Computational Linguistics: Parsing

Oct 18, 2013

Dynamic Programming Solution

- Recall: convert the grammar into Chomsky Normal Form (CNF)
  - `A` → `B C`
  - `A` → `w`

Original Grammar

- S → NP VP
- PP → Prep NP
- NP → an Noun
- NP → NP PP
- VP → Verb NP
- VP → Verb
- VP → VP PP
- NP → my Noun
- NP → Noun
- NP → Pronoun
- Noun → elephant
- Pronoun → I
- Prep → in
- Noun → pajamas
- Noun → shot
- Verb → shot

CNF Grammar

- S → NP VP
- PP → Prep NP
- NP → Det Noun
- Det → an
- NP → NP PP
- VP → Verb NP
- VP → shot
- VP → VP PP
- Det → my
- NP → elephant
- NP → pajamas
- NP → shot
- NP → I
- Noun → elephant
- Pronoun → I
- Prep → in
- Noun → pajamas
- Noun → shot
- Verb → shot

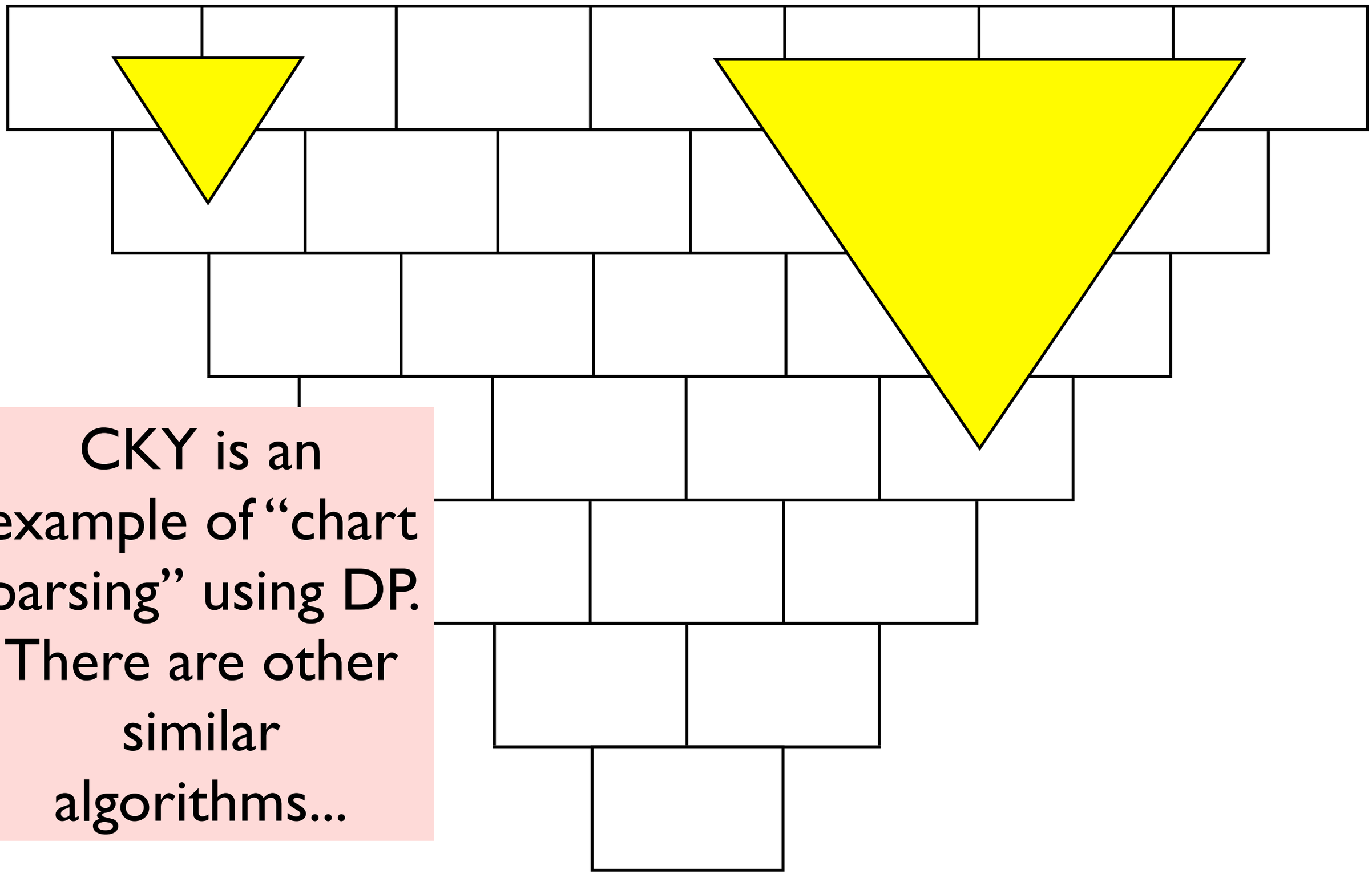
Recognition:

Is the sentence valid?

i.e., Is it parseable by the grammar?

CKY (or CYK) Algorithm

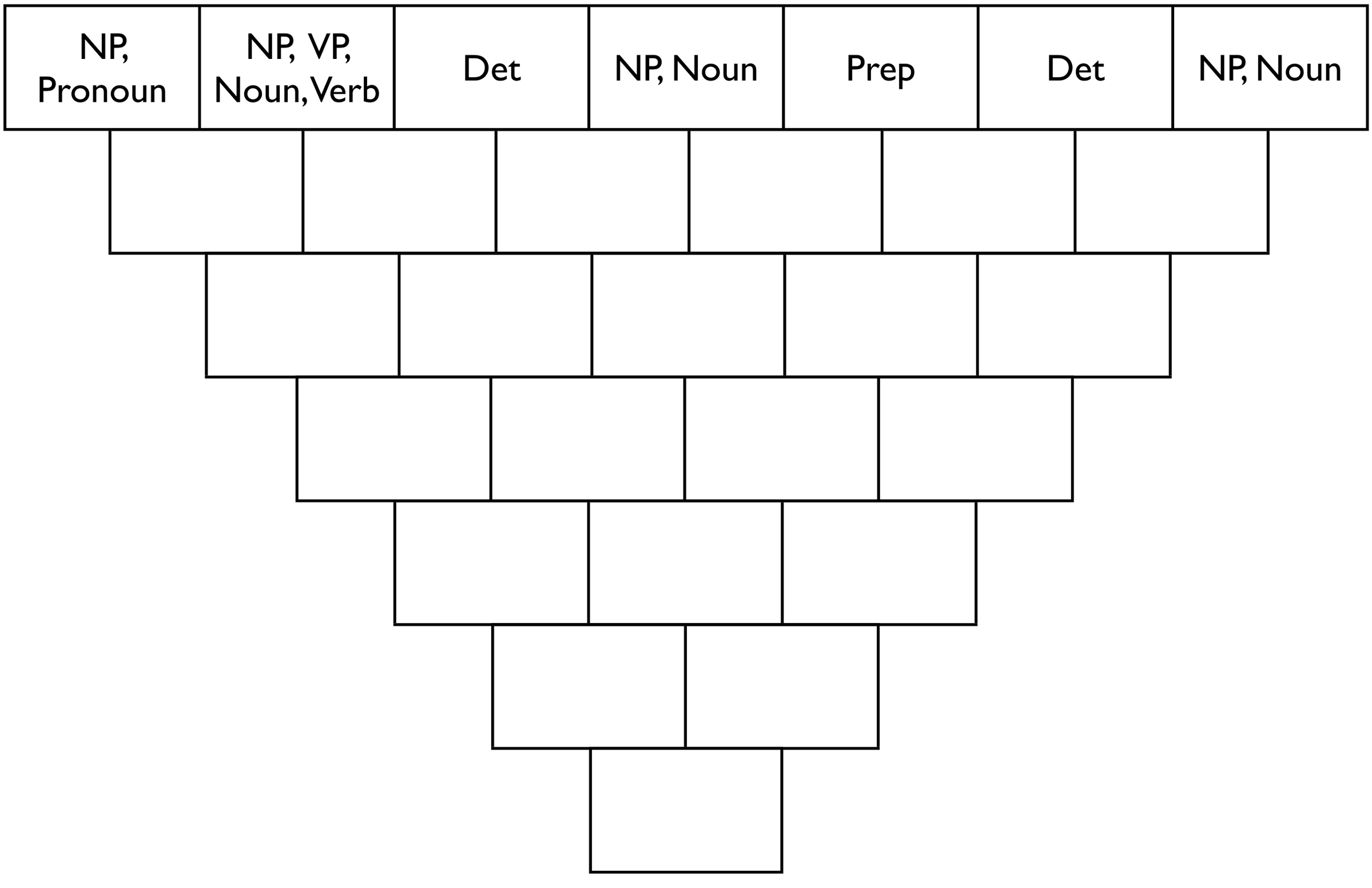
I shot an elephant in my pajamas



CKY is an example of "chart parsing" using DP. There are other similar algorithms...

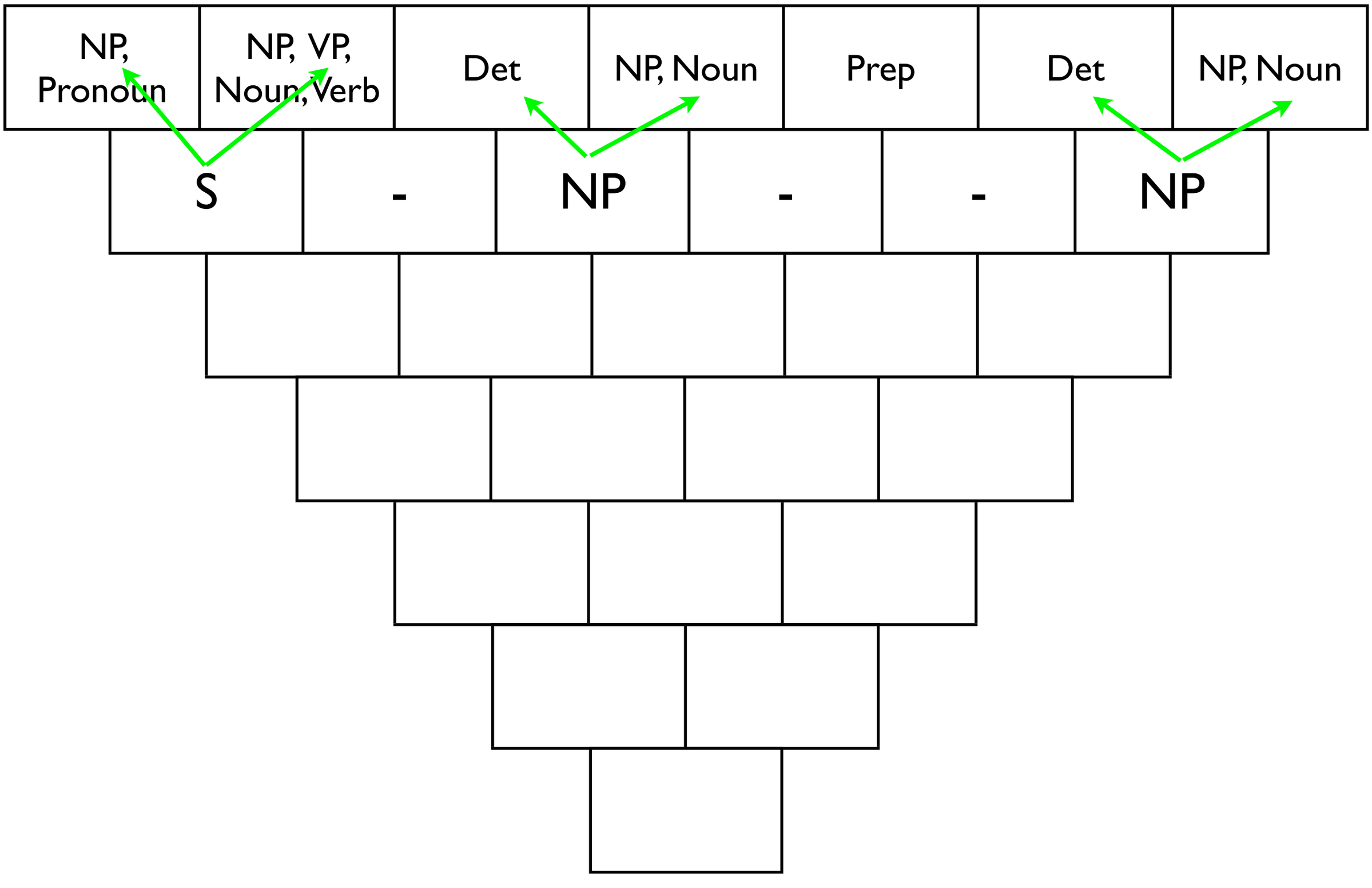
CKY (or CYK) Algorithm

I shot an elephant in my pajamas



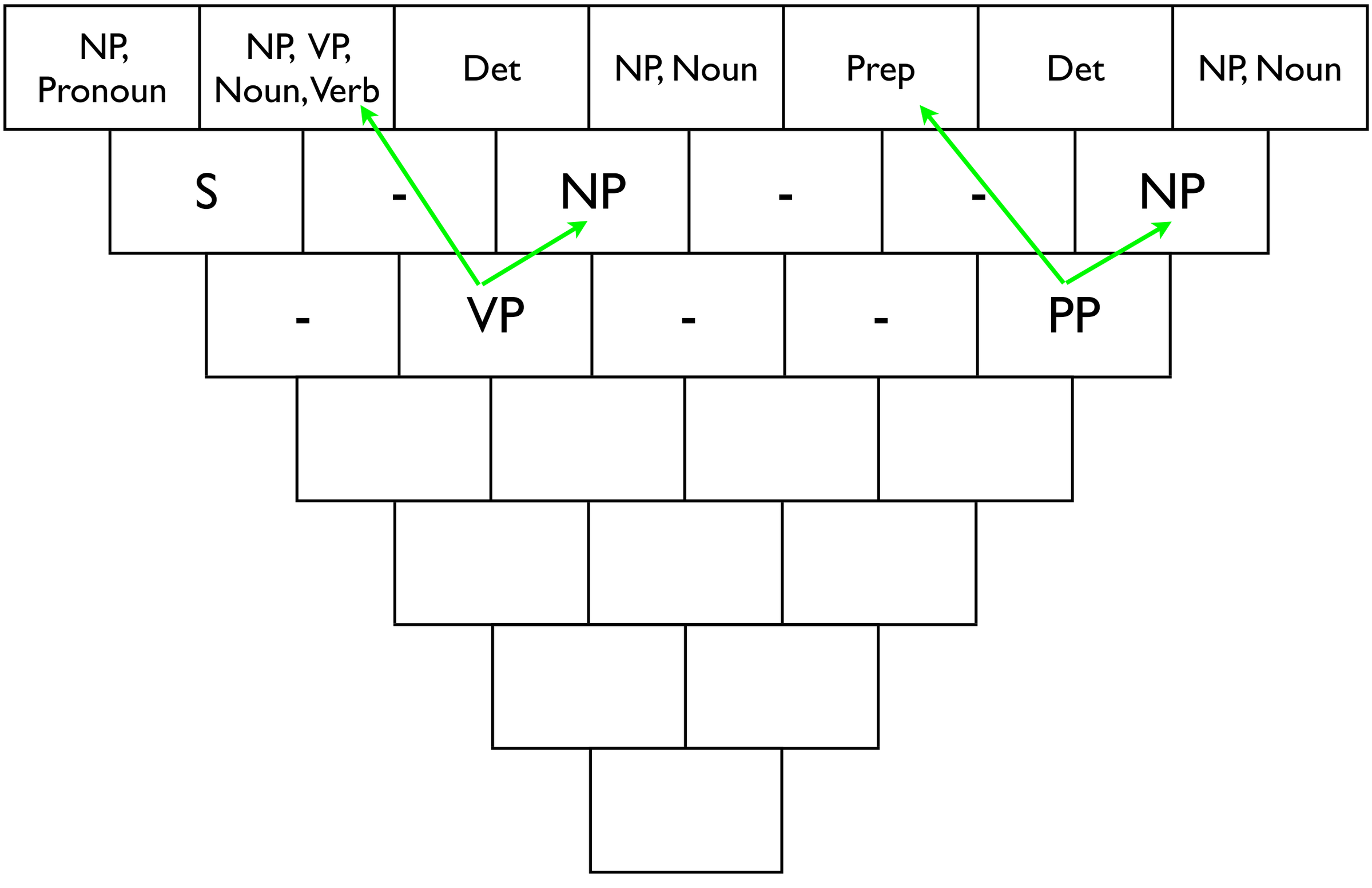
CKY (or CYK) Algorithm

I shot an elephant in my pajamas



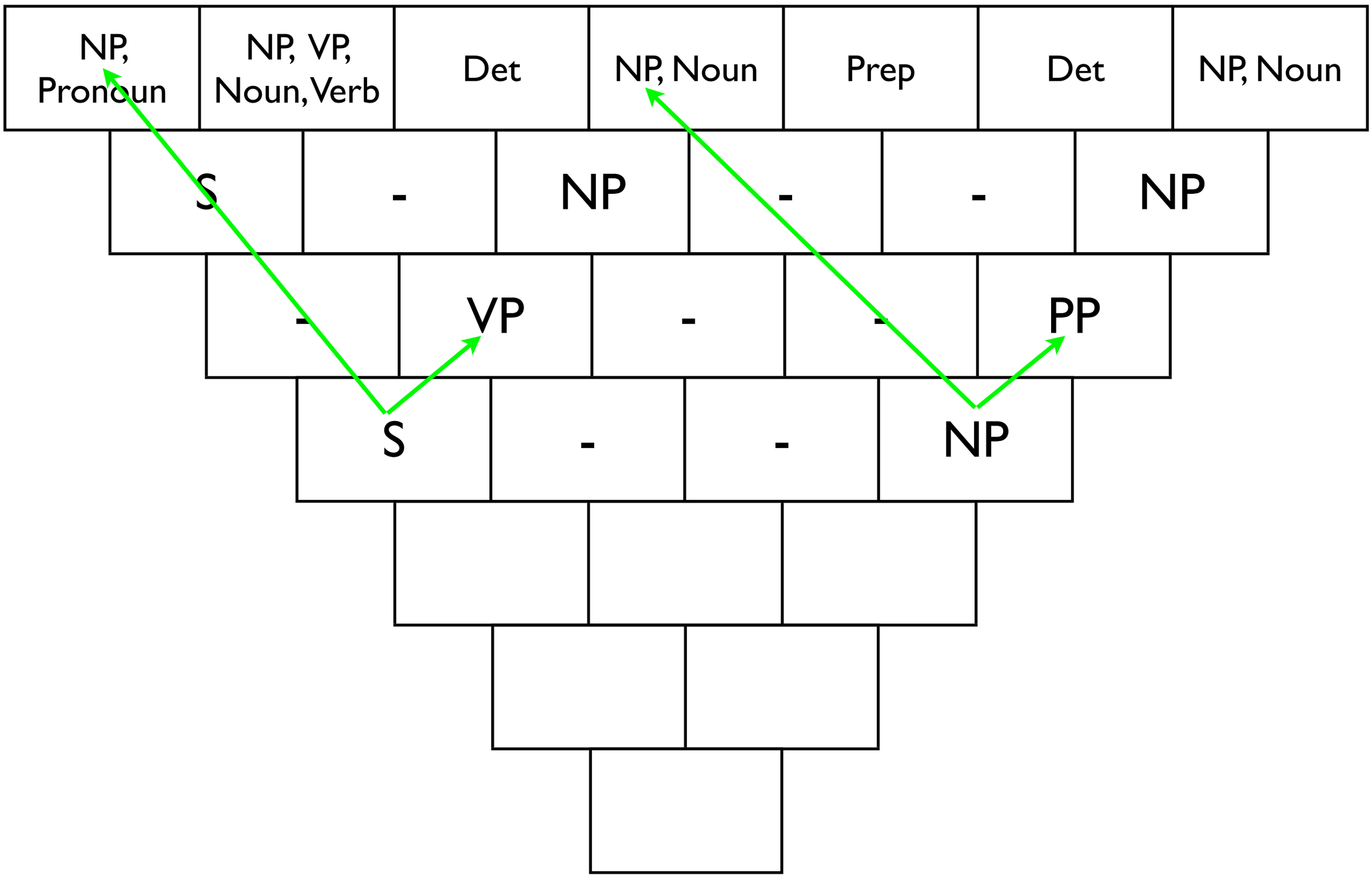
CKY (or CYK) Algorithm

I shot an elephant in my pajamas



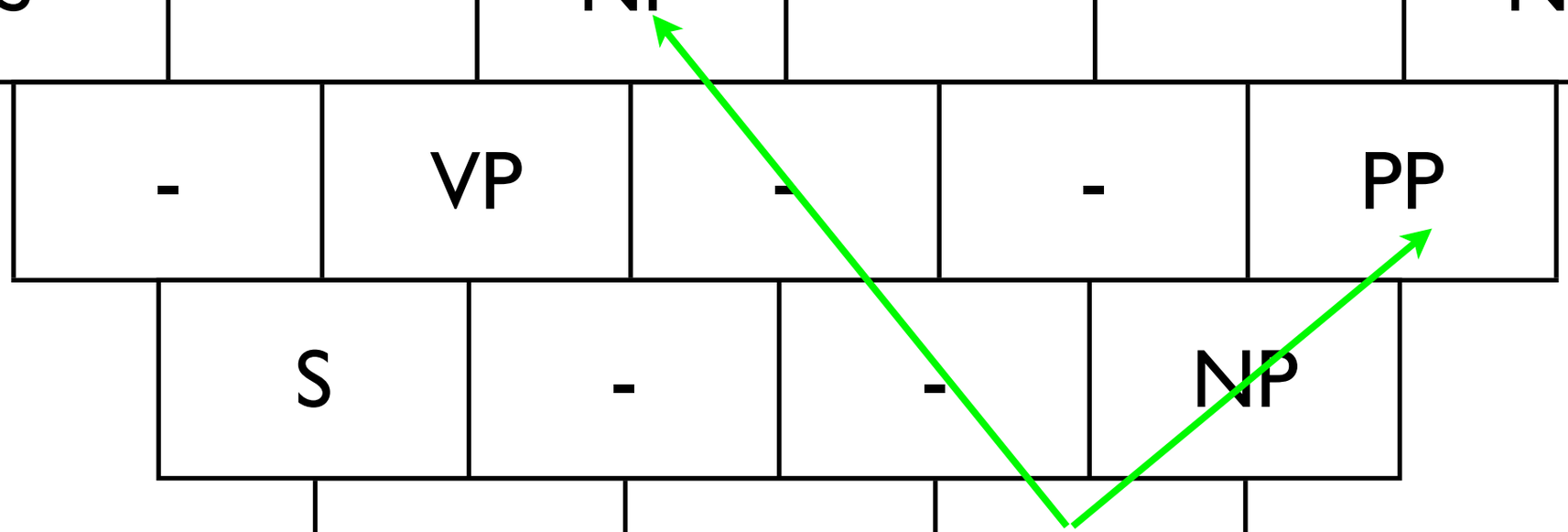
CKY (or CYK) Algorithm

I shot an elephant in my pajamas



CKY (or CYK) Algorithm

I	shot	an	elephant	in	my	pajamas
NP, Pronoun	NP, VP, Noun, Verb	Det	NP, Noun	Prep	Det	NP, Noun
S	-	NP	-	-	NP	
-	VP	-	-	PP		
S	-	-	NP			
-	-	NP				



CKY (or CYK) Algorithm

I	shot	an	elephant	in	my	pajamas
NP, Pronoun	NP, VP, Noun, Verb	Det	NP, Noun	Prep	Det	NP, Noun
S	-	NP	-	-	NP	
-	VP	-	-	PP		
S	-	-	NP			
-	-	NP				
-	VP,VP					

CKY (or CYK) Algorithm

I	shot	an	elephant	in	my	pajamas
NP, Pronoun	NP, VP, Noun, Verb	Det	NP, Noun	Prep	Det	NP, Noun
S	-	NP	-	-	NP	
-	VP	-	-	PP		
S	-	-	NP			
-	-	NP				
-	VP,VP					
S						

Is sentence grammatical
= is the root an S?

Time and Space Complexity of CKY?

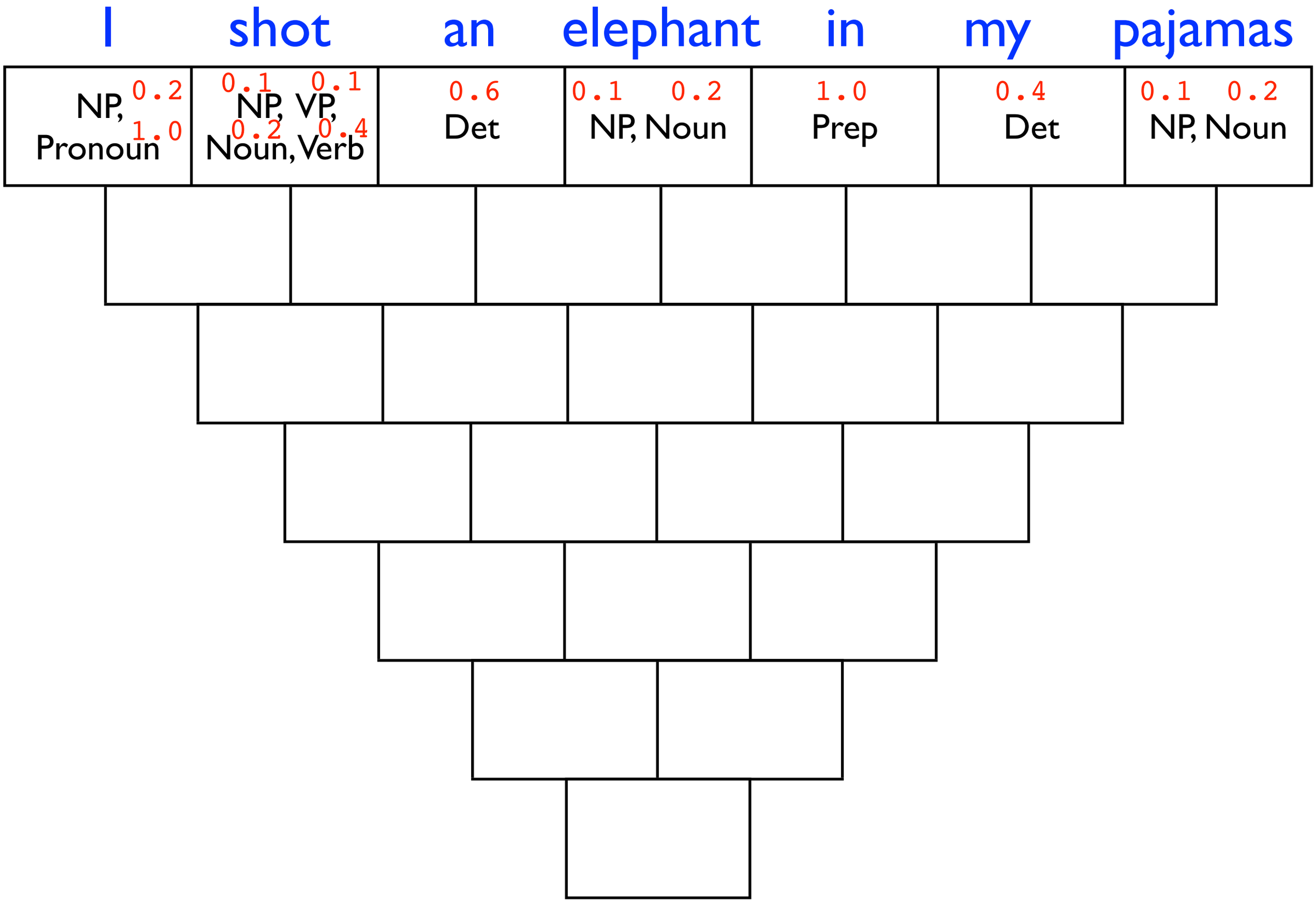
Decoding:

What is the best parse of the sentence?

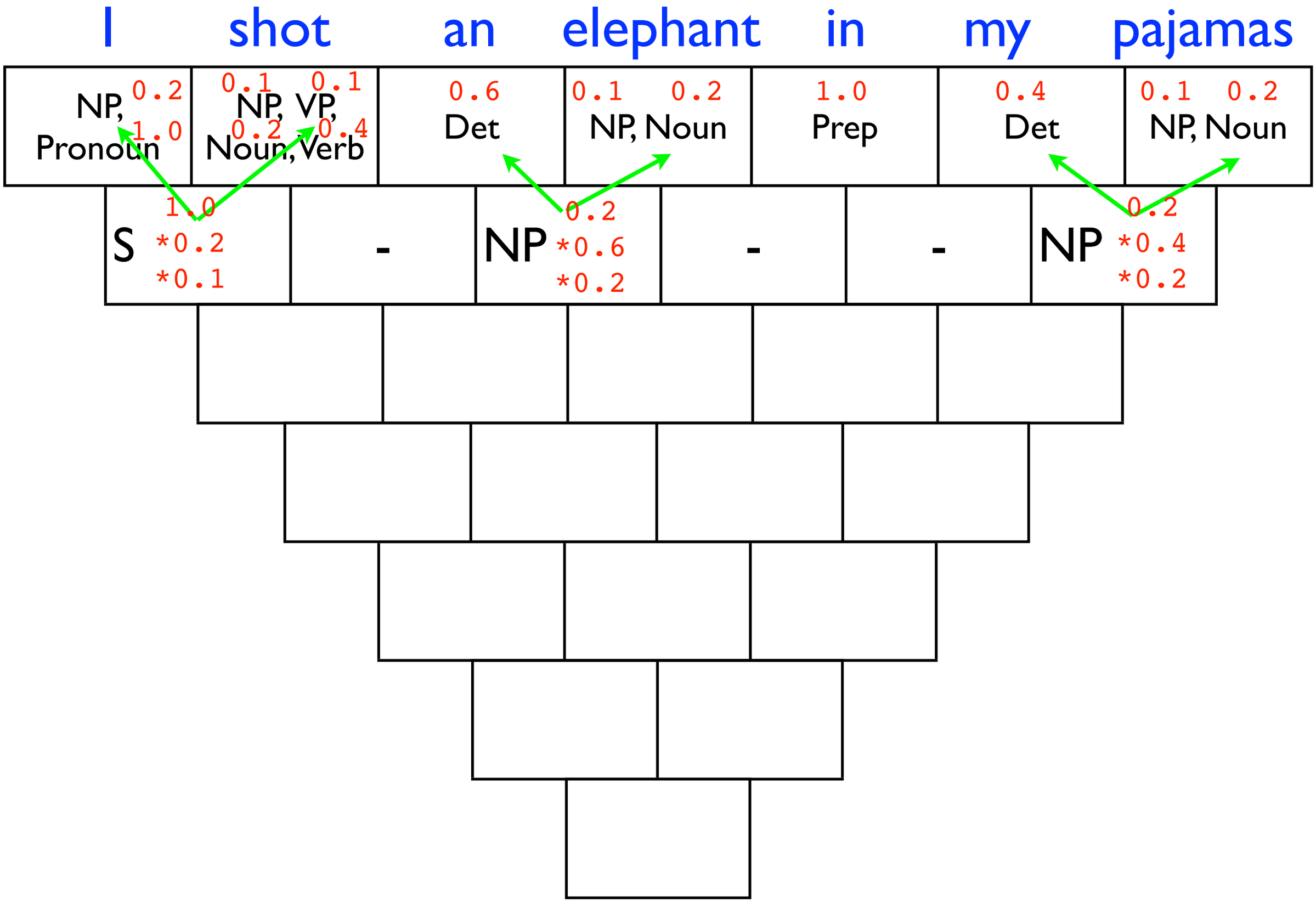
Probabilistic Grammar

- S → NP VP 1.0
- PP → Prep NP 1.0
- NP → Det Noun 0.2
- NP → NP PP 0.3
- NP → elephant 0.1
- NP → pajamas 0.1
- NP → shot 0.1
- NP → I 0.2
- Pronoun → I 1.0
- Det → an 0.6
- Det → my 0.4
- VP → Verb NP 0.5
- VP → shot 0.1
- VP → VP PP 0.4
- Prep → in 1.0
- Noun → elephant 0.2
- Noun → pajamas 0.2
- Noun → shot 0.2
- Verb → shot 0.4

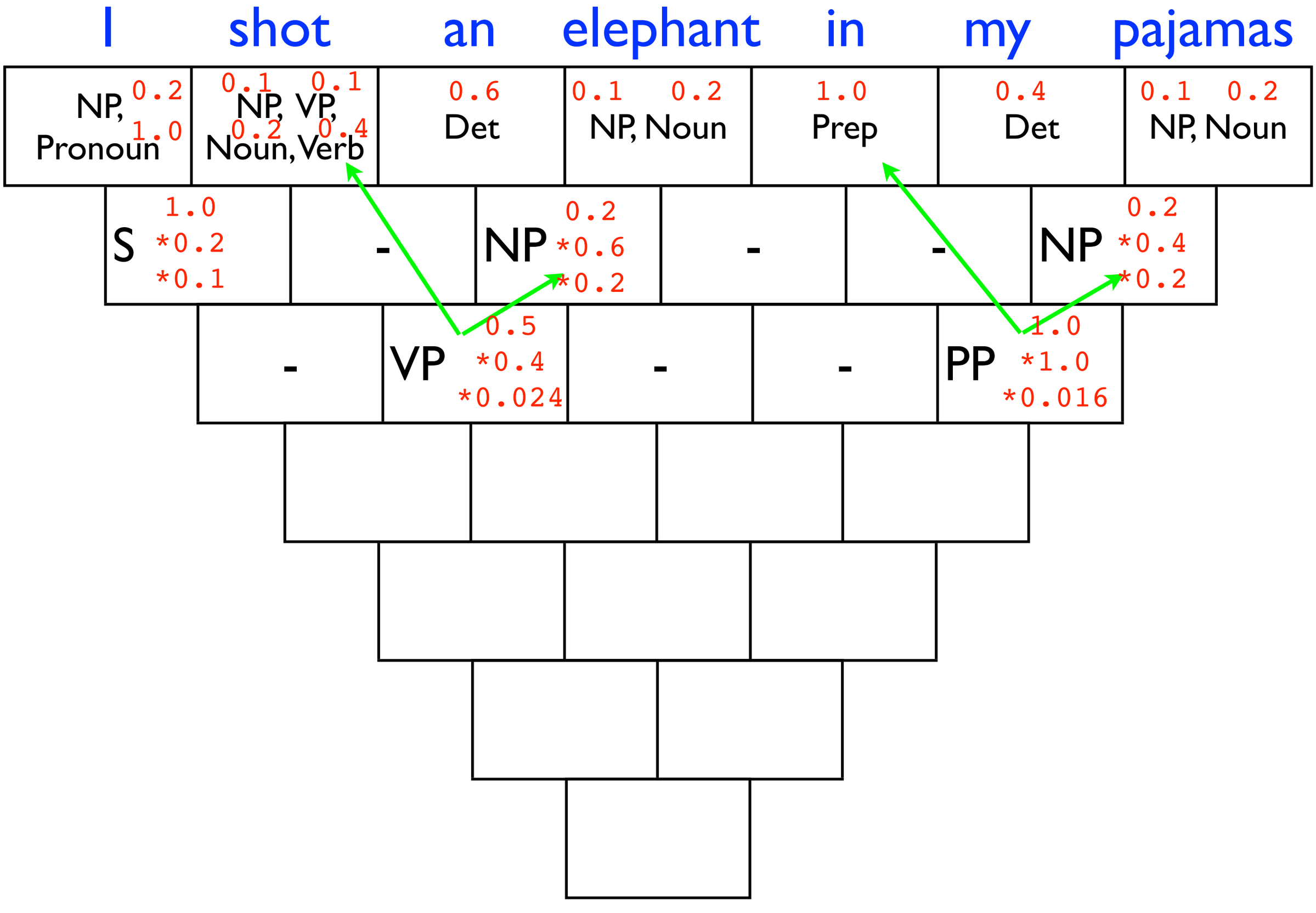
CKY (or CYK) Algorithm



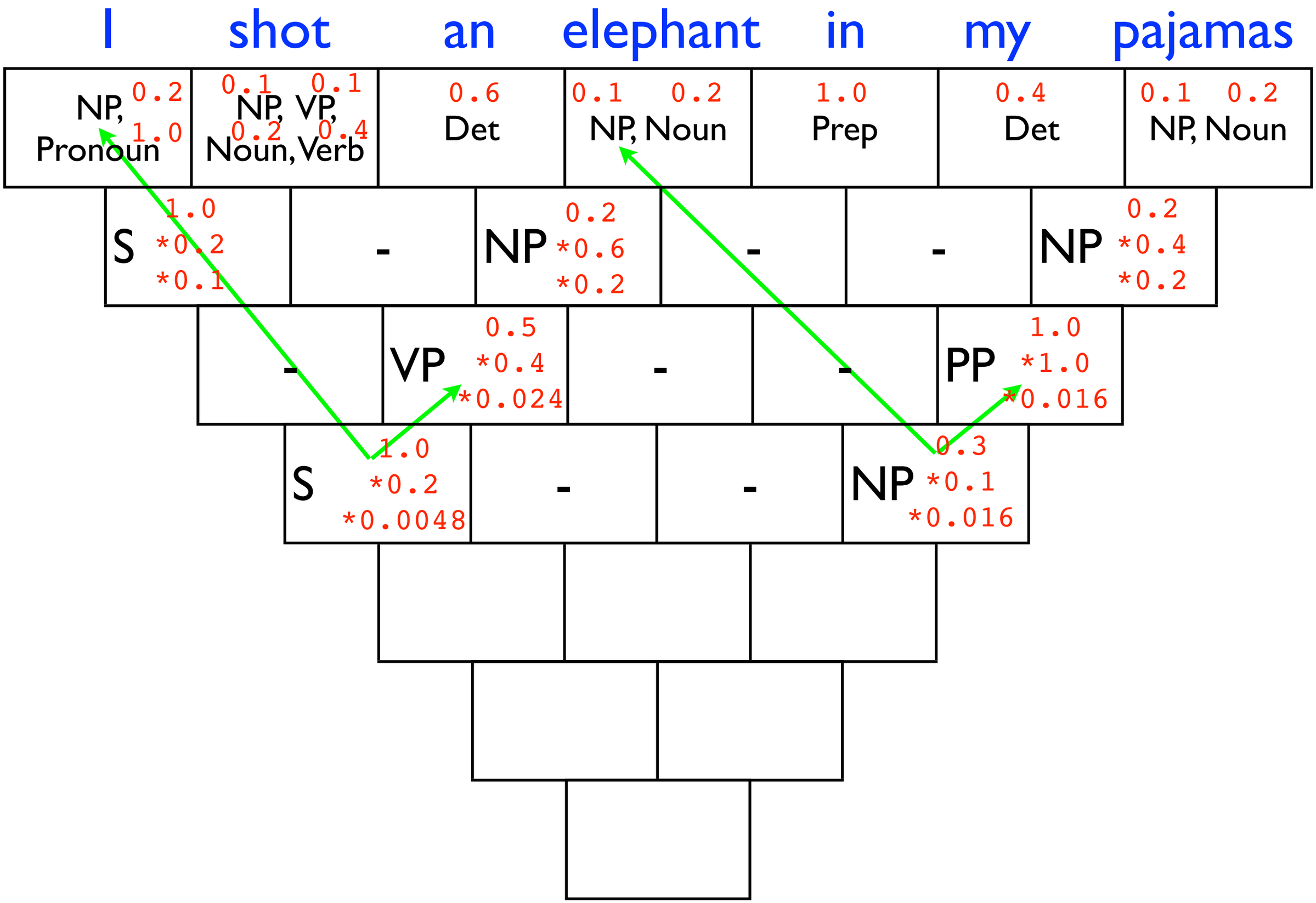
CKY (or CYK) Algorithm



CKY (or CYK) Algorithm



CKY (or CYK) Algorithm



CKY (or CYK) Algorithm

I	shot	an	elephant	in	my	pajamas
NP, Pronoun 0.2 1.0	0.1 0.1 NP, VP, Noun, Verb 0.2 0.4	0.6 Det	0.1 0.2 NP, Noun	1.0 Prep	0.4 Det	0.1 0.2 NP, Noun
S 1.0 *0.2 *0.1	-	NP 0.2 *0.6 *0.2	-	-	NP 0.2 *0.4 *0.2	
-	VP 0.5 *0.4 *0.024	-	-	PP 1.0 *1.0 *0.016		
S 1.0 *0.2 *0.0048	-	-	NP 0.3 *0.1 *0.016			
-	-	NP 0.3 *0.024 *0.016				

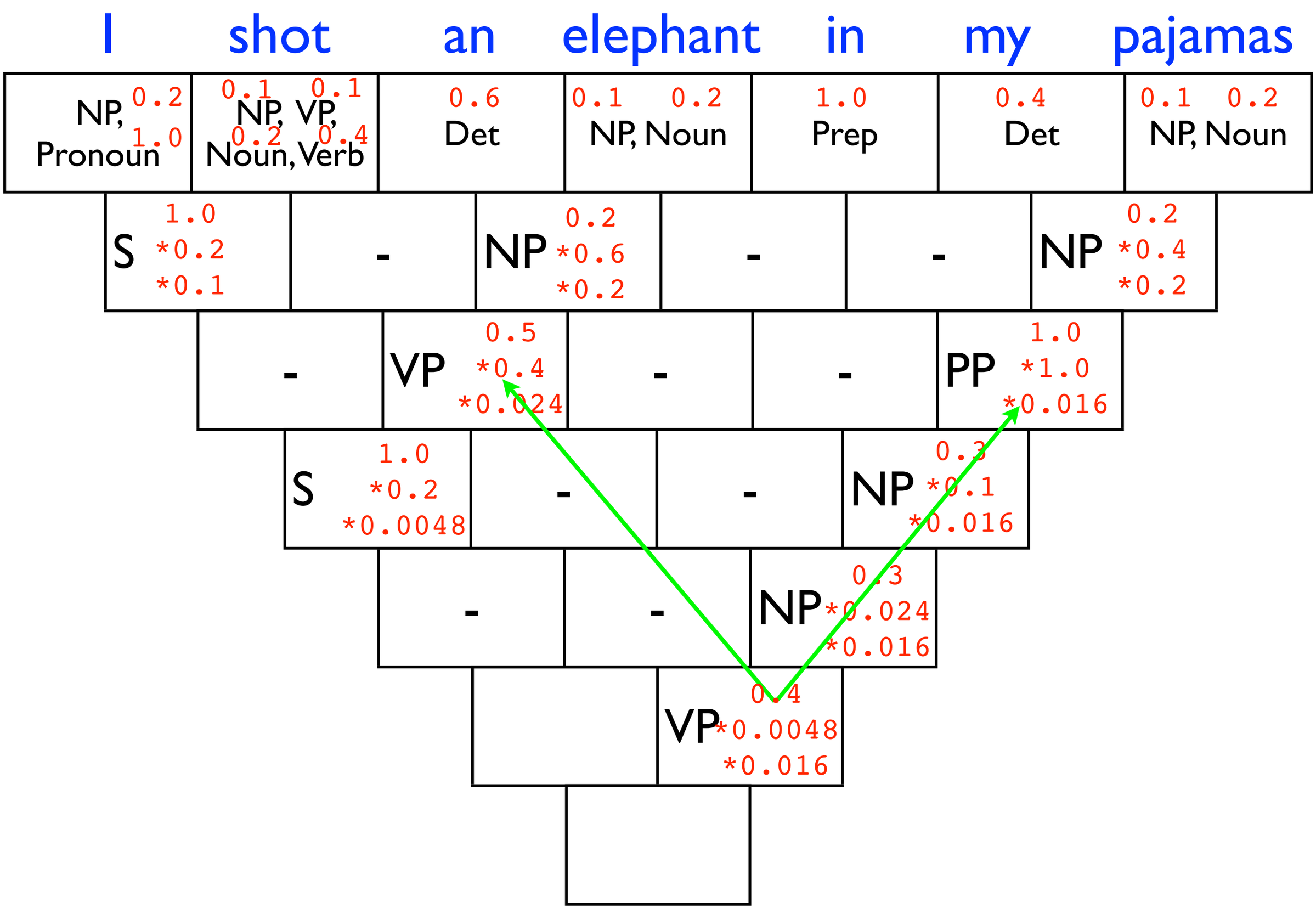
CKY (or CYK) Algorithm

I	shot	an	elephant	in	my	pajamas
NP, Pronoun 0.2 1.0	0.1 0.1 NP, VP, Noun, Verb 0.2 0.4	0.6 Det	0.1 0.2 NP, Noun	1.0 Prep	0.4 Det	0.1 0.2 NP, Noun
S 1.0 *0.2 *0.1	-	NP 0.2 *0.6 *0.2	-	-	NP 0.2 *0.4 *0.2	
-	VP 0.5 *0.4 *0.024	-	-	PP 1.0 *1.0 *0.016		
S 1.0 *0.2 *0.0048	-	-	NP 0.3 *0.1 *0.016			
-	-	NP 0.3 *0.024 *0.016				
	VP					

First:
0.5
*0.4
*0.0001152

Second:
0.4
*0.0048
*0.016

CKY (or CYK) Algorithm



CKY (or CYK) Algorithm

I	shot	an	elephant	in	my	pajamas
NP, Pronoun 0.2 1.0	NP, VP, Noun, Verb 0.1 0.1 0.2 0.4	Det 0.6	NP, Noun 0.1 0.2	Prep 1.0	Det 0.4	NP, Noun 0.1 0.2
S 1.0 *0.2 *0.1	-	NP 0.2 *0.6 *0.2	-	-	NP 0.2 *0.4 *0.2	
-	VP 0.5 *0.4 *0.024	-	-	PP 1.0 *1.0 *0.016		
S 1.0 *0.2 *0.0048	-	-	NP 0.3 *0.1 *0.016			
-	-	NP 0.3 *0.024 *0.016				
	VP 0.4 *0.0048 *0.016					
S 1.0 *0.2 *0.000031						

Also called the "Viterbi Parse"

Implementation Details

- Assume we have a parsed corpus
  - Very few exist: major one is the Penn Treebank (Wall Street Journal)

```
( (S (NP-SBJ (NP (NNP Pierre) (NNP Vinken))
             (, ,)
             (ADJP (NML (CD 61) (NNS years))
              (JJ old))
             (, ,))
       (VP (MD will)
     (VP (VB join)
         (NP (DT the) (NN board))
         (PP-CLR (IN as)
            (NP (DT a) (JJ nonexecutive) (NN director)))
         (NP-TMP (NNP Nov.) (CD 29))))
       (. .)))
```

Implementation Details

- Assume we have a parsed corpus
  - Very few exist: major one is the Penn Treebank (Wall Street Journal)
- Count rules in the trees (just like counting n-grams!) and normalize to get CFG probabilities
- Typically do some smoothing at the rules that generate terminals

What if we don't have an annotated corpus?

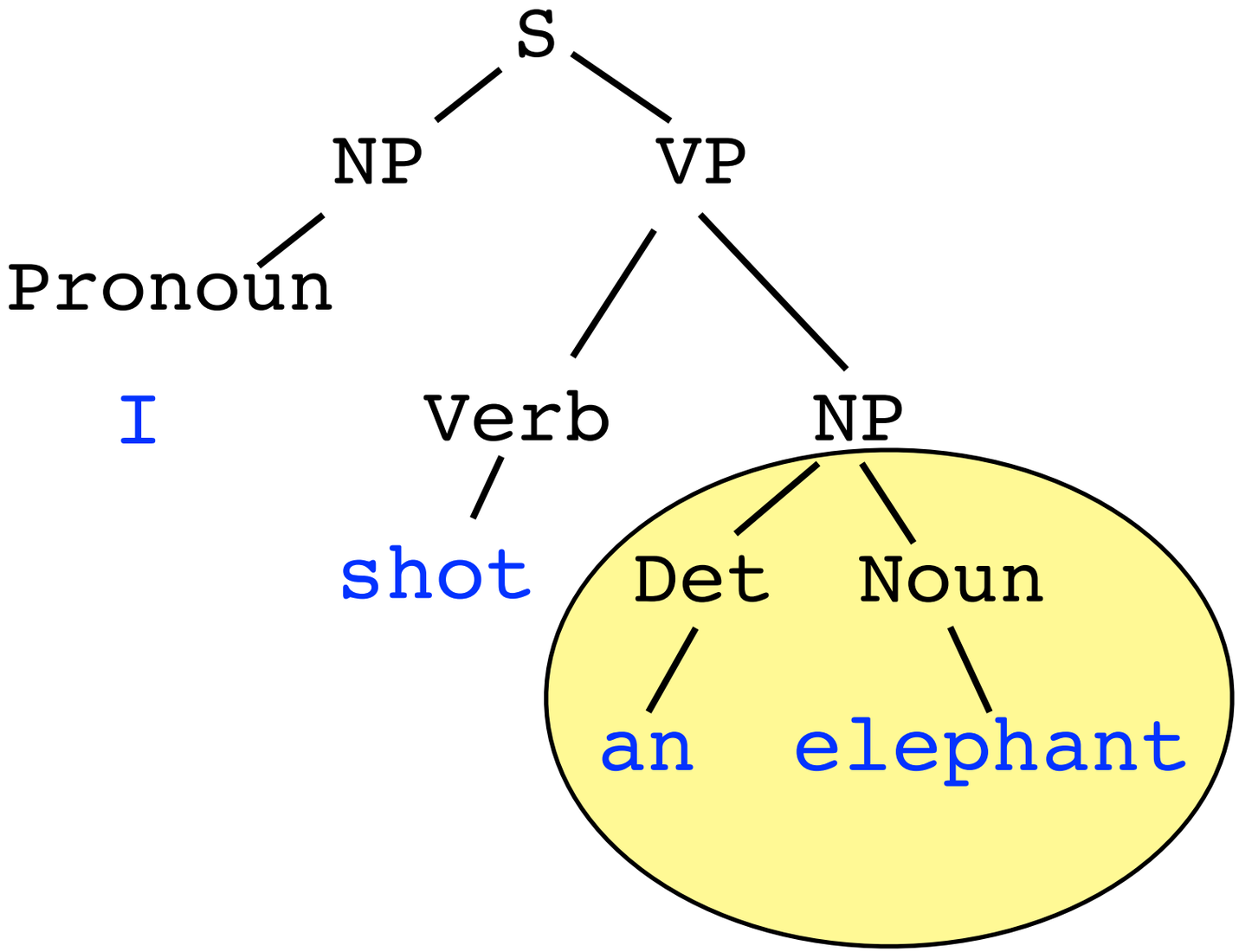
- Expectation Maximization to estimate CFG probabilities

- Inside Probabilities: Bottom-up CKY, but find marginal probability at each non-terminal rather than the max

- Outside Probabilities: Use top-down CKY (expanding from root S to sentence)

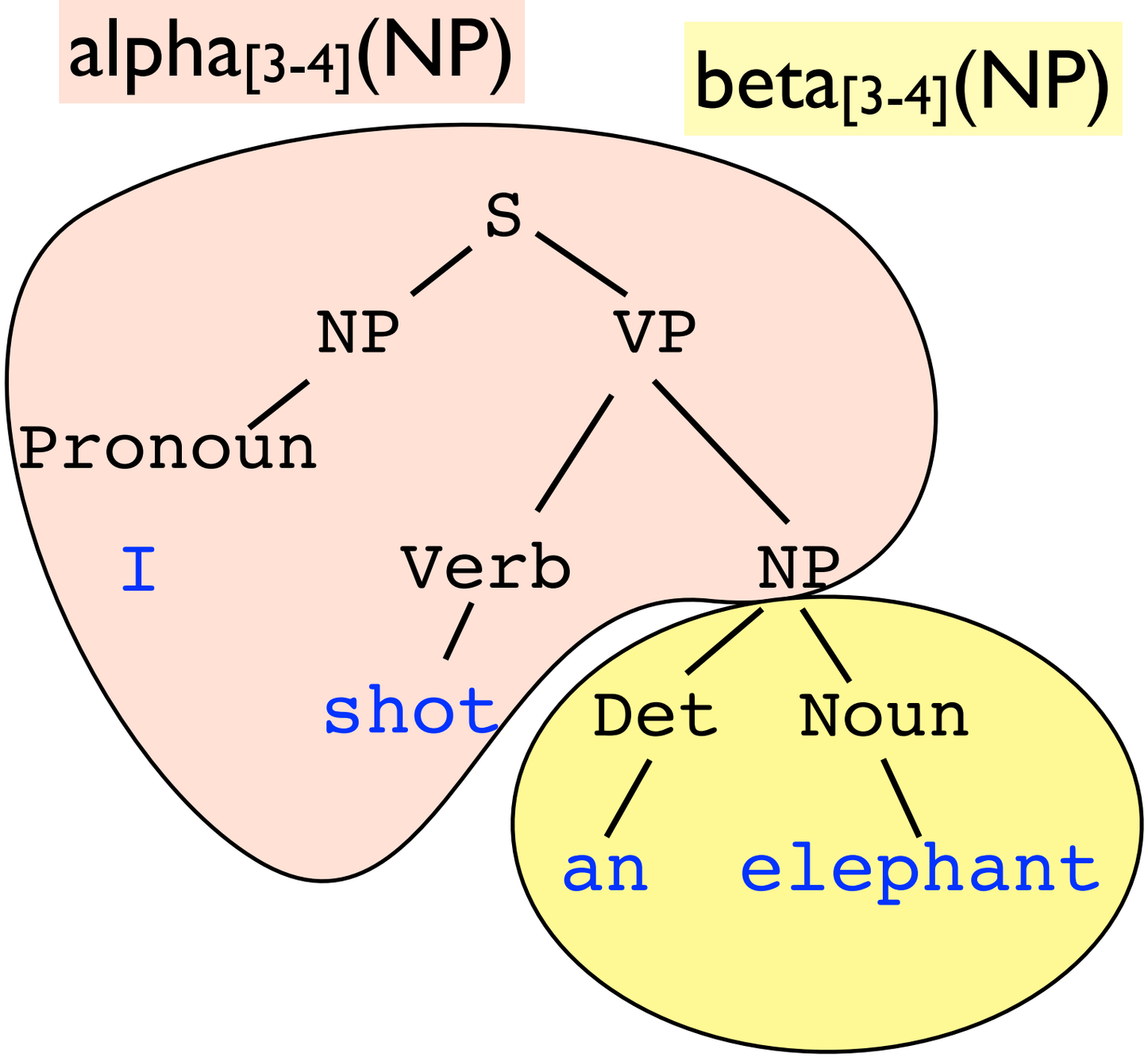
What if we don't have an annotated corpus?

$beta_{[3-4]}(NP)$



- Inside probabilities: Analogous to backward probabilities in finite state machines

What if we don't have an annotated corpus?



- Inside probabilities: Analogous to backward probabilities in finite state machines
- Outside probabilities: Analogous to forward probabilities

Some unfamiliar terms he used?

- Agenda
- A* heuristic
- Dependencies
- Slashes and features

Agenda

- An agenda is the list of cells to be processed
- In CKY, we just went up to down, left to right, but we can use any other agenda
- Best-first parsing: may never need to expand cells that are low in the agenda
  - Pro: Saves time on pointless cells
  - Con: if agenda is bad, don't get the correct parse

A* Parsing

- Compute agenda by assigning priorities to cells
- Priority comes from a heuristic estimating the total probability of parsing from current cell to the root. Prefer higher probabilities.
- If heuristic is good, we get correct and fast parsing!

Dependency Grammars

- The CFGs so far are constituency grammars
- Sometimes, we don't care what the intermediate non-terminals are (NP, PP, etc), and only want the relationships between words

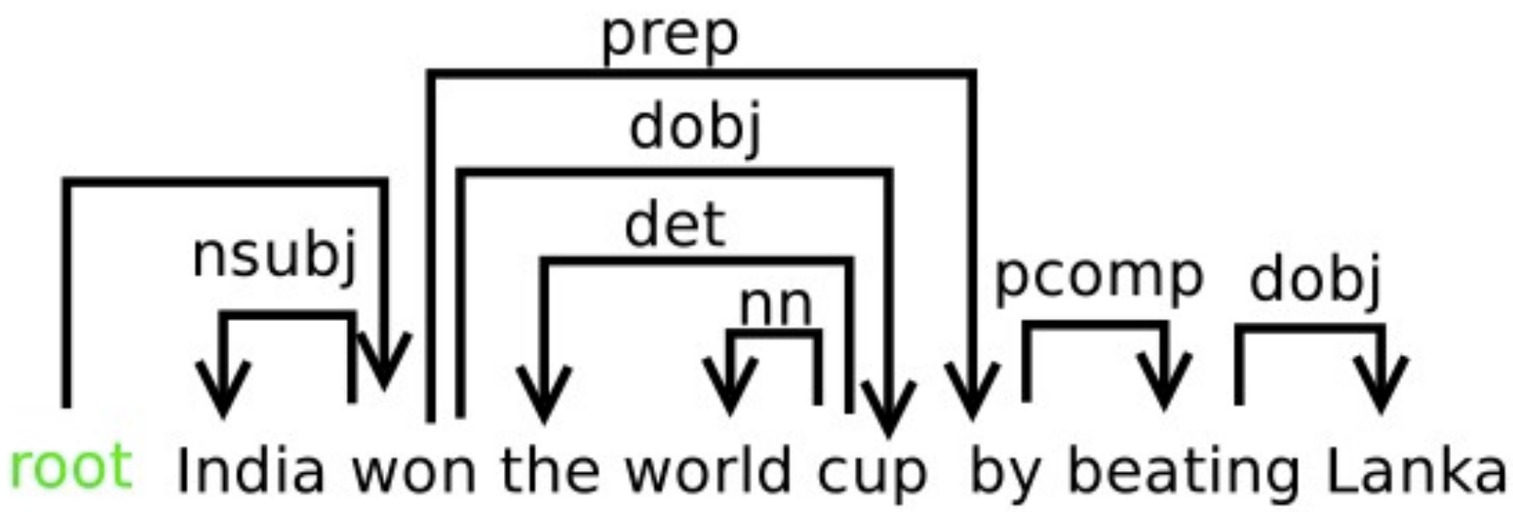


Figure : Output of Stanford dependency parser

Slashes and Features?

- We often want to augment CFGs a little to get more expressiveness
  - Compositional Semantics, X-Bar Theory, etc.
- Eg: Combinatory Categorial Grammars
  - likes = (S\NP)/NP