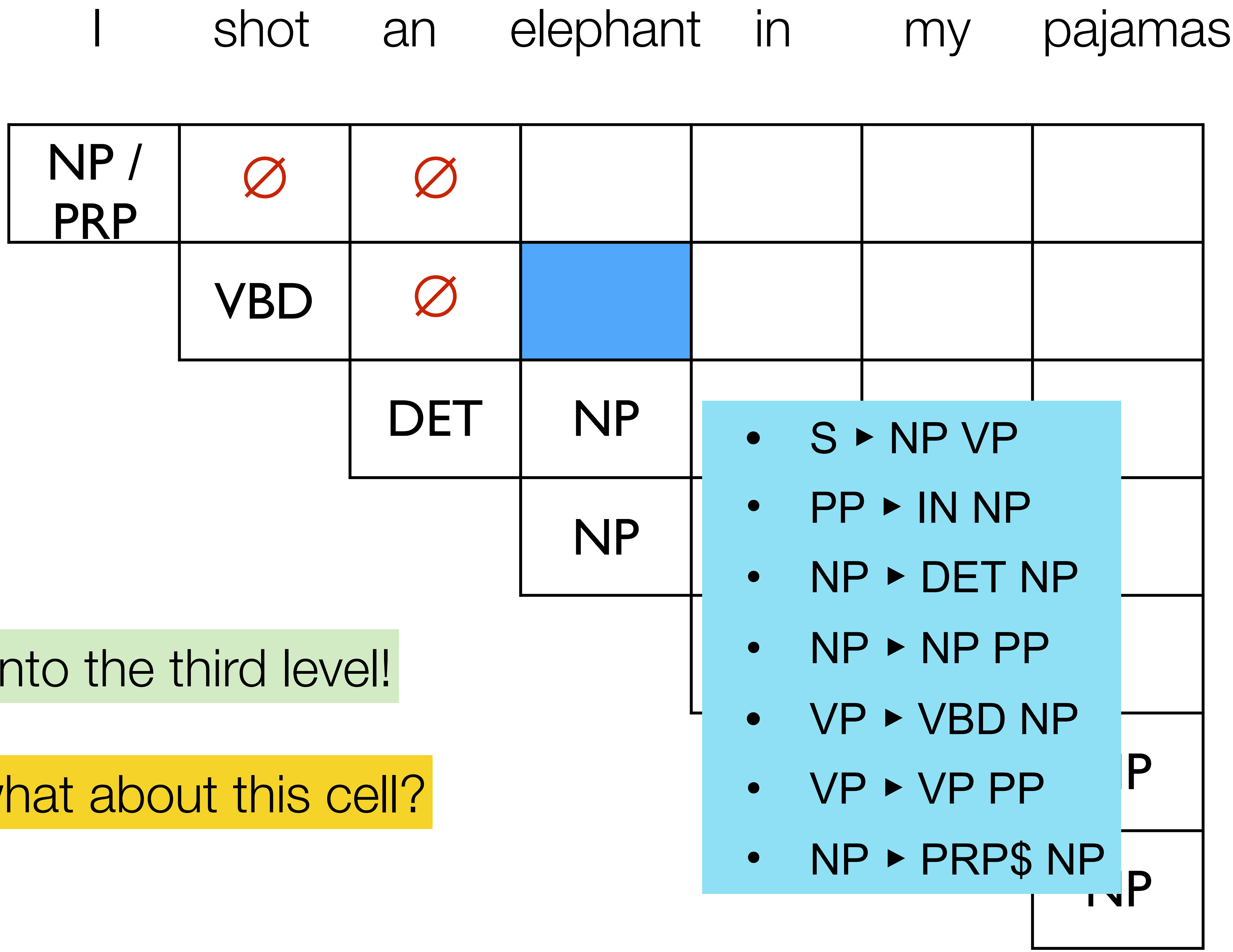
onto the third level!

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

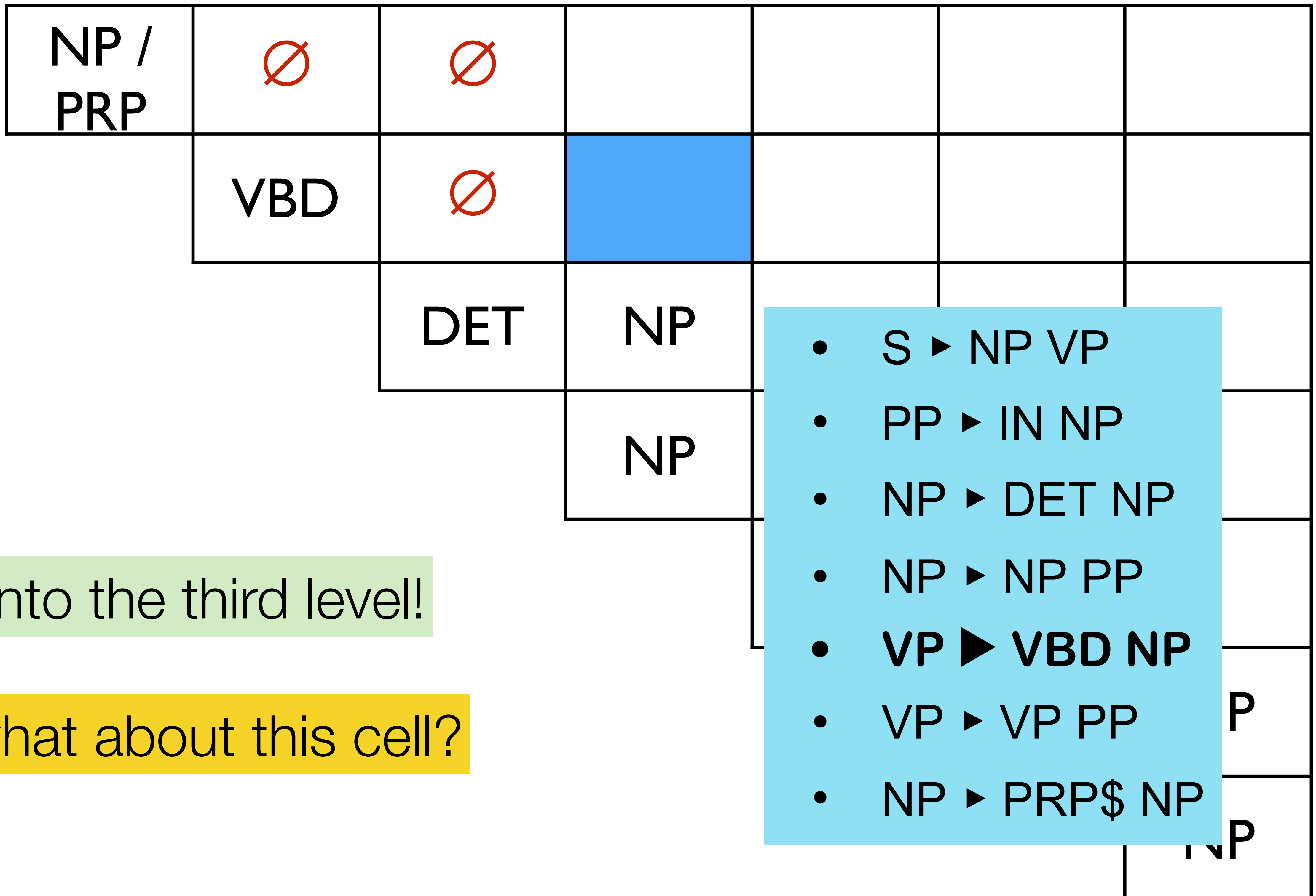


- S ▸ NP VP
- PP ▸ IN NP
- NP ▸ DET NP
- NP ▸ NP PP
- VP ▸ VBD NP
- VP ▸ VP PP
- NP ▸ PRP$ NP

onto the third level!

what about this cell?

I shot an elephant in my pajamas



- 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 ← from 1 to LENGTH(words) do
     for all {A | A → words[j] ∈ grammar}
          table[j−1, j] ← table[j−1, j] ∪ A
     for i ← from j−2 downto 0 do
        for k ← i+1 to j−1 do
          for all {A | A → BC ∈ grammar and B ∈ table[i,k] and C ∈ table[k, j]}
               table[i,j] ← table[i,j] ∪ 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)

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

fill in first level (words) with possible derivations

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)

- 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

how do we compute this cell's probability?

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

- 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

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)

- 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

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)

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

$p(NP_1) = ?$
$p(NP_2) = ?$

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)

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

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)

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

$p(VP_1) = ?$
$p(VP_2) = ?$

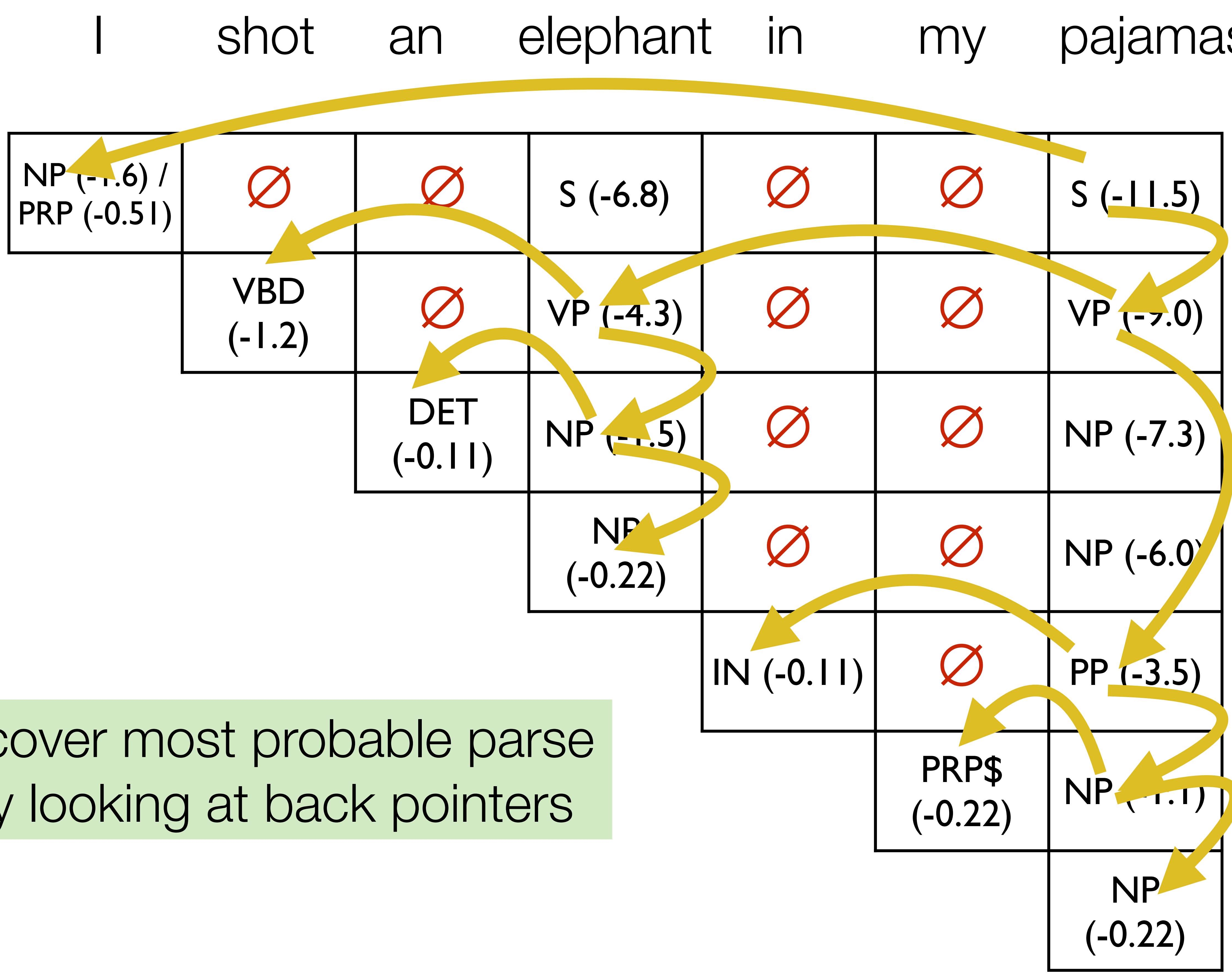
I	shot	an	elephant	in	my	pajamas
NP (-1.6) / PRP (-0.51)	∅	∅	S (-6.8)	∅	∅	
	VBD (-1.2)	∅	VP (-4.3)	∅	∅	VP_1 (-10.1) / VP_2 (-9.0)
		DET (-0.11)	NP (-1.5)	∅	∅	NP (-7.3)
			NP (-0.22)	∅	∅	NP (-6.0)
				IN (-0.11)	∅	PP (-3.5)
					PRP$ (-0.22)	NP (-1.1)
						NP (-0.22)

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

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)

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



recover most probable parse
by looking at back pointers

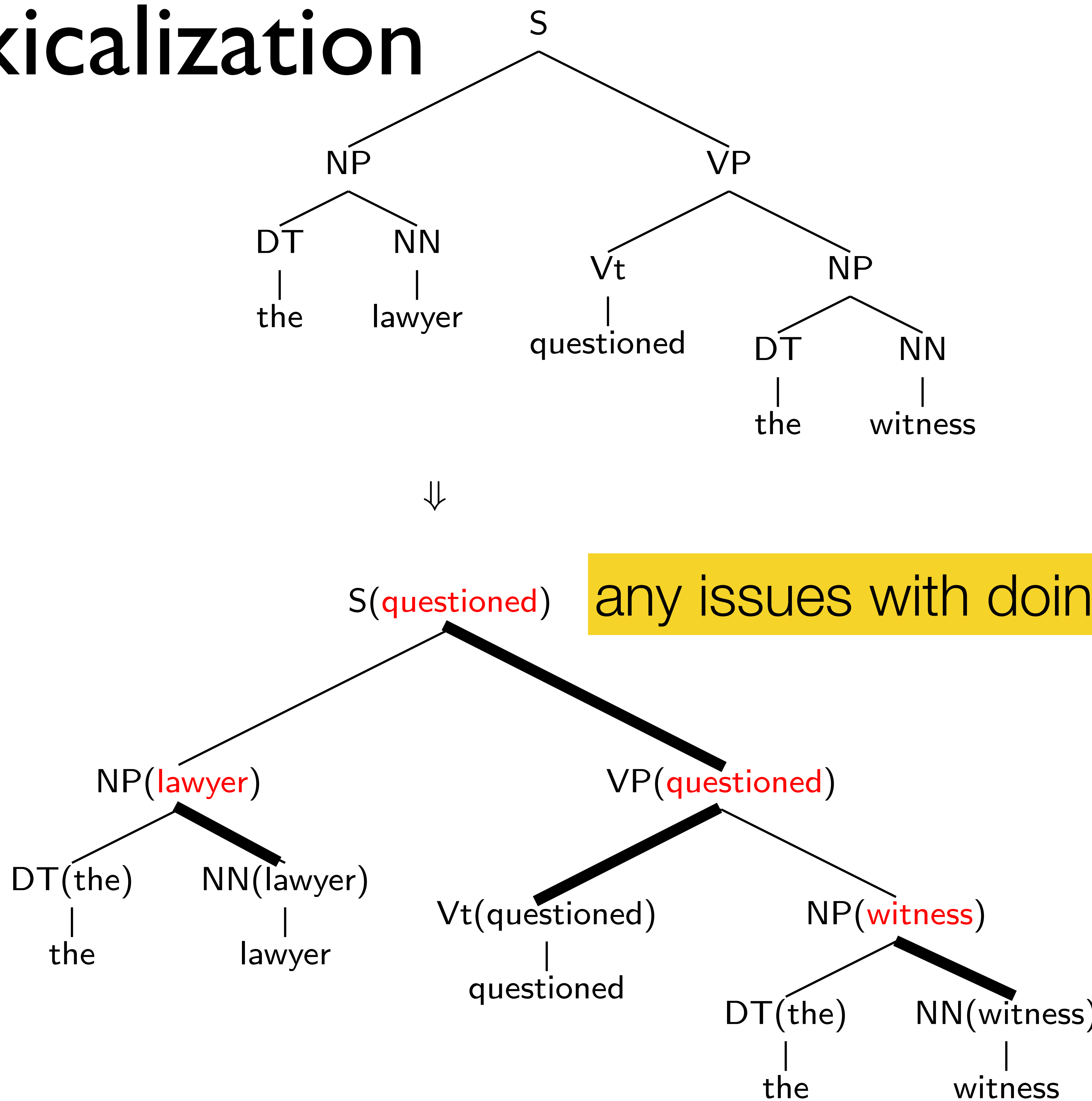
issues w/ PCFGs

- 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!