

# Context-Free Parsing: CKY & Earley Algorithms and Probabilistic Parsing

Natural Language Processing  
CS 6120—Spring 2013  
Northeastern University

David Smith  
with some slides  
from Jason Eisner & Andrew McCallum

# From Shift-Reduce to CKY

- Shift-reduce parsing can make wrong turns, needs backtracking
- Shift-reduce must pop the top of the stack, but how many items to pop?
- Time-space tradeoff
- Chomsky normal form

# Chomsky Normal Form

- Any CFL can be generated by an equivalent grammar in CNF
- Rules of three types
  - $X \rightarrow YZ$                        $X, Y, Z$  nonterminals
  - $X \rightarrow a$                        $X$  nonterminal,  $a$  terminal
  - $S \rightarrow \epsilon$                        $S$  the start symbol
- NB: the derivation of a given string may change

# CNF Conversion

- Create new start symbol
- Remove NTs that can generate epsilon
- Remove NTs that can generate each other, (unary rule cycles)
- Chain rules with RHS  $> 2$
- Related topic: rule Markovization (later)

# CNF Conversion

Original

$$S \rightarrow TST \mid aB$$

$$T \rightarrow B \mid S$$

$$B \rightarrow b \mid \epsilon$$

# CNF Conversion

## Original

$$S \rightarrow TST \mid aB$$

$$T \rightarrow B \mid S$$

$$B \rightarrow b \mid \epsilon$$

## New start symbol

$$S_0 \rightarrow S$$

$$S \rightarrow TST \mid aB$$

$$T \rightarrow B \mid S$$

$$B \rightarrow b \mid \epsilon$$

# CNF Conversion

Original

$$S \rightarrow TST \mid aB$$

$$T \rightarrow B \mid S$$

$$B \rightarrow b \mid \epsilon$$

Remove nulls

$$S_0 \rightarrow S$$

$$S \rightarrow TST \mid aB \mid a \mid ST \mid TS$$

$$T \rightarrow B \mid S$$

$$B \rightarrow b \mid \epsilon$$

New start symbol

$$S_0 \rightarrow S$$

$$S \rightarrow TST \mid aB$$

$$T \rightarrow B \mid S$$

$$B \rightarrow b \mid \epsilon$$

# CNF Conversion

## Original

$$S \rightarrow TST \mid aB$$

$$T \rightarrow B \mid S$$

$$B \rightarrow b \mid \epsilon$$

## Remove nulls

$$S_0 \rightarrow S$$

$$S \rightarrow TST \mid aB \mid a \mid ST \mid TS$$

$$T \rightarrow B \mid S$$

$$B \rightarrow b \mid \epsilon$$

## New start symbol

$$S_0 \rightarrow S$$

$$S \rightarrow TST \mid aB$$

$$T \rightarrow B \mid S$$

$$B \rightarrow b \mid \epsilon$$

## Remove unary cycles

$$S_0 \rightarrow TST \mid aB \mid a \mid ST \mid TS$$

$$S \rightarrow TST \mid aB \mid a \mid ST \mid TS$$

$$T \rightarrow b \mid TST \mid aB \mid a \mid ST \mid TS$$

$$B \rightarrow b$$



# CNF Conversion

$S_0 \rightarrow TST \mid UB \mid a \mid ST \mid TS$

$S \rightarrow TST \mid UB \mid a \mid ST \mid TS$

$T \rightarrow b \mid TST \mid UB \mid a \mid ST \mid TS$

$B \rightarrow b$

$U \rightarrow a$

**Make terminal  
rules unary**

# CNF Conversion

$$S_0 \rightarrow TST \mid UB \mid a \mid ST \mid TS$$

$$S \rightarrow TST \mid UB \mid a \mid ST \mid TS$$

$$T \rightarrow b \mid TST \mid UB \mid a \mid ST \mid TS$$

$$B \rightarrow b$$

$$U \rightarrow a$$

**Make terminal  
rules unary**

$$S_0 \rightarrow TZ \mid UB \mid a \mid ST \mid TS$$

$$S \rightarrow TZ \mid UB \mid a \mid ST \mid TS$$

$$T \rightarrow b \mid TZ \mid UB \mid a \mid ST \mid TS$$

$$B \rightarrow b$$

$$U \rightarrow a$$

$$Z \rightarrow ST$$

**Make non-terminal  
rules binary**

# CKY Algorithm

- Input: string of  $n$  words
- Output (of recognizer): grammatical or not
- Dynamic programming in a **chart**:
  - rows labeled 0 to  $n-1$
  - columns labeled 1 to  $n$
  - cell  $[i,j]$  lists possible constituents spanning words between  $i$  and  $j$

# CKY Algorithm

- **for**  $i := 1$  to  $n$ 
  - Add to  $[i-1, i]$  all (part-of-speech) categories for the  $i^{\text{th}}$  word
- **for** width  $:= 2$  to  $n$ 
  - **for** start  $:= 0$  to  $n$ -width
    - Define end  $:=$  start + width
    - **for** mid  $:=$  start+1 to end-1
      - **for** every constituent X in [start,mid]
      - **for** every constituent Y in [mid,end]
      - **for** all ways of combining X and Y (if any)
      - Add the resulting constituent to [start,end] ~~if it's not already there.~~

# CKY Algorithm

- **for**  $i := 1$  to  $n$ 
  - Add to  $[i-1, i]$  all (part-of-speech) categories for the  $i^{\text{th}}$  word
- **for** width  $:= 2$  to  $n$ 
  - **for** start  $:= 0$  to  $n$ -width
    - Define end  $:=$  start + width
    - **for** mid  $:=$  start+1 to end-1
      - **for** every constituent  $X$  in  $[start, mid]$
      - **for** every constituent  $Y$  in  $[mid, end]$
      - **for** all ways of combining  $X$  and  $Y$  (if any)
      - Add the resulting constituent to  $[start, end]$  ~~if it's not already there.~~

Time complexity?

# CKY Algorithm

- **for**  $i := 1$  to  $n$ 
  - Add to  $[i-1, i]$  all (part-of-speech) categories for the  $i^{\text{th}}$  word
- **for** width  $:= 2$  to  $n$ 
  - **for** start  $:= 0$  to  $n$ -width
    - Define end  $:=$  start + width
    - **for** mid  $:=$  start+1 to end-1
      - **for** every constituent  $X$  in  $[start, mid]$
      - **for** every constituent  $Y$  in  $[mid, end]$
      - **for** all ways of combining  $X$  and  $Y$  (if any)
      - Add the resulting constituent to  $[start, end]$  ~~if it's not already there.~~

Time complexity?

$O(Gn^3)$

# CKY Algorithm

- **for**  $i := 1$  to  $n$ 
  - Add to  $[i-1, i]$  all (part-of-speech) categories for the  $i^{\text{th}}$  word
- **for** width  $:= 2$  to  $n$ 
  - **for** start  $:= 0$  to  $n$ -width
    - Define end  $:=$  start + width
    - **for** mid  $:=$  start+1 to end-1
      - **for** every constituent  $X$  in  $[start, mid]$
      - **for** every constituent  $Y$  in  $[mid, end]$
      - **for** all ways of combining  $X$  and  $Y$  (if any)
      - Add the resulting constituent to  $[start, end]$  ~~if it's not already there.~~

Time complexity?

$O(Gn^3)$

Space complexity?

# CKY Algorithm

- for  $i := 1$  to  $n$ 
  - Add to  $[i-1, i]$  all (part-of-speech) categories for the  $i^{\text{th}}$  word
- for width  $:= 2$  to  $n$ 
  - for start  $:= 0$  to  $n$ -width
    - Define end  $:=$  start + width
    - for mid  $:=$  start+1 to end-1
      - for every constituent  $X$  in  $[start, mid]$
      - for every constituent  $Y$  in  $[mid, end]$
      - for all ways of combining  $X$  and  $Y$  (if any)
      - Add the resulting constituent to  $[start, end]$  ~~if it's not already there.~~

Time complexity?

$O(Gn^3)$

Space complexity?

$O(Tn^2)$



time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3				
1		NP 4 VP 4			
2			P 2 V 5		
3				Det 1	
4					N 8

NP → time  
Vst → time  
NP → flies  
VP → flies  
P → like  
V → like  
Det → an  
N → arrow

1 S → NP VP  
6 S → Vst NP  
2 S → S PP  
1 VP → V NP  
2 VP → VP PP  
1 NP → Det N  
2 NP → NP PP  
3 NP → NP NP  
0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3				
1		NP 4 VP 4			
2			P 2 V 5		
3				Det 1	
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10			
1		NP 4 VP 4			
2			P 2 V 5		
3				Det 1	
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8			
1		NP 4 VP 4			
2			P 2 V 5		
3				Det 1	
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13			
1		NP 4 VP 4			
2			P 2 V 5		
3				Det 1	
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13			
1		NP 4 VP 4	-		
2			P 2 V 5	-	
3				Det 1	
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13			
1		NP 4 VP 4	-		
2			P 2 V 5	-	
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13			
1		NP 4 VP 4	-		
2			P 2 V 5	-	
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP



time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-		
1		NP 4 VP 4	-	-	
2			P 2 V 5	-	PP 12
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-		
1		NP 4 VP 4	-	-	
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-		
1		NP 4 VP 4	-	-	
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	
1		NP 4 VP 4	-	-	NP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	
1		NP 4 VP 4	-	-	NP 18 S 21
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP



time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22 S 27
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22 S 27
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22 S 27 NP 24
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22 S 27 NP 24 S 27
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22 S 27 NP 24 S 27 S 22
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22 S 27 NP 24 S 27 S 22 S 27
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

# Follow backpointers ... **S**

time 1 flies 2 like 3 an 4 arrow 5

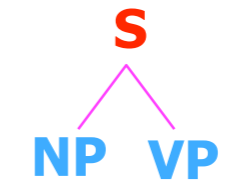
0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 <b>S 22</b> S 27 NP 24 S 27 S 22 S 27
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8

- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP



time 1 flies 2 like 3 an 4 arrow 5

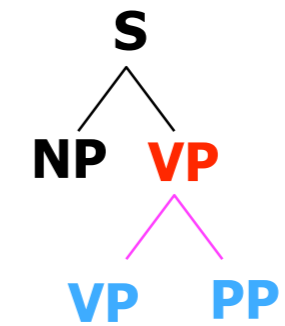
0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22 S 27 NP 24 S 27 S 22 S 27
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8



- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

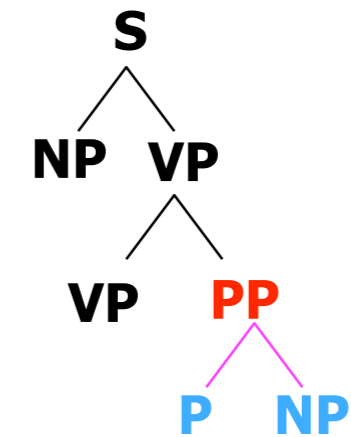
0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22 S 27 NP 24 S 27 S 22 S 27
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8



- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

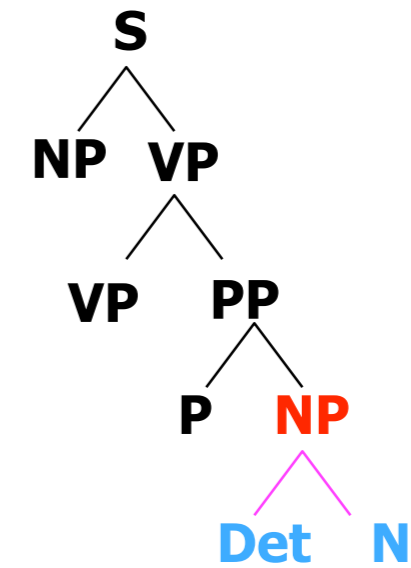
0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22 S 27 NP 24 S 27 S 22 S 27
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8



- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

0	NP 3 Vst 3	NP 10 S 8 S 13	-	-	NP 24 S 22 S 27 NP 24 S 27 S 22 S 27
1		NP 4 VP 4	-	-	NP 18 S 21 VP 18
2			P 2 V 5	-	PP 12 VP 16
3				Det 1	NP 10
4					N 8



- 1 S → NP VP
- 6 S → Vst NP
- 2 S → S PP
- 1 VP → V NP
- 2 VP → VP PP
- 1 NP → Det N
- 2 NP → NP PP
- 3 NP → NP NP
- 0 PP → P NP

# Parsing as Deduction

- CKY as inference rules
- CKY as Prolog program
- But Prolog is top-down with backtracking
  - i.e., “backward chaining”, CKY is “forward chaining”
- Inference rules as Boolean semiring

# Probabilistic CFGs

- Generative process (already familiar)
- It's context free: Rules are applied independently, therefore we multiply their probabilities
- How to estimate probabilities?
  - Supervised and unsupervised

# Questions for PCFGs

- What is the most likely parse for a sentence? (parsing)
- What is the probability of a sentence? (language modeling)
- What rule probabilities maximize the probability of a sentence? (parameter estimation)

# Algorithms for PCFGs

- Exact analogues to HMM algorithms
- Parsing: Viterbi CKY
- Language modeling: inside probabilities
- Parameter estimation: inside-outside probabilities with EM



# Parsing as Deduction

$\forall A, B, C \in N, W \in V, 0 \leq i, j, k \leq m$

$constit(B, i, j) \wedge constit(C, j, k) \wedge A \rightarrow BC \Rightarrow constit(A, i, k)$

$word(W, i) \wedge A \rightarrow W \Rightarrow constit(A, i, i + 1)$

## In Prolog:

```
constit(A, I1, I) :-  
    word(I, W),  
    (A ----> [W]),  
    I1 is I - 1.
```

```
constit(A, I, K) :-  
    constit(B, I, J),  
    constit(C, J, K),  
    (A ----> [B, C]).
```

*But Prolog uses top-down search with backtracking...*

# Parsing as Deduction

$$\forall A, B, C \in N, W \in V, 0 \leq i, j, k \leq m$$

$$\text{constit}(B, i, j) \wedge \text{constit}(C, j, k) \wedge A \rightarrow BC \Rightarrow \text{constit}(A, i, k)$$

$$\text{word}(W, i) \wedge A \rightarrow W \Rightarrow \text{constit}(A, i, i + 1)$$

$$\text{constit}(A, i, k) = \bigvee_{B, C, j} \text{constit}(B, i, j) \wedge \text{constit}(C, j, k) \wedge A \rightarrow B C$$

$$\text{constit}(A, i, j) = \bigvee_W \text{word}(W, i, j) \wedge A \rightarrow W$$

# Parsing as Deduction

$$\mathit{constit}(A, i, k) = \bigvee_{B, C, j} \mathit{constit}(B, i, j) \wedge \mathit{constit}(C, j, k) \wedge A \rightarrow B C$$

$$\mathit{constit}(A, i, j) = \bigvee_W \mathit{word}(W, i, j) \wedge A \rightarrow W$$

$$\begin{aligned} \mathit{score}(\mathit{constit}(A, i, k)) &= \max_{B, C, j} \mathit{score}(\mathit{constit}(B, i, j)) \\ &\quad \cdot \mathit{score}(\mathit{constit}(C, j, k)) \\ &\quad \cdot \mathit{score}(A \rightarrow B C) \end{aligned}$$

$$\mathit{score}(\mathit{constit}(A, i, j)) = \max_W \mathit{score}(\mathit{word}(W, i, j)) \cdot \mathit{score}(A \rightarrow W)$$

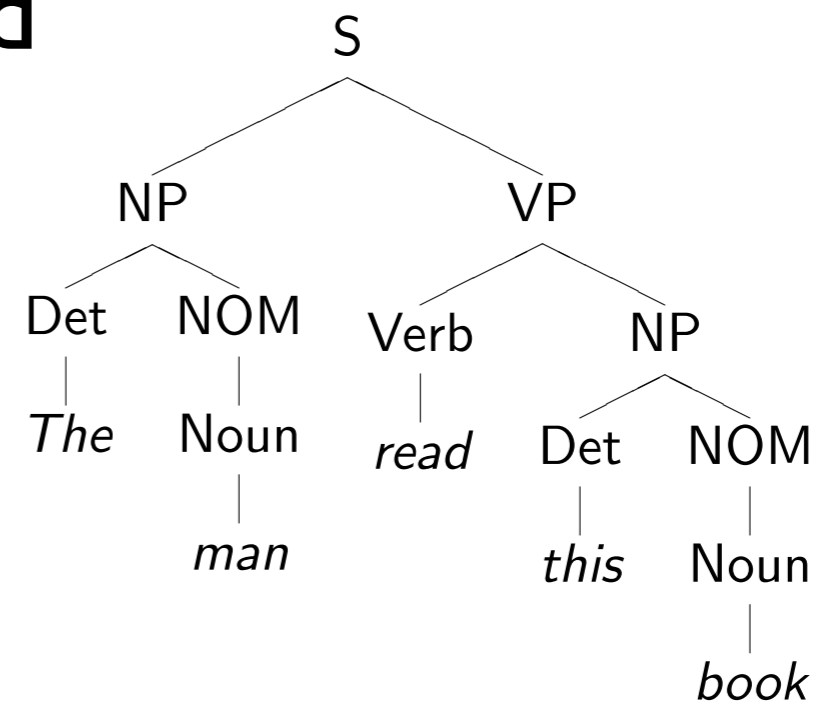
And how about the inside algorithm?

# Problems with Inside-Outside EM

- Each sentence at each iteration takes  $O(m^3n^3)$
- Local maxima even more problematic than for HMMs: Charniak (1993) found a different maximum for each of 300 trials
- More NTs needed to learn a good model
- NTs don't correspond to intuitions: HMMs are easier to constrain with tag dictionaries

# Treebank Grammars

- What rules would you extract from this tree?
- What probabilities would you assign them?



# Treebank Grammars

- Penn Treebank
- Lots of rules have high fanout (flat phrases)
- Lots of unary cycles
- How should we evaluate?
- What are the consequences of CNF conversion?

# Overview

- Complete inside-outside: outside computations
- Earley's algorithm: combining bottom-up and top-down parsing
- Better treebank grammars: annotating nonterminals

# Inside & Viterbi Algorithms

**Let**  $\beta_A(i, j) = p(\text{constit}(A, i, j))$   
 $= p(w_{ij} \mid \text{nonterminal } A \text{ from } i \text{ to } j)$

NB: index *between* words;  
M&S index words

$$\beta_A(i, k) = \sum_{B, C, j} \beta_B(i, j) \cdot \beta_C(j, k) \cdot p(A \rightarrow B C)$$

**Let**  $\delta_A(i, j) = p_{best}(\text{constit}(A, i, j))$

$$\delta_A(i, k) = \max_{B, C, j} \delta_B(i, j) \cdot \delta_C(j, k) \cdot p(A \rightarrow B C)$$

$$\beta_S(0, n) = ?$$

$$\delta_S(0, n) = ?$$



# Inside & Outside

$\text{constit}(A, i, j)$

$p(\text{words } 0-i, \text{ words } j-n, \text{ constit})$

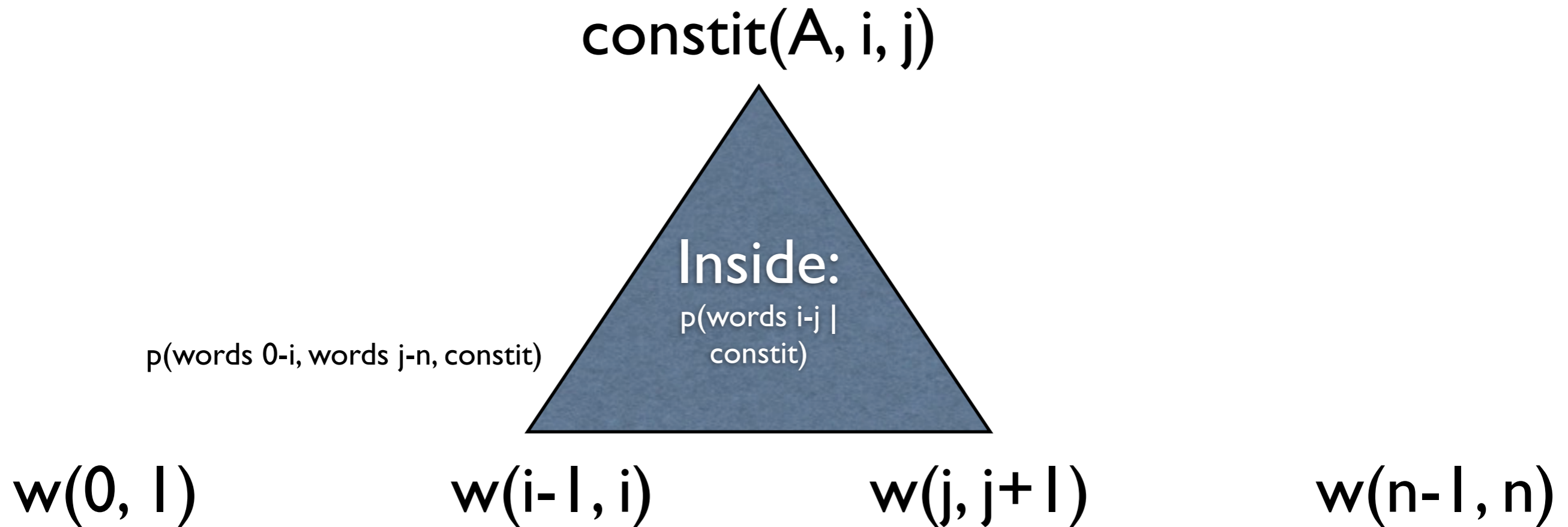
$w(0, 1)$

$w(i-1, i)$

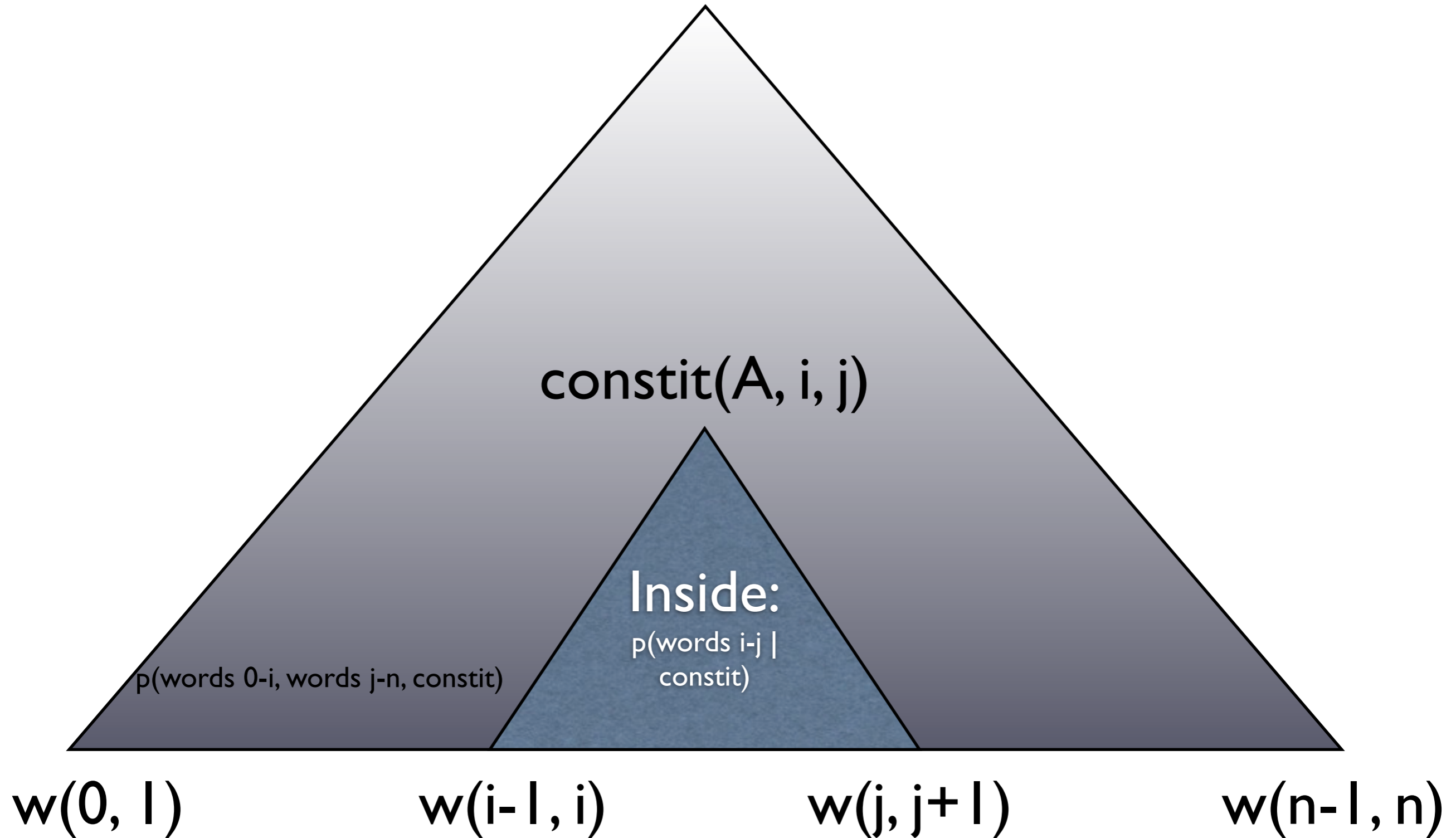
$w(j, j+1)$

$w(n-1, n)$

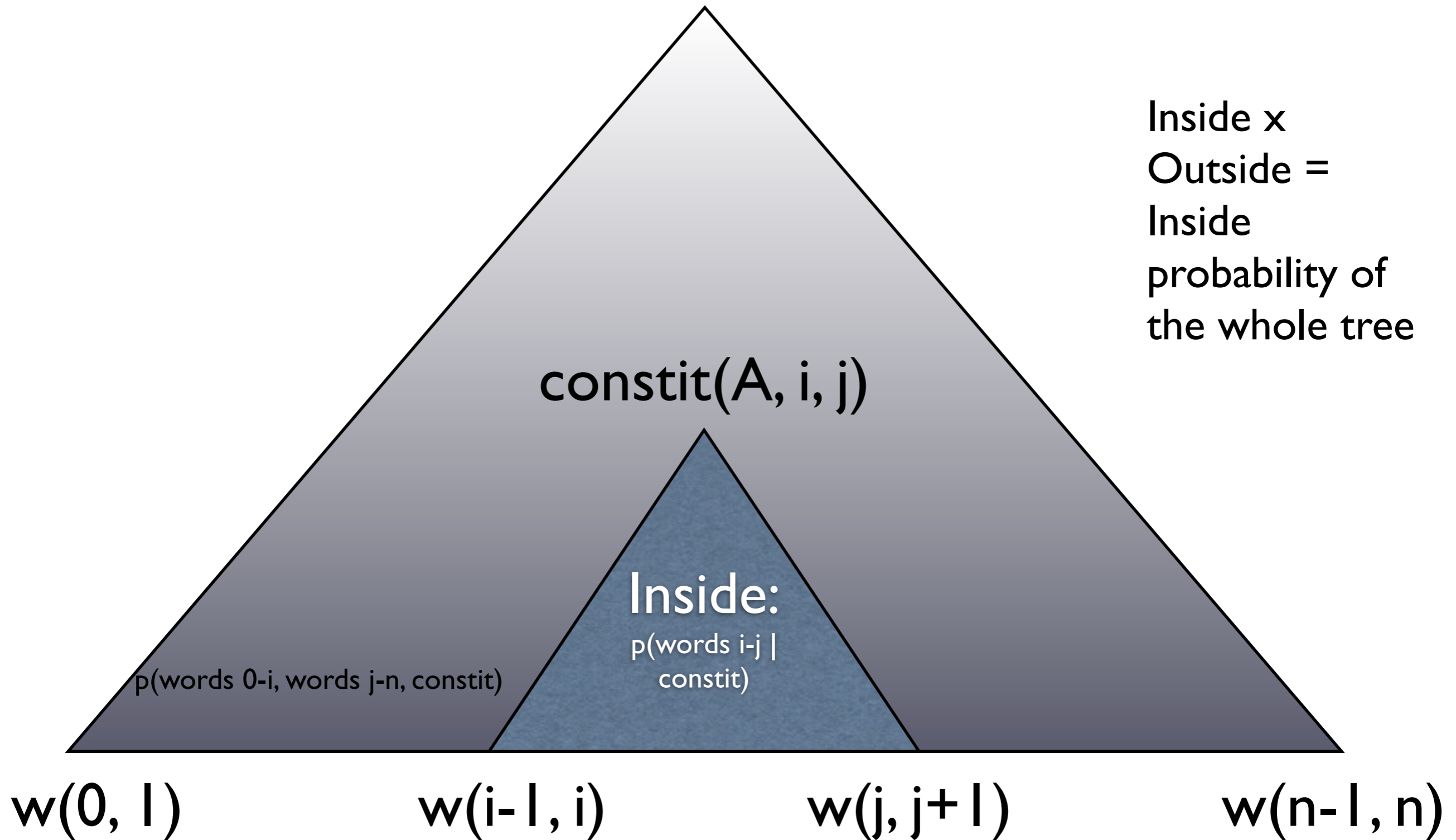
# Inside & Outside



# Inside & Outside



# Inside & Outside



Inside x  
Outside =  
Inside  
probability of  
the whole tree

# Outside Algorithm

$$\alpha_A(i, j) = p(w_{0,i}, A_{i,j}, w_{j,n})$$

Uses inside  
probs.

$$\alpha_A(i, j) = \sum_{B, C, k=j}^n \alpha_B(i, k) \cdot \beta_C(j, k) \cdot p(B \rightarrow A C) \\ + \sum_{B, C, k=0}^i \alpha_B(k, j) \cdot \beta_C(k, i) \cdot p(B \rightarrow C A)$$

$$\alpha_S(0, n) = ?$$

$$\alpha_{PP}(0, n) = ?$$

Some  
resemblance  
to derivative  
product rule

# Top-Down/Bottom-Up

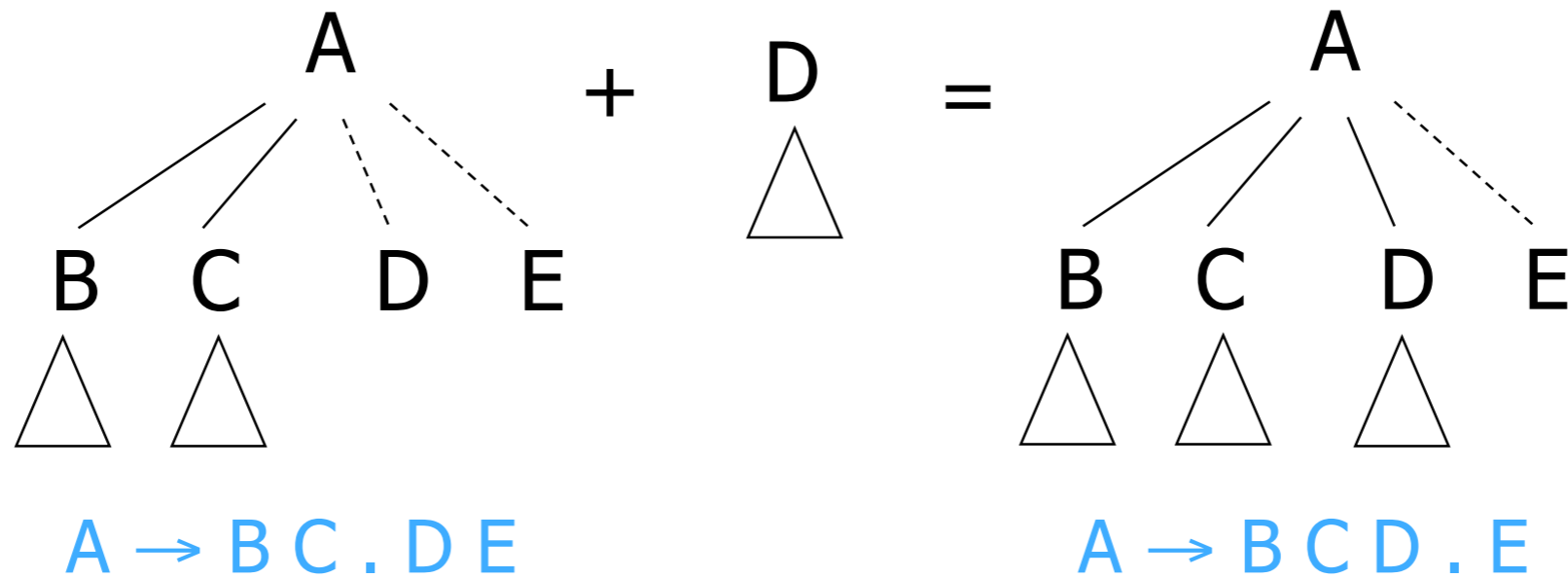
- Top-down parsers
  - Can get caught in infinite loops
  - Take exponential time backtracking
- CKY
  - Needs Chomsky normal form
  - Builds all possible constituents

# Earley Parser (1970)

- Nice combination of
  - dynamic programming
  - incremental interpretation
  - avoids infinite loops
  - no restrictions on the form of the context-free grammar.  
 **$A \rightarrow B C \textit{ the D of}$**  causes no problems
  - $O(n^3)$  worst case, but faster for many grammars
  - Uses left context and optionally right context to constrain search.

# Earley's Overview

- Finds constituents and partial constituents in input
  - $A \rightarrow BC.DE$  is partial: only the first half of the  $A$





# Earley's Overview

- Proceeds incrementally left-to-right
  - Before it reads word 5, it has already built all hypotheses that are consistent with first 4 words
  - Reads word 5 & attaches it to immediately preceding hypotheses. Might yield new constituents that are then attached to hypotheses immediately preceding *them* ...
  - E.g., attaching **D** to  $A \rightarrow B C . D E$  gives  $A \rightarrow B C D . E$
  - Attaching **E** to that gives  $A \rightarrow B C D E .$
  - Now we have a complete **A** that we can attach to hypotheses immediately preceding the **A**, etc.

# The Parse Table

- Columns 0 through n corresponding to the gaps between words
- Entries in column 5 look like (3, NP → NP . PP)
  - (but we'll omit the → etc. to save space)
  - Built while processing word 5
  - Means that the input substring from 3 to 5 matches the initial NP portion of a NP → NP PP rule
  - Dot shows how much we've matched as of column 5
  - Perfectly fine to have entries like (3, VP → is it . true that S)

# The Parse Table

- Entries in column 5 look like (3, NP → NP . PP)
- What will it mean that we have this entry?
  - *Unknown right context*: Doesn't mean we'll necessarily be able to find a VP starting at column 5 to complete the S.
  - *Known left context*: Does mean that some dotted rule back in column 3 is looking for an S that starts at 3.
    - So if we actually do find a VP starting at column 5, allowing us to complete the S, then we'll be able to attach the S to something.
    - And when that something is complete, it too will have a customer to *its* left ...
    - In short, a top-down (i.e., goal-directed) parser: it chooses to start building a constituent not because of the input but because that's what the left context needs. In **the spoon**, won't build **spoon** as a verb because there's no way to use a verb there.
    - So any hypothesis in column 5 *could* get used in the correct parse, if words 1-5 are continued in just the right way by words 6-n.

# Earley's as a Recognizer

- Add **ROOT**  $\rightarrow$  **. S** to column 0.
- For each  $j$  from 0 to  $n$ :
  - For each dotted rule in column  $j$ , (including those we add as we go!) look at what's after the dot:
    - If it's a word  $w$ , **SCAN**:
      - If  $w$  matches the input word between  $j$  and  $j+1$ , advance the dot and add the resulting rule to column  $j+1$
    - If it's a non-terminal  $X$ , **PREDICT**:
      - Add all rules for  $X$  to the bottom of column  $j$ , with the dot at the start: e.g. **X**  $\rightarrow$  **. Y Z**
    - If there's nothing after the dot, **ATTACH**:
      - We've finished some constituent,  $A$ , that started in column  $l < j$ . So for each rule in column  $j$  that has  $A$  after the dot: Advance the dot and add the result to the bottom of column  $j$ .
- Output “yes” just if last column has **ROOT**  $\rightarrow$  **S .**
- **NOTE: Don't add an entry to a column if it's already there!**

# Earley's Summary

- Process all hypotheses one at a time in order.  
(Current hypothesis is shown in blue.)
- This may add **new hypotheses** to the end of the to-do list, or try to add **old hypotheses** again.
- Process a hypothesis according to what follows the dot:
  - If a word, **scan** input and see if it matches
  - If a nonterminal, **predict** ways to match it
    - (we'll predict blindly, but could reduce # of predictions by *looking ahead* k symbols in the input and only making predictions that are compatible with this limited *right context*)
  - If nothing, then we have a complete constituent, so **attach** it to all its customers

# A (Whimsical) Grammar

S → NP VP

NP → Det N

NP → NP PP

VP → V NP

VP → VP PP

PP → P NP

NP → Papa

N → caviar

N → spoon

V → ate

P → with

Det → the

Det → a

## An Input Sentence

*Papa ate the caviar with a spoon.*

0
0 ROOT . S

**initialize**



*Remember this stands for (0, ROOT → . S)*

0
0 ROOT . S
0 S . NP VP

**predict** the kind of S we are looking for

*Remember this stands for (0, S → . NP VP)*



0
0 ROOT . S
0 S . NP VP
0 NP . Det N
0 NP . NP PP
0 NP . Papa

**predict** the kind of NP we are looking for  
*(actually we'll look for 3 kinds: any of the 3 will do)*

0
0 ROOT . S
0 S . NP VP
0 NP . Det N
0 NP . NP PP
0 NP . Papa
0 Det . the
0 Det . a

**predict** the kind of Det we are looking for (*2 kinds*)

0
0 ROOT . S
0 S . NP VP
0 NP . Det N
0 NP . NP PP
0 NP . Papa
0 Det . the
0 Det . a

**predict** the kind of NP we're looking for  
*but we were already looking for these so  
 don't add duplicate goals! Note that this happened  
 when we were processing a left-recursive rule.*

0	Papa	1
0 ROOT . S	0 NP Papa .	
0 S . NP VP		
0 NP . Det N		
0 NP . NP PP		
0 NP . Papa		
0 Det . the		
0 Det . a		

**scan:** the desired word is in the input!

0	Papa	1
0 ROOT . S	0 NP Papa .	
0 S . NP VP		
0 NP . Det N		
0 NP . NP PP		
0 NP . Papa		
0 Det . the		
0 Det . a		

scan: failure

0 Papa 1	
0 ROOT . S	0 NP Papa .
0 S . NP VP	
0 NP . Det N	
0 NP . NP PP	
0 NP . Papa	
0 Det . the	
0 Det . a	

scan: failure

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		
0 NP . Papa		
0 Det . the		
0 Det . a		

**attach** the newly created NP  
 (which starts at 0) to its **customers**  
 (incomplete constituents that *end* at 0  
 and have NP after the dot)

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		
0 Det . a		

predict



0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		

predict

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate

predict

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate

predict

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate
		1 P . with

predict

0	Papa	1	ate	2
0 ROOT . S	0 NP Papa .	1 V ate .		
0 S . NP VP	0 S NP . VP			
0 NP . Det N	0 NP NP . PP			
0 NP . NP PP	1 VP . V NP			
0 NP . Papa	1 VP . VP PP			
0 Det . the	1 PP . P NP			
0 Det . a	1 V . ate	scan: success!		
	1 P . with			

0	Papa	1	ate	2
0 ROOT . S	0 NP Papa .	1 V ate .		
0 S . NP VP	0 S NP . VP			
0 NP . Det N	0 NP NP . PP			
0 NP . NP PP	1 VP . V NP			
0 NP . Papa	1 VP . VP PP			
0 Det . the	1 PP . P NP			
0 Det . a	1 V . ate			
	1 P . with	<b>scan: failure</b>		

0	Papa	1	ate	2
0 ROOT . S	0 NP Papa .			1 V ate .
0 S . NP VP	0 S NP . VP			1 VP V . NP
0 NP . Det N	0 NP NP . PP			
0 NP . NP PP	1 VP . V NP			
0 NP . Papa	1 VP . VP PP			
0 Det . the	1 PP . P NP			
0 Det . a	1 V . ate			
	1 P . with			

attach

0	Papa	1	ate	2
0 ROOT . S	0 NP Papa .	1 V ate .		
0 S . NP VP	0 S NP . VP	1 VP V . NP		
0 NP . Det N	0 NP NP . PP	2 NP . Det N		
0 NP . NP PP	1 VP . V NP	2 NP . NP PP		
0 NP . Papa	1 VP . VP PP	2 NP . Papa		
0 Det . the	1 PP . P NP			
0 Det . a	1 V . ate			
	1 P . with			

predict



0	Papa	1	ate	2
0 ROOT . S	0 NP Papa .	1 V ate .		
0 S . NP VP	0 S NP . VP	1 VP V . NP		
0 NP . Det N	0 NP NP . PP	2 NP . Det N		
0 NP . NP PP	1 VP . V NP	2 NP . NP PP		
0 NP . Papa	1 VP . VP PP	2 NP . Papa		
0 Det . the	1 PP . P NP	2 Det . the		
0 Det . a	1 V . ate	2 Det . a		
	1 P . with			

**predict** (these next few steps should look familiar)

0	Papa	1	ate	2
0 ROOT . S	0 NP Papa .	1 V ate .		
0 S . NP VP	0 S NP . VP	1 VP V . NP		
0 NP . Det N	0 NP NP . PP	2 NP . Det N		
0 NP . NP PP	1 VP . V NP	2 NP . NP PP		
0 NP . Papa	1 VP . VP PP	2 NP . Papa		
0 Det . the	1 PP . P NP	2 Det . the		
0 Det . a	1 V . ate	2 Det . a		
	1 P . with			

predict

0	Papa	1	ate	2
0 ROOT . S	0 NP Papa .	1 V ate .		
0 S . NP VP	0 S NP . VP	1 VP V . NP		
0 NP . Det N	0 NP NP . PP	2 NP . Det N		
0 NP . NP PP	1 VP . V NP	2 NP . NP PP		
0 NP . Papa	1 VP . VP PP	2 NP . Papa		
0 Det . the	1 PP . P NP	2 Det . the		
0 Det . a	1 V . ate	2 Det . a		
	1 P . with			

**scan** (*this time we fail since Papa is not the next word*)

0	Papa	1	ate	2	the	3
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .			
0 S . NP VP	0 S NP . VP	1 VP V . NP				
0 NP . Det N	0 NP NP . PP	2 NP . Det N				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa				
0 Det . the	1 PP . P NP	2 Det . the				scan: success!
0 Det . a	1 V . ate	2 Det . a				
	1 P . with					

0	Papa	1	ate	2	the	3
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .			
0 S . NP VP	0 S NP . VP	1 VP V . NP				
0 NP . Det N	0 NP NP . PP	2 NP . Det N				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa				
0 Det . the	1 PP . P NP	2 Det . the				
0 Det . a	1 V . ate	2 Det . a				
	1 P . with					

0	Papa	1	ate	2	the	3
0 ROOT . S	0 NP Papa .	1 V ate .		2 Det the .		
0 S . NP VP	0 S NP . VP	1 VP V . NP		2 NP Det . N		
0 NP . Det N	0 NP NP . PP	2 NP . Det N				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa				
0 Det . the	1 PP . P NP	2 Det . the				
0 Det . a	1 V . ate	2 Det . a				
	1 P . with					

0	Papa	1	ate	2	the	3
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .			
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N			
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar			
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon			
0 NP . Papa	1 VP . VP PP	2 NP . Papa				
0 Det . the	1 PP . P NP	2 Det . the				
0 Det . a	1 V . ate	2 Det . a				
	1 P . with					

0	Papa	1	ate	2	the	3	caviar	4
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N					
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar					
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon					
0 NP . Papa	1 VP . VP PP	2 NP . Papa						
0 Det . the	1 PP . P NP	2 Det . the						
0 Det . a	1 V . ate	2 Det . a						
	1 P . with							



0	Papa	1	ate	2	the	3	caviar	4
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N					
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar					
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon					
0 NP . Papa	1 VP . VP PP	2 NP . Papa						
0 Det . the	1 PP . P NP	2 Det . the						
0 Det . a	1 V . ate	2 Det . a						
	1 P . with							

0	Papa	1	ate	2	the	3	caviar	4
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	3 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar					
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon					
0 NP . Papa	1 VP . VP PP	2 NP . Papa						
0 Det . the	1 PP . P NP	2 Det . the						
0 Det . a	1 V . ate	2 Det . a						
	1 P . with							

attach

0	Papa	1	ate	2	the	3	caviar	4
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa						
0 Det . the	1 PP . P NP	2 Det . the						
0 Det . a	1 V . ate	2 Det . a						
	1 P . with							

**attach**  
*(again!)*

0	Papa	1	ate	2	the	3	caviar	4
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa					0 S NP VP .	
0 Det . the	1 PP . P NP	2 Det . the					1 VP VP . PP	
0 Det . a	1 V . ate	2 Det . a						
	1 P . with							

**attach**  
*(again!)*

0	Papa	1	ate	2	the	3	caviar	4
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa						0 S NP VP .
0 Det . the	1 PP . P NP	2 Det . the						1 VP VP . PP
0 Det . a	1 V . ate	2 Det . a						4 PP . P NP
	1 P . with							

	0	Papa	1	ate	2	the	3	caviar	4
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .					
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .					
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .					
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP					
0 NP . Papa	1 VP . VP PP	2 NP . Papa						0 S NP VP .	
0 Det . the	1 PP . P NP	2 Det . the						1 VP VP . PP	
0 Det . a	1 V . ate	2 Det . a						4 PP . P NP	
	1 P . with							0 ROOT S .	

**attach**  
*(again!)*

0	Papa	1	ate	2	the	3	caviar	4
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP				
	1 P . with			0 ROOT S .				

0	Papa	1	ate	2	the	3	caviar	4
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP				
	1 P . with			0 ROOT S .				
				4 P . with				



0	Papa	1	ate	2	the	3	caviar	4
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP				
	1 P . with							0 ROOT S .
								4 P . with

0	Papa	1	ate	2	the	3	caviar	4	with	5
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	4 P with .					
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .						
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .						
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP						
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .						
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP						
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP						
	1 P . with			0 ROOT S .						
				4 P . with						

0	Papa	1	ate	2	the	3	caviar	4	with	5
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	4 P with .					
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP					
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .						
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP						
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .						
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP						
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP						
	1 P . with							0 ROOT S .		
								4 P . with		

0	Papa	1	ate	2	the	3	caviar	4	with	5
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	4 P with .					
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP					
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .	5 NP . Det N					
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP	5 NP . NP PP					
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .	5 NP . Papa					
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP						
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP						
	1 P . with			0 ROOT S .						
				4 P . with						

0	Papa	1	ate	2	the	3	caviar	4	with	5
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	4 P with .					
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP					
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .	5 NP . Det N					
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP	5 NP . NP PP					
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .	5 NP . Papa					
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP	5 Det . the					
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP	5 Det . a					
	1 P . with			0 ROOT S .						
				4 P . with						

0	Papa	1	ate	2	the	3	caviar	4	with	5
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	4 P with .					
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP					
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .	5 NP . Det N					
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP	5 NP . NP PP					
0 NP . Papa	1 VP . VP PP	2 NP . Papa			0 S NP VP .	5 NP . Papa				
0 Det . the	1 PP . P NP	2 Det . the			1 VP VP . PP	5 Det . the				
0 Det . a	1 V . ate	2 Det . a			4 PP . P NP	5 Det . a				
	1 P . with				0 ROOT S .					
					4 P . with					

0	Papa	1	ate	2	the	3	caviar	4	with	5
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	4 P with .					
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP					
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .	5 NP . Det N					
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP	5 NP . NP PP					
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .	5 NP . Papa					
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP	5 Det . the					
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP	5 Det . a					
	1 P . with			0 ROOT S .						
				4 P . with						

0	Papa	1	ate	2	the	3	caviar	4	with	5
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	4 P with .					
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP					
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .	5 NP . Det N					
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP	5 NP . NP PP					
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .	5 NP . Papa					
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP	5 Det . the					
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP	5 Det . a					
	1 P . with			0 ROOT S .						
				4 P . with						



e	2	the	3	caviar	4	with	5	a	6
	1 V ate .	2 Det the .	3 N caviar .	4 P with .	5 Det a .				
	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP					
P	2 NP . Det N	3 N . caviar	1 VP V NP .	5 NP . Det N					
	2 NP . NP PP	3 N . spoon	2 NP NP . PP	5 NP . NP PP					
ρ	2 NP . Papa		0 S NP VP .	5 NP . Papa					
	2 Det . the		1 VP VP . PP	5 Det . the					
	2 Det . a		4 PP . P NP	5 Det . a					
			0 ROOT S .						
			4 P . with						

	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
e	1 V ate .	2 Det the .	3 N caviar .	4 P with .	5 Det a .
	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP	5 NP Det . N
P	2 NP . Det N	3 N . caviar	1 VP V NP .	5 NP . Det N	
	2 NP . NP PP	3 N . spoon	2 NP NP . PP	5 NP . NP PP	
ɔ	2 NP . Papa		0 S NP VP .	5 NP . Papa	
	2 Det . the		1 VP VP . PP	5 Det . the	
	2 Det . a		4 PP . P NP	5 Det . a	
			0 ROOT S .		
			4 P . with		

	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
e	1 V ate .	2 Det the .	3 N caviar .	4 P with .	5 Det a .
	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP	5 NP Det . N
P	2 NP . Det N	3 N . caviar	1 VP V NP .	5 NP . Det N	6 N . caviar
	2 NP . NP PP	3 N . spoon	2 NP NP . PP	5 NP . NP PP	6 N . spoon
ρ	2 NP . Papa		0 S NP VP .	5 NP . Papa	
	2 Det . the		1 VP VP . PP	5 Det . the	
	2 Det . a		4 PP . P NP	5 Det . a	
			0 ROOT S .		
			4 P . with		

	<b>e</b>	<b>2</b>	<b>the</b>	<b>3</b>	<b>caviar</b>	<b>4</b>	<b>with</b>	<b>5</b>	<b>a</b>	<b>6</b>	
		1 V	ate .	2 Det	the .	3 N	caviar .	4 P	with .	5 Det	a .
		1 VP	V . NP	2 NP	Det . N	2 NP	Det N .	4 PP	P . NP	5 NP	Det . N
P		2 NP	. Det N	3 N	. caviar	1 VP	V NP .	5 NP	. Det N	6 N	. caviar
		2 NP	. NP PP	3 N	. spoon	2 NP	NP . PP	5 NP	. NP PP	6 N	. spoon
ρ		2 NP	. Papa			0 S	NP VP .	5 NP	. Papa		
		2 Det	. the			1 VP	VP . PP	5 Det	. the		
		2 Det	. a			4 PP	. P NP	5 Det	. a		
						0 ROOT	S .				
						4 P	. with				

	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>
e	1 V ate .	2 Det the .	3 N caviar .	4 P with .	5 Det a .	6 N spoon .
	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP	5 NP Det . N	
P	2 NP . Det N	3 N . caviar	1 VP V NP .	5 NP . Det N	6 N . caviar	
	2 NP . NP PP	3 N . spoon	2 NP NP . PP	5 NP . NP PP	6 N . spoon	
ρ	2 NP . Papa		0 S NP VP .	5 NP . Papa		
	2 Det . the		1 VP VP . PP	5 Det . the		
	2 Det . a		4 PP . P NP	5 Det . a		
			0 ROOT S .			
			4 P . with			

	e	2	the	3	caviar	4	with	5	a	6	spoon	7	
		1 V	ate .	2 Det	the .	3 N	caviar .	4 P	with .	5 Det	a .	6 N	spoon .
		1 VP	V . NP	2 NP	Det . N	2 NP	Det N .	4 PP	P . NP	5 NP	Det . N	5 NP	Det N .
P		2 NP	. Det N	3 N	. caviar	1 VP	V NP .	5 NP	. Det N	6 N	. caviar		
		2 NP	. NP PP	3 N	. spoon	2 NP	NP . PP	5 NP	. NP PP	6 N	. spoon		
ρ		2 NP	. Papa			0 S	NP VP .	5 NP	. Papa				
		2 Det	. the			1 VP	VP . PP	5 Det	. the				
		2 Det	. a			4 PP	. P NP	5 Det	. a				
						0	ROOT S .						
						4 P	. with						

e	2	the	3	caviar	4	with	5	a	6	spoon	7
	1 V ate .	2 Det the .	3 N caviar .	4 P with .	5 Det a .	6 N spoon .					
	1 VP V . NP	2 NP Det . N	2 NP Det N .	4 PP P . NP	5 NP Det . N	5 NP Det N .					
P	2 NP . Det N	3 N . caviar	1 VP V NP .	5 NP . Det N	6 N . caviar	4 PP P NP .					
	2 NP . NP PP	3 N . spoon	2 NP NP . PP	5 NP . NP PP	6 N . spoon	5 NP NP . PP					
ɔ	2 NP . Papa		0 S NP VP .	5 NP . Papa							
	2 Det . the		1 VP VP . PP	5 Det . the							
	2 Det . a		4 PP . P NP	5 Det . a							
			0 ROOT S .								
			4 P . with								

0 Papa 1 ate 2 the 3 caviar 4 with a spoon 7						
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		
	1 P . with			0 ROOT S .		
				4 P . with		



0 Papa 1 ate 2 the 3 caviar 4 with a spoon 7						
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP
	1 P . with			0 ROOT S .		
				4 P . with		

0	Papa	1	ate	2	the	3	caviar	4	with a spoon	7
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP				
	1 P . with			0 ROOT S .		1 VP V NP .				
				4 P . with		2 NP NP . PP				

0 Papa 1 ate 2 the 3 caviar 4 with a spoon 7						
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP
	1 P . with			0 ROOT S .		1 VP V NP .
				4 P . with		2 NP NP . PP
						0 S NP VP .
						1 VP VP . PP

0	Papa	1	ate	2	the	3	caviar	4	with a spoon	7
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP				
	1 P . with			0 ROOT S .		1 VP V NP .				
				4 P . with		2 NP NP . PP				
						0 S NP VP .				
						1 VP VP . PP				
						7 P . with				

0	Papa	1	ate	2	the	3	caviar	4	with a spoon	7
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP				
	1 P . with			0 ROOT S .		1 VP V NP .				
				4 P . with		2 NP NP . PP				
						0 S NP VP .				
						1 VP VP . PP				
						7 P . with				

0	Papa	1	ate	2	the	3	caviar	4	with a spoon	7
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP				
	1 P . with			0 ROOT S .		1 VP V NP .				
				4 P . with		2 NP NP . PP				
						0 S NP VP .				
						1 VP VP . PP				
						7 P . with				

0 Papa 1 ate 2 the 3 caviar 4 with a spoon 7						
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP
	1 P . with			0 ROOT S .		1 VP V NP .
				4 P . with		2 NP NP . PP
						0 S NP VP .
						1 VP VP . PP
						7 P . with
						0 ROOT S .

0	Papa	1	ate	2	the	3	caviar	4	with a spoon	7
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP				
	1 P . with			0 ROOT S .		1 VP V NP .				
				4 P . with		2 NP NP . PP				
						0 S NP VP .				
						1 VP VP . PP				
						7 P . with				
						0 ROOT S .				



0 Papa 1 ate 2 the 3 caviar 4 with a spoon 7						
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP
	1 P . with			0 ROOT S .		1 VP V NP .
				4 P . with		2 NP NP . PP
						0 S NP VP .
						1 VP VP . PP
						7 P . with
						0 ROOT S .

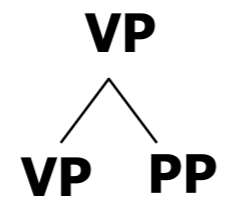
0 Papa 1 ate 2 the 3 caviar 4 with a spoon 7						
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP
	1 P . with			0 ROOT S .		1 VP V NP .
				4 P . with		2 NP NP . PP
						0 S NP VP .
						1 VP VP . PP
						7 P . with
						0 ROOT S .

0	Papa	1	ate	2	the	3	caviar	4	with a spoon	7
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP				
	1 P . with			0 ROOT S .		1 VP V NP .				
				4 P . with		2 NP NP . PP				
						0 S NP VP .				
						1 VP VP . PP				
						7 P . with				
						0 ROOT S .				

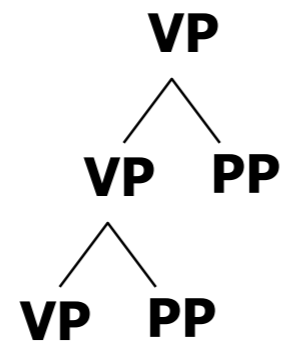
# Left Recursion Kills Pure Top-Down Parsing ...

**VP**

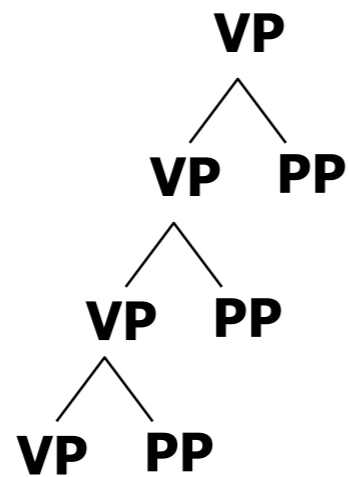
# Left Recursion Kills Pure Top-Down Parsing ...



# Left Recursion Kills Pure Top-Down Parsing ...



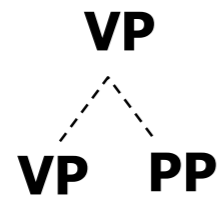
## Left Recursion Kills Pure Top-Down Parsing ...



makes new hypotheses  
ad infinitum before we've  
seen the PPs at all

hypotheses try to predict  
in advance how many  
PP's will arrive in input

... but Earley's Alg is Okay!

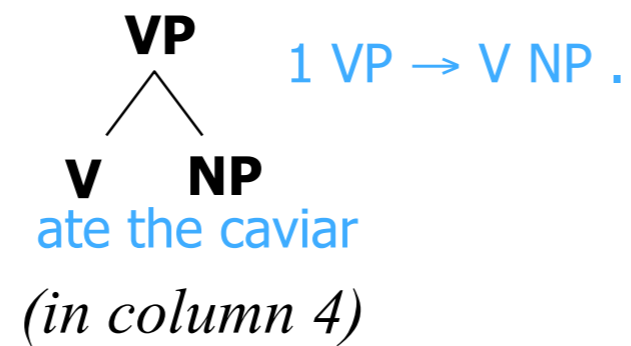
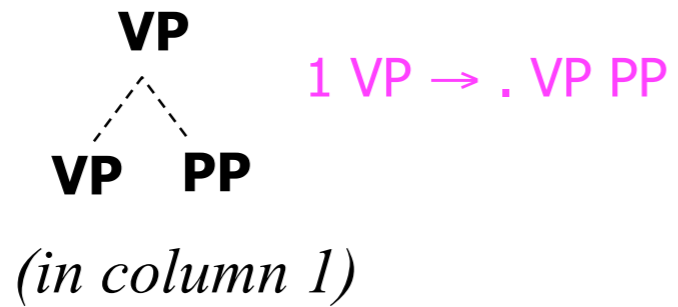


1 VP → . VP PP

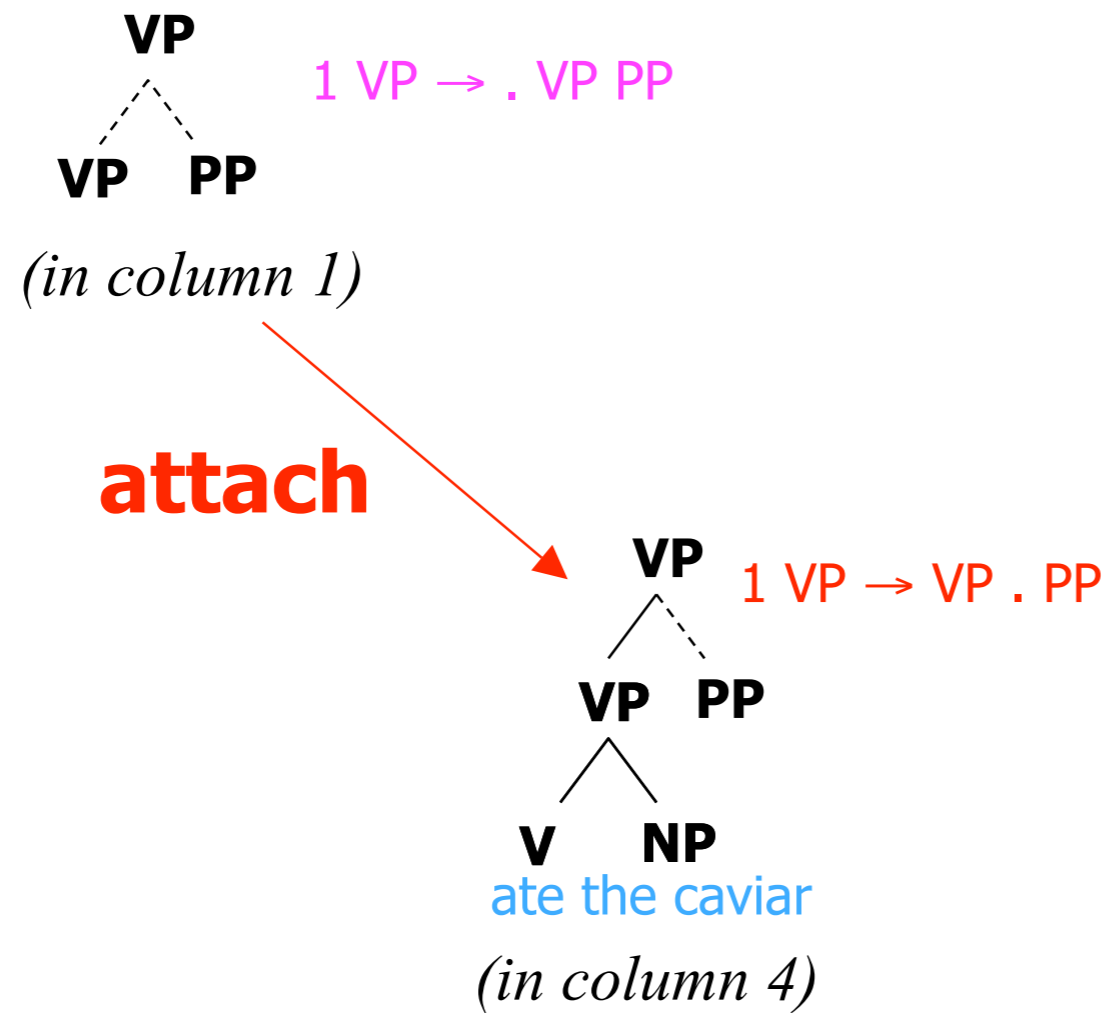
*(in column 1)*



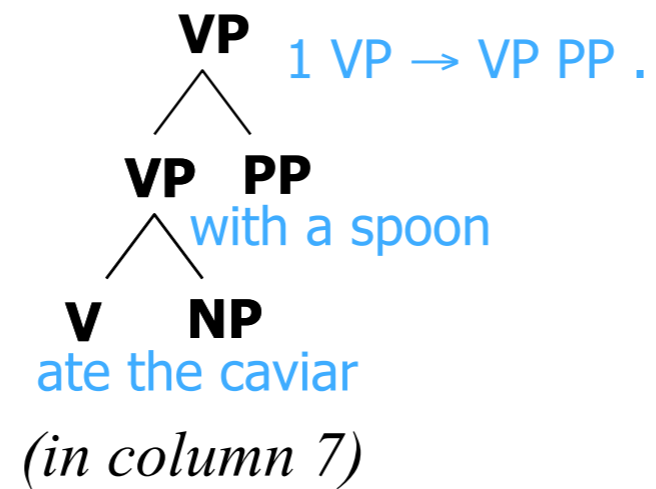
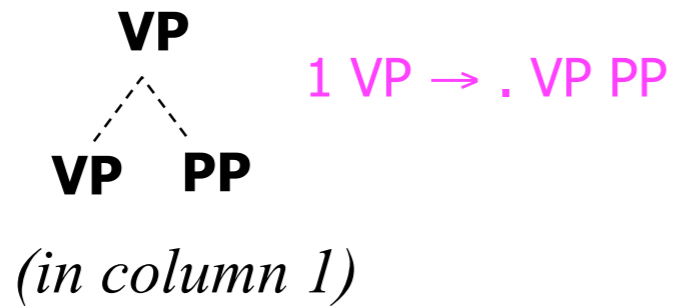
## ... but Earley's Alg is Okay!



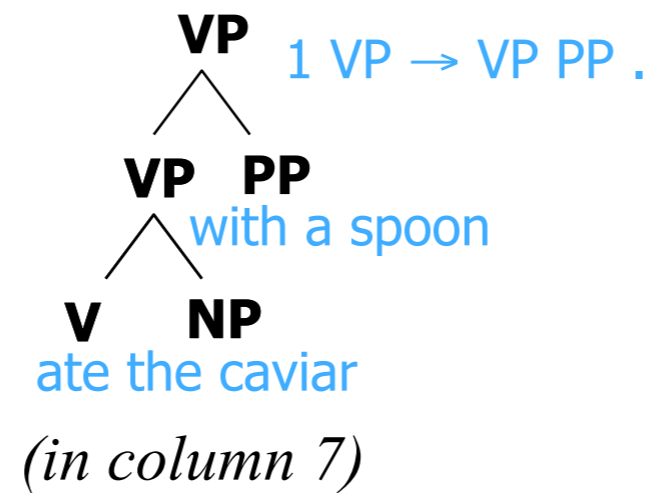
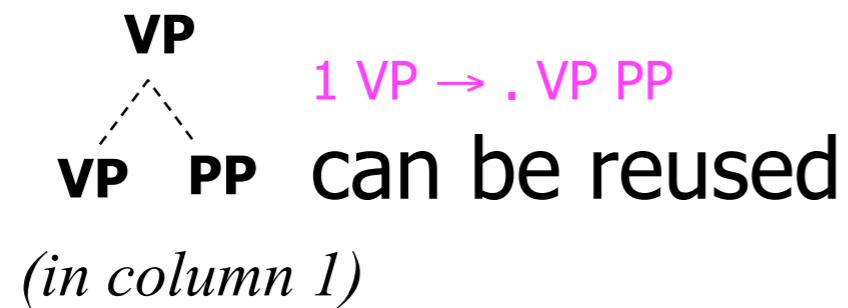
## ... but Earley's Alg is Okay!



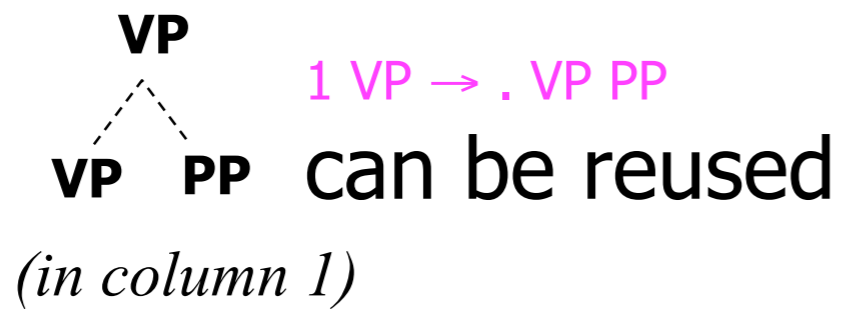
## ... but Earley's Alg is Okay!



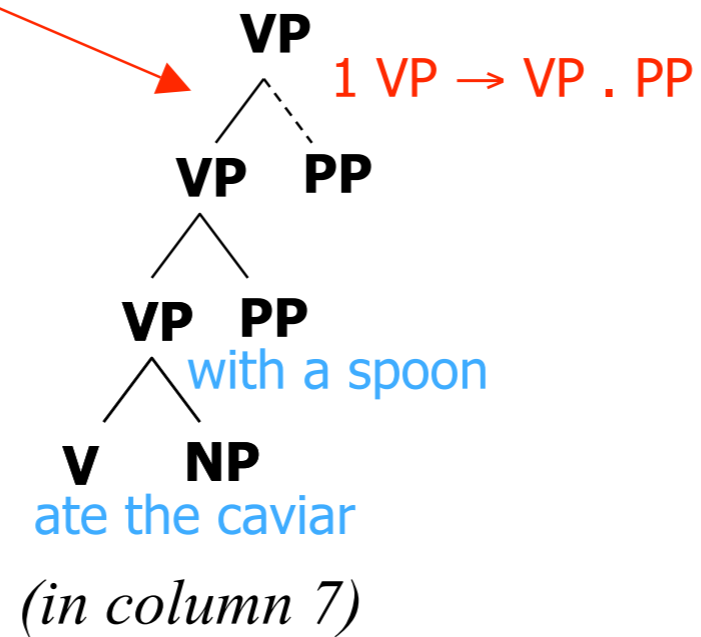
## ... but Earley's Alg is Okay!



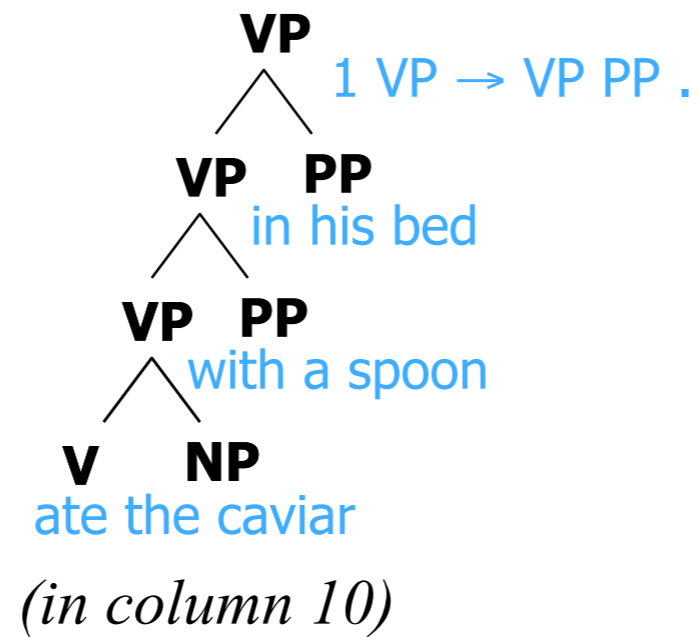
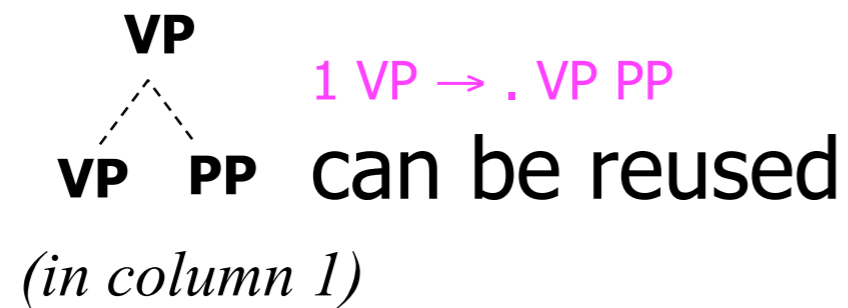
## ... but Earley's Alg is Okay!



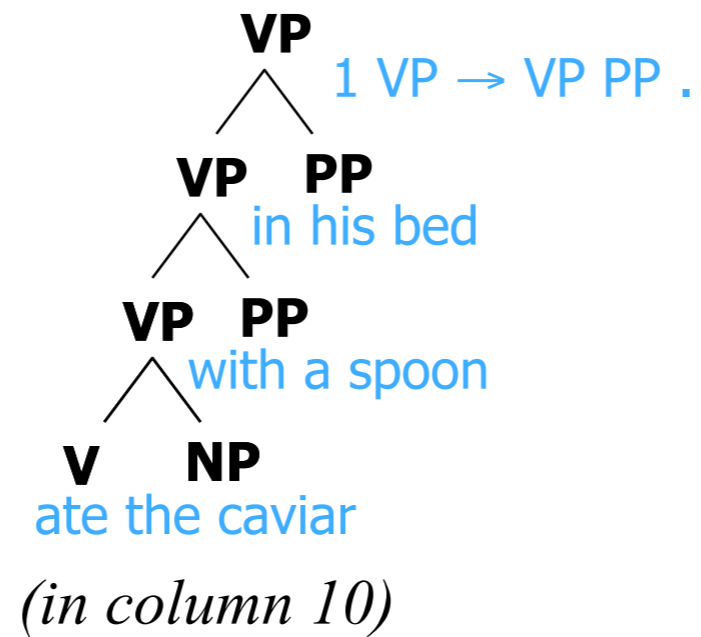
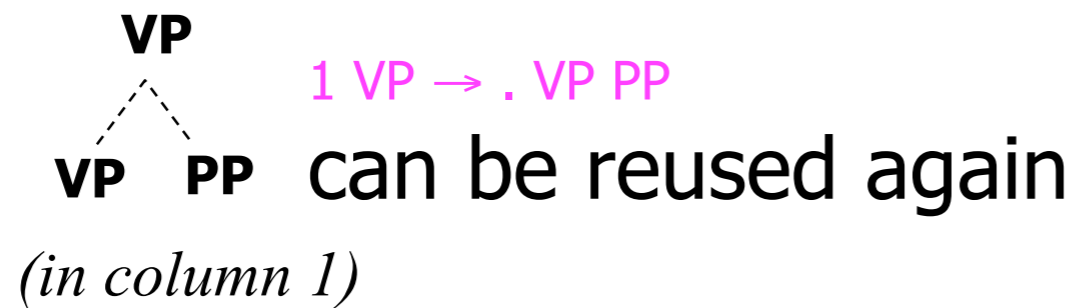
**attach**



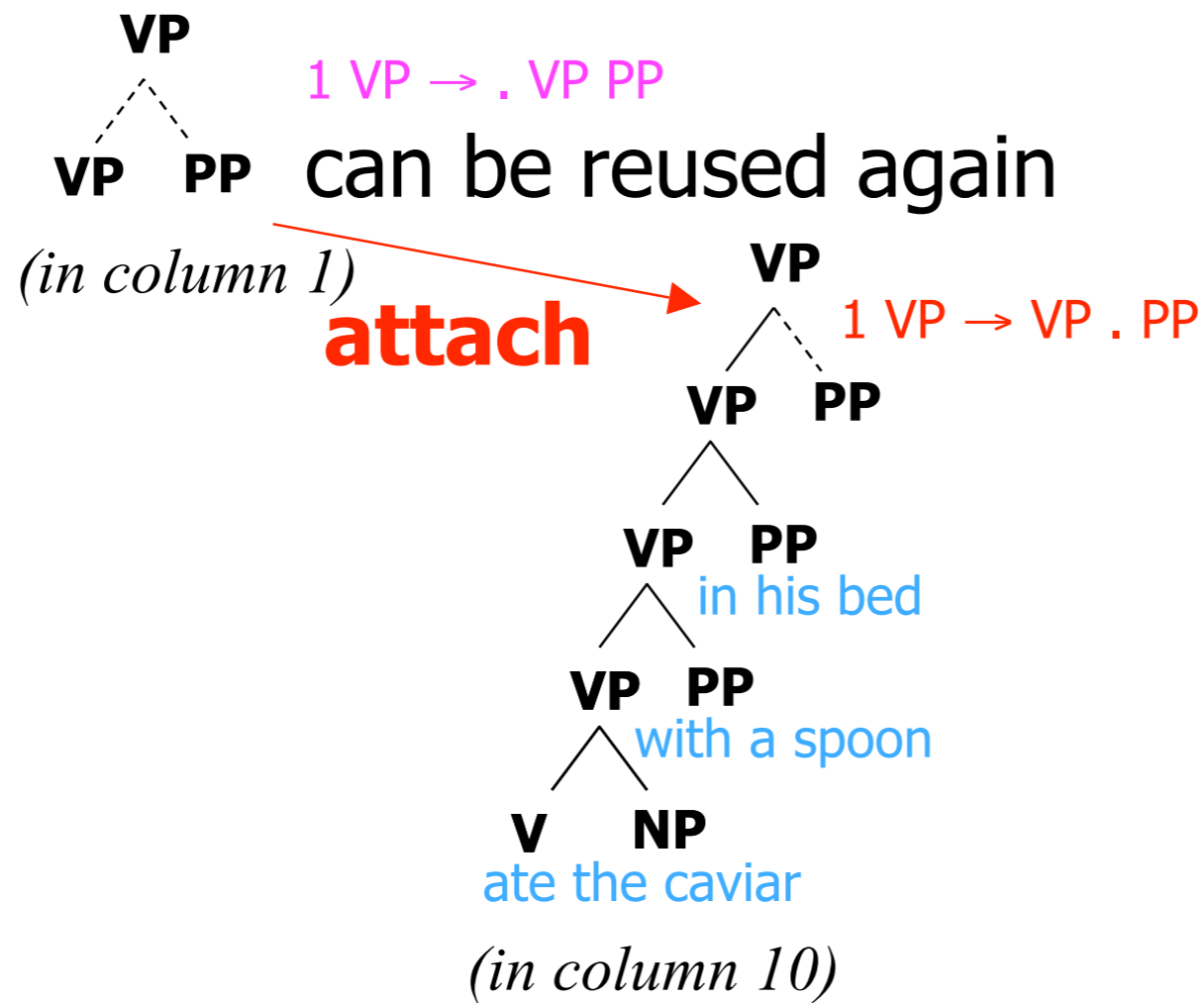
## ... but Earley's Alg is Okay!



## ... but Earley's Alg is Okay!



## ... but Earley's Alg is Okay!





0	Papa	1	ate	2	the	3	caviar	4	with a spoon	7
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP				
	1 P . with			0 ROOT S .		1 VP V NP .				
				4 P . with		2 NP NP . PP				
						0 S NP VP .				
						1 VP VP . PP				
						7 P . with				
						0 ROOT S .				

completed a VP in col 4  
col 1 lets us use it in a VP PP structure

0	Papa	1	ate	2	the	3	caviar	4	with a spoon	7
0 ROOT . S	0 NP Papa .	1 V ate .	2 Det the .	3 N caviar .	...	6 N spoon .				
0 S . NP VP	0 S NP . VP	1 VP V . NP	2 NP Det . N	2 NP Det N .		5 NP Det N .				
0 NP . Det N	0 NP NP . PP	2 NP . Det N	3 N . caviar	1 VP V NP .		4 PP P NP .				
0 NP . NP PP	1 VP . V NP	2 NP . NP PP	3 N . spoon	2 NP NP . PP		5 NP NP . PP				
0 NP . Papa	1 VP . VP PP	2 NP . Papa		0 S NP VP .		2 NP NP PP .				
0 Det . the	1 PP . P NP	2 Det . the		1 VP VP . PP		1 VP VP PP .				
0 Det . a	1 V . ate	2 Det . a		4 PP . P NP		7 PP . P NP				
	1 P . with			0 ROOT S .		1 VP V NP .				
				4 P . with		2 NP NP . PP				
						0 S NP VP .				
						1 VP VP . PP				
						7 P . with				
						0 ROOT S .				

completed that VP = VP PP in col 7  
col 1 would let us use *it* in a VP PP structure  
can reuse col 1 as often as we need

# Beyond Recognition

- So far, we've described an Earley *recognizer*
- Note what we did when we tried to create entries that already existed
- What should we do when combining items?
- How to derive outside algorithm?

# Parsing Tricks

# Left-Corner Parsing

- Technique for 1 word of lookahead in algorithms like Earley's
- (can also do multi-word lookahead but it's harder)

# Basic Earley's Algorithm

0	Papa	1
0 ROOT . S	0 NP Papa .	
0 S . NP VP	0 S NP . VP	
0 NP . Det N	0 NP NP . PP	
0 NP . NP PP		
0 NP . Papa		
0 Det . the		
0 Det . a		

**attach**

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		
0 Det . a		

**predict**

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		

**predict**



0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate
		1 V . drank
		1 V . snorted

**predict**

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate
		1 V . drank
		1 V . snorted

**predict**

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate
		1 V . drank
		1 V . snorted

## predict

- **.V** makes us add all the verbs in the vocabulary!
- **Slow** – we'd like a shortcut.

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate
		1 V . drank
		1 V . snorted

**predict**

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate
		1 V . drank
		1 V . snorted

**predict**

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate
		1 V . drank
		1 V . snorted

## predict

- Every .VP adds all VP → ... rules again.
- Before adding a rule, check it's not a duplicate.
- **Slow** if there are > 700 VP → ... rules, so what will you do in Homework 3?

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate
		1 V . drank
		1 V . snorted
		1 P . with

**predict**

0	Papa	1
0 ROOT . S		0 NP Papa .
0 S . NP VP		0 S NP . VP
0 NP . Det N		0 NP NP . PP
0 NP . NP PP		1 VP . V NP
0 NP . Papa		1 VP . VP PP
0 Det . the		1 PP . P NP
0 Det . a		1 V . ate
		1 V . drank
		1 V . snorted
		1 P . with

## predict

- .P makes us add all the prepositions ...



# 1-word lookahead would help

0	Papa	1	ate
0 ROOT . S		0 NP Papa .	
0 S . NP VP		0 S NP . VP	
0 NP . Det N		0 NP NP . PP	
0 NP . NP PP		1 VP . V NP	
0 NP . Papa		1 VP . VP PP	
0 Det . the		1 PP . P NP	
0 Det . a		1 V . ate	
		1 V . drank	
		1 V . snorted	
		1 P . with	

# 1-word lookahead would help

0	Papa	1	ate
0 ROOT . S		0 NP Papa .	
0 S . NP VP		0 S NP . VP	
0 NP . Det N		0 NP NP . PP	
0 NP . NP PP		1 VP . V NP	
0 NP . Papa		1 VP . VP PP	
0 Det . the		1 PP . P NP	
0 Det . a		1 V . ate	
		<del>1 V . drank</del>	
		<del>1 V . snorted</del>	
		<del>1 P . with</del>	

No point in adding words other than ate

# 1-word lookahead would help

0	Papa	1	ate
0 ROOT . S		0 NP Papa .	
0 S . NP VP		0 S NP . VP	
0 NP . Det N		<del>0 NP NP . PP</del>	
0 NP . NP PP		1 VP . V NP	
0 NP . Papa		1 VP . VP PP	
0 Det . the		<del>1 PP . P NP</del>	
0 Det . a		1 V . ate	
		<del>1 V . drank</del>	
		<del>1 V . snorted</del>	
		<del>1 P . with</del>	

In fact, no point in adding any constituent that can't start with ate  
 Don't bother adding PP, P, etc.

No point in adding words other than ate

# With Left-Corner Filter

0	Papa	1	ate
0 ROOT . S	0 NP Papa .		
0 S . NP VP	0 S NP . VP		
0 NP . Det N	<del>0 NP NP . PP</del>		
0 NP . NP PP			
0 NP . Papa			
0 Det . the			
0 Det . a			

**attach**

# With Left-Corner Filter

0	Papa	1	ate
0 ROOT . S	0 NP Papa .		
0 S . NP VP	0 S NP . VP		
0 NP . Det N	<del>0 NP NP . PP</del>		
0 NP . NP PP			
0 NP . Papa			
0 Det . the			
0 Det . a			

**attach**

PP can't start with ate

# With Left-Corner Filter

0	Papa	1	ate
0 ROOT . S	0 NP Papa .		
0 S . NP VP	0 S NP . VP		
0 NP . Det N	<del>0 NP NP . PP</del>		
0 NP . NP PP			
0 NP . Papa			
0 Det . the			
0 Det . a			

**attach**

PP can't start with ate

Pruning— now we won't predict

1 PP . P NP

1 PP . ate

either!

# With Left-Corner Filter

0	Papa	1	ate
0 ROOT . S	0 NP Papa .		
0 S . NP VP	0 S NP . VP		
0 NP . Det N	<del>0 NP NP . PP</del>		
0 NP . NP PP			
0 NP . Papa			
0 Det . the			
0 Det . a			

**attach**

PP can't start with ate

Pruning— now we won't predict

1 PP . P NP

1 PP . ate

either!

Need to know that ate can't start PP

# With Left-Corner Filter

0	Papa	1	ate
0 ROOT . S	0 NP Papa .		
0 S . NP VP	0 S NP . VP		
0 NP . Det N	<del>0 NP NP . PP</del>		
0 NP . NP PP			
0 NP . Papa			
0 Det . the			
0 Det . a			

**attach**

PP can't start with ate

Pruning— now we won't predict

1 PP . P NP

1 PP . ate

either!

Need to know that ate can't start PP

Take closure of all categories that it does

start ...



0	Papa	1	ate
0 ROOT . S		0 NP Papa .	
0 S . NP VP		0 S NP . VP	
0 NP . Det N		<del>0 NP NP . PP</del>	
0 NP . NP PP		1 VP . V NP	
0 NP . Papa		1 VP . VP PP	
0 Det . the			
0 Det . a			

**predict**

0	Papa	1	ate
0 ROOT . S	0 NP Papa .		
0 S . NP VP	0 S NP . VP		
0 NP . Det N	<del>0 NP NP . PP</del>		
0 NP . NP PP	1 VP . V NP		
0 NP . Papa	1 VP . VP PP		
0 Det . the	1 V . ate		
0 Det . a	<del>1 V . drank</del>		
	<del>1 V . snorted</del>		

**predict**

0	Papa	1	ate
0 ROOT . S	0 NP Papa .		
0 S . NP VP	0 S NP . VP		
0 NP . Det N	<del>0 NP NP . PP</del>		
0 NP . NP PP	1 VP . V NP		
0 NP . Papa	1 VP . VP PP		
0 Det . the	1 V . ate		
0 Det . a	<del>1 V . drank</del>		
	<del>1 V . snorted</del>		

**predict**

0	Papa	1	ate
0 ROOT . S	0 NP Papa .		
0 S . NP VP	0 S NP . VP		
0 NP . Det N	<del>0 NP NP . PP</del>		
0 NP . NP PP	1 VP . V NP		
0 NP . Papa	1 VP . VP PP		
0 Det . the	1 V . ate		
0 Det . a	<del>1 V . drank</del>		
	<del>1 V . snorted</del>		

**predict**

# Merging Right-Hand Sides

- Grammar might have rules

$X \rightarrow A G H P$

$X \rightarrow B G H P$

- Could end up with both of these in chart:

$(2, X \rightarrow A . G H P)$  in column 5

$(2, X \rightarrow B . G H P)$  in column 5

- But these are now interchangeable: if one produces  $X$  then so will the other
- To avoid this redundancy, can always use dotted rules of this form:  $X \rightarrow \dots G H P$

# Merging Right-Hand Sides

- Similarly, grammar might have rules
$$X \rightarrow A G H P$$
$$X \rightarrow A G H Q$$
- Could end up with both of these in chart:
$$(2, X \rightarrow A . G H P) \text{ in column 5}$$
$$(2, X \rightarrow A . G H Q) \text{ in column 5}$$
- Not interchangeable, but we'll be processing them in parallel for a while ...
- Solution: write grammar as  $X \rightarrow A G H (P|Q)$

# Merging Right-Hand Sides

- Combining the two previous cases:

**$X \rightarrow A G H P$**

**$X \rightarrow A G H Q$**

**$X \rightarrow B G H P$**

**$X \rightarrow B G H Q$**

becomes

**$X \rightarrow (A \mid B) G H (P \mid Q)$**

- And often nice to write stuff like

**$NP \rightarrow (Det \mid \varepsilon) Adj^* N$**

# Merging Right-Hand Sides



# Merging Right-Hand Sides

$X \rightarrow (A \mid B) G H (P \mid Q)$

$NP \rightarrow (\text{Det} \mid \varepsilon) \text{Adj}^* N$

# Merging Right-Hand Sides

$X \rightarrow (A \mid B) G H (P \mid Q)$

$NP \rightarrow (\text{Det} \mid \varepsilon) \text{Adj}^* N$

- These are regular expressions!

# Merging Right-Hand Sides

$X \rightarrow (A \mid B) G H (P \mid Q)$

$NP \rightarrow (\text{Det} \mid \varepsilon) \text{Adj}^* N$

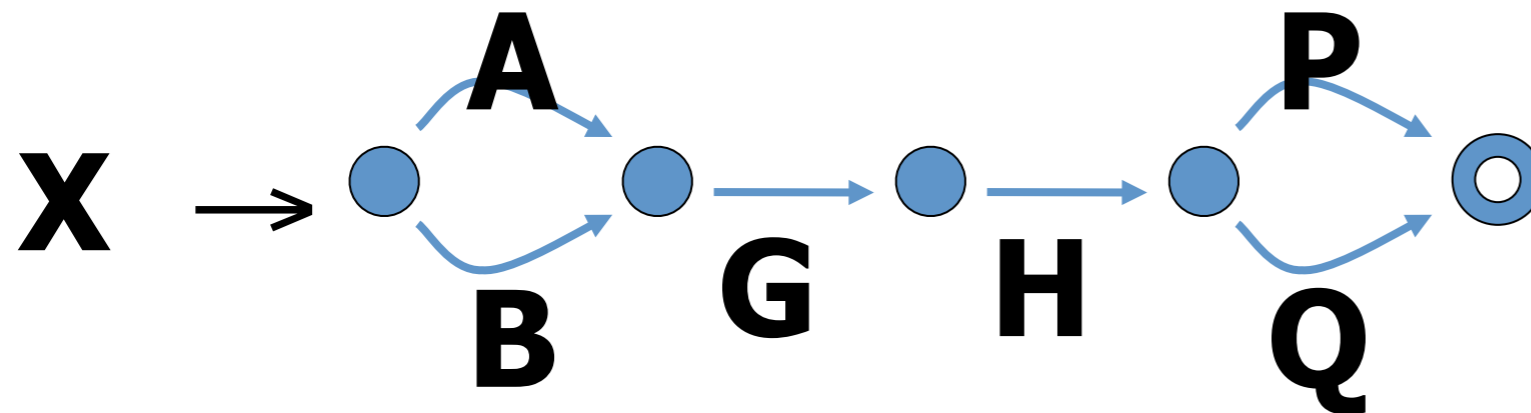
- These are regular expressions!
- Build their minimal DFAs:

# Merging Right-Hand Sides

$X \rightarrow (A \mid B) G H (P \mid Q)$

$NP \rightarrow (\text{Det} \mid \varepsilon) \text{Adj}^* N$

- These are regular expressions!
- Build their minimal DFAs:

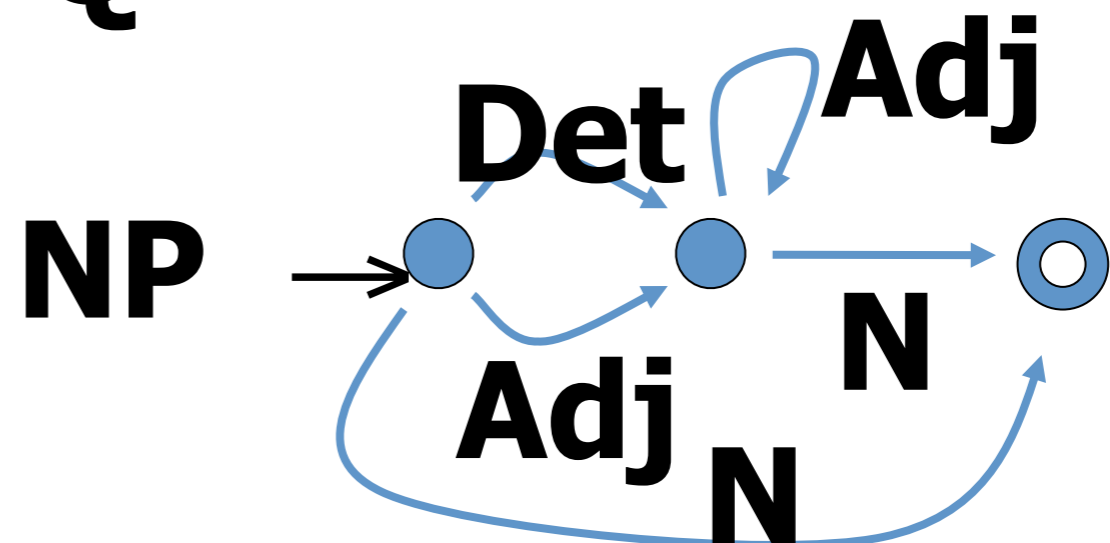
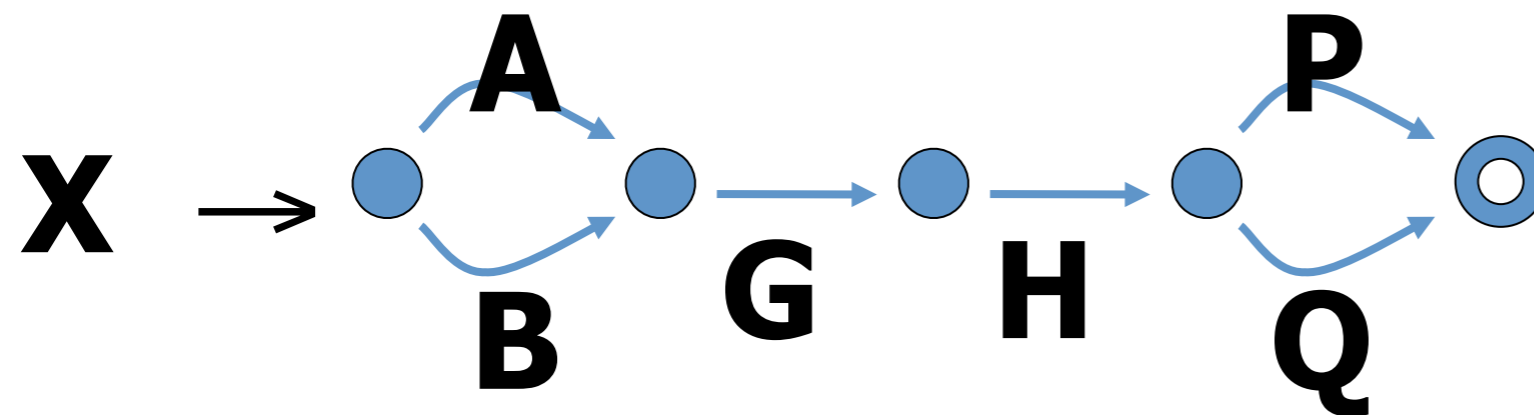


# Merging Right-Hand Sides

$X \rightarrow (A \mid B) G H (P \mid Q)$

$NP \rightarrow (Det \mid \varepsilon) Adj^* N$

- These are regular expressions!
- Build their minimal DFAs:

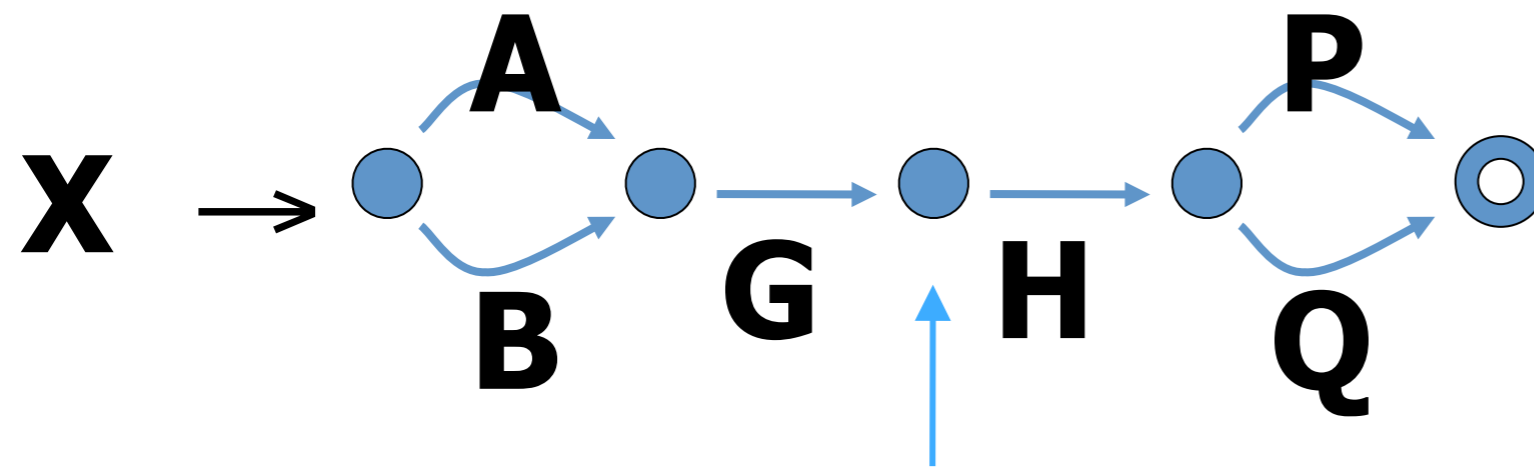


# Merging Right-Hand Sides

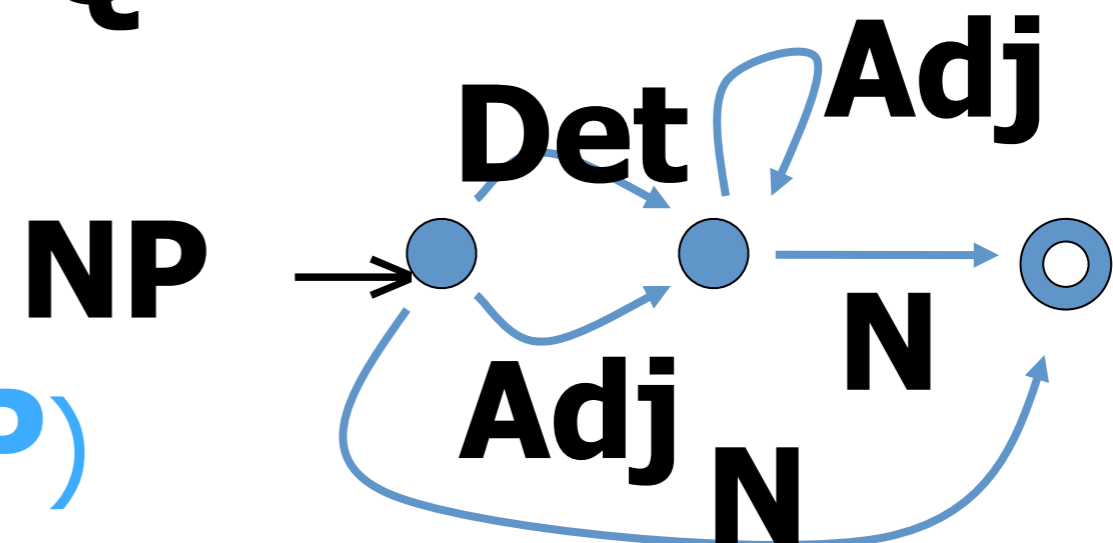
$X \rightarrow (A \mid B) G H (P \mid Q)$

$NP \rightarrow (Det \mid \epsilon) Adj^* N$

- These are regular expressions!
- Build their minimal DFAs:



- Automaton states replace dotted rules ( $X \rightarrow A G . H P$ )



# Merging Right-Hand Sides

Indeed, *all* **NP** → rules can be unioned into a single DFA!

NP → ADJP ADJP JJ JJ NN NNS

NP → ADJP DT NN

NP → ADJP JJ NN

NP → ADJP JJ NN NNS

NP → ADJP JJ NNS

NP → ADJP NN

NP → ADJP NN NN

NP → ADJP NN NNS

NP → ADJP NNS

NP → ADJP NPR

NP → ADJP NPRS

NP → DT

NP → DT ADJP

NP → DT ADJP , JJ NN

NP → DT ADJP ADJP NN

NP → DT ADJP JJ JJ NN

NP → DT ADJP JJ NN

NP → DT ADJP JJ NN NN

etc

# Merging Right-Hand Sides

Indeed, *all* **NP** → rules can be unioned into a single DFA!

NP → ADJP ADJP JJ JJ NN NNS

| ADJP DT NN

| ADJP JJ NN

| ADJP JJ NN NNS

| ADJP JJ NNS

| ADJP NN

| ADJP NN NN

| ADJP NN NNS

| ADJP NNS

| ADJP NPR

| ADJP NPRS

| DT

| DT ADJP

| DT ADJP , JJ NN

| DT ADJP ADJP NN

| DT ADJP JJ JJ NN

| DT ADJP JJ NN

| DT ADJP JJ NN NN

**etc.**



# Merging Right-Hand Sides

Indeed, *all* **NP** → rules can be unioned into a single DFA!

NP → ADJP ADJP JJ JJ NN NNS

| ADJP DT NN

| ADJP JJ NN

| ADJP JJ NN NNS

| ADJP JJ NNS

| ADJP NN

| ADJP NN NN

| ADJP NN NNS

| ADJP NNS

| ADJP NPR

| ADJP NPRS

| DT

| DT ADJP

| DT ADJP , JJ NN

| DT ADJP ADJP NN

| DT ADJP JJ JJ NN

| DT ADJP JJ NN

| DT ADJP JJ NN NN

**etc.**

regular  
expression

# Merging Right-Hand Sides

Indeed, *all* **NP**  $\rightarrow$  rules can be unioned into a single DFA!

NP  $\rightarrow$  ADJP ADJP JJ JJ NN NNS

| ADJP DT NN

| ADJP JJ NN

| ADJP JJ NN NNS

| ADJP JJ NNS

| ADJP NN

| ADJP NN NN

| ADJP NN NNS

| ADJP NNS

| ADJP NPR

| ADJP NPRS

| DT

| DT ADJP

| DT ADJP , JJ NN

| DT ADJP ADJP NN

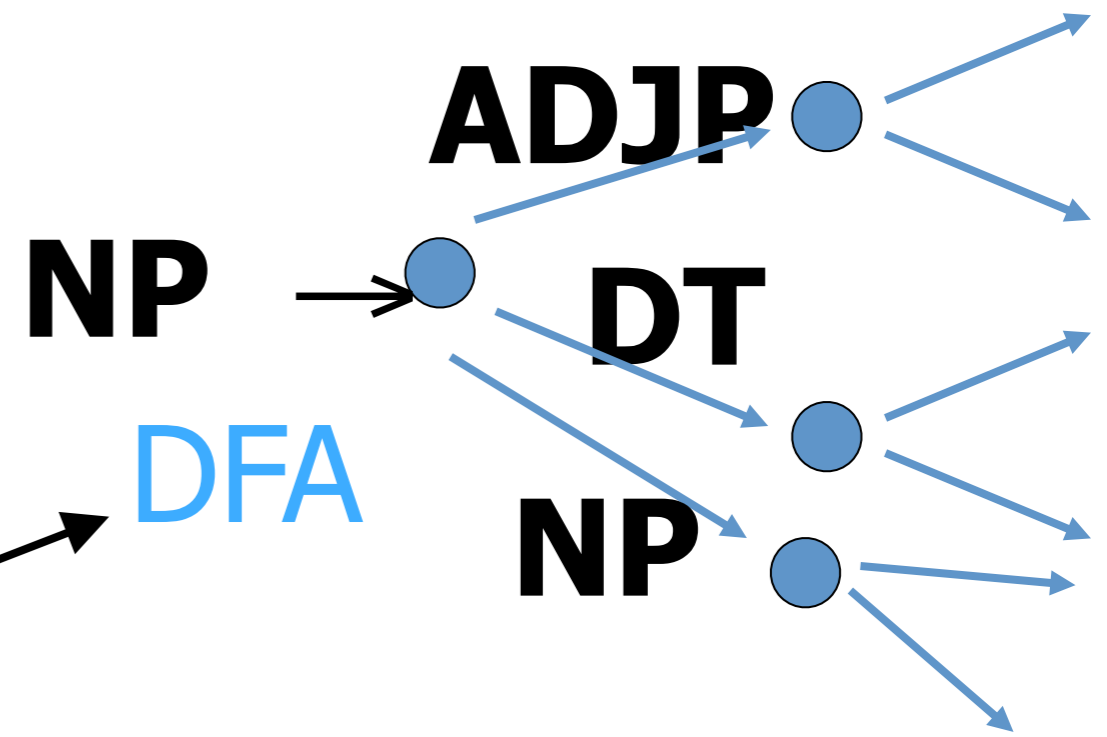
| DT ADJP JJ JJ NN

| DT ADJP JJ NN

| DT ADJP JJ NN NN

**etc.**

regular  
expression



# Merging Right-Hand Sides

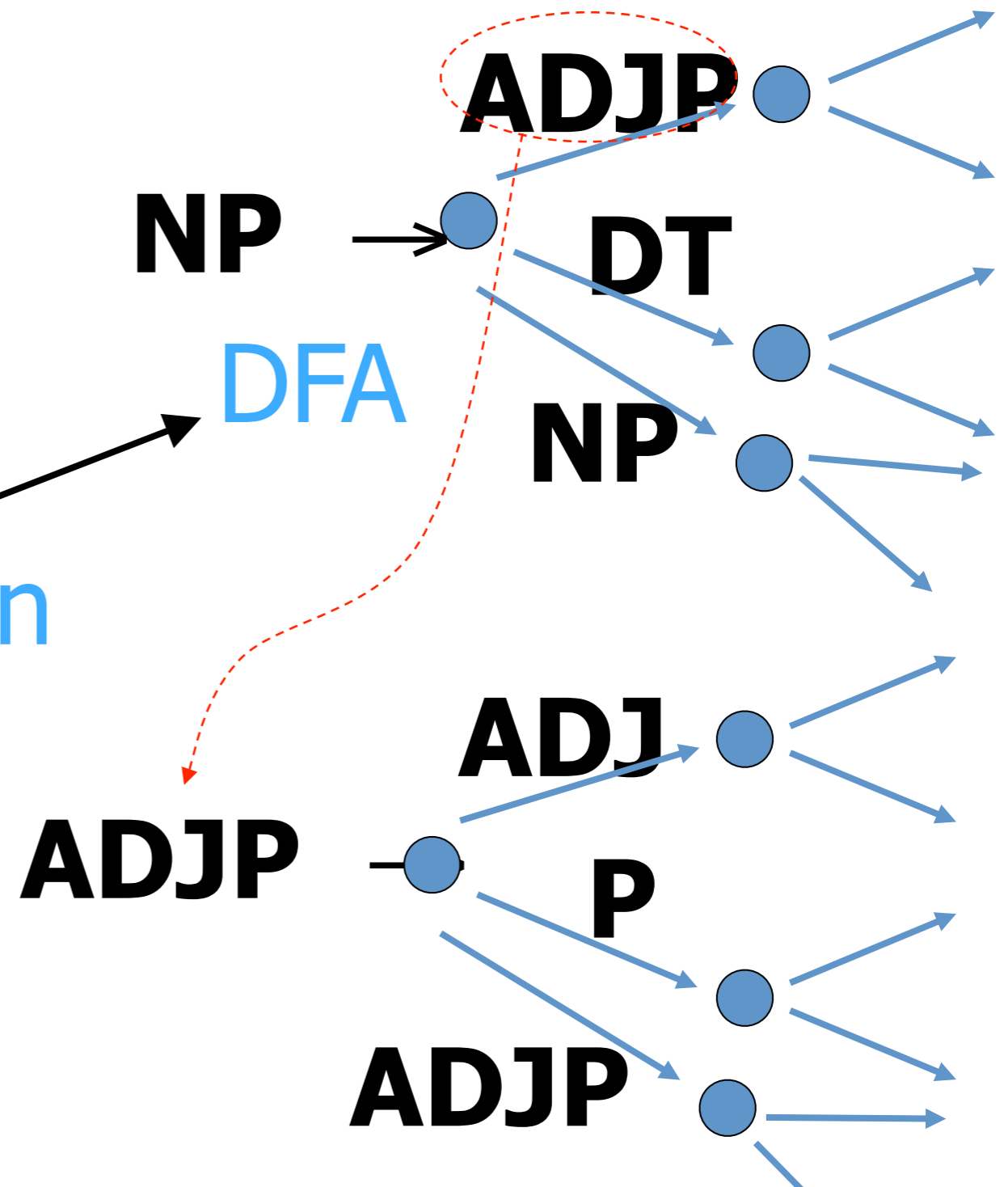
Indeed, *all* **NP**  $\rightarrow$  rules can be unioned into a single DFA!

NP  $\rightarrow$  ADJP ADJP JJ JJ NN NNS

- | ADJP DT NN
- | ADJP JJ NN
- | ADJP JJ NN NNS
- | ADJP JJ NNS
- | ADJP NN
- | ADJP NN NN
- | ADJP NN NNS
- | ADJP NNS
- | ADJP NPR
- | ADJP NPRS
- | DT
- | DT ADJP
- | DT ADJP , JJ NN
- | DT ADJP ADJP NN
- | DT ADJP JJ JJ NN
- | DT ADJP JJ NN
- | DT ADJP JJ NN NN

**etc.**

regular  
expression



# Merging Right-Hand Sides

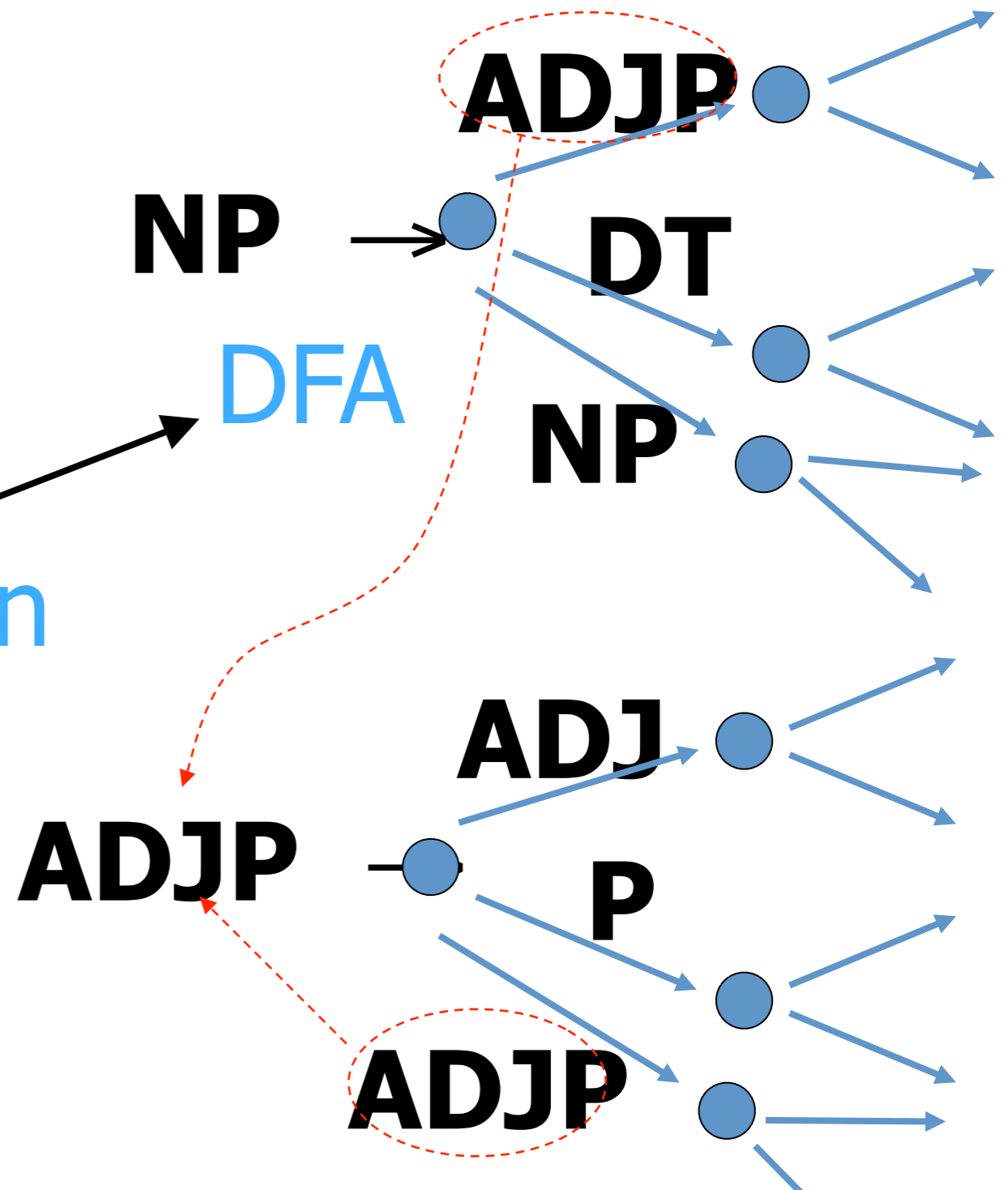
Indeed, *all* **NP**  $\rightarrow$  rules can be unioned into a single DFA!

NP  $\rightarrow$  ADJP ADJP JJ JJ NN NNS

- | ADJP DT NN
- | ADJP JJ NN
- | ADJP JJ NN NNS
- | ADJP JJ NNS
- | ADJP NN
- | ADJP NN NN
- | ADJP NN NNS
- | ADJP NNS
- | ADJP NPR
- | ADJP NPRS
- | DT
- | DT ADJP
- | DT ADJP , JJ NN
- | DT ADJP ADJP NN
- | DT ADJP JJ JJ NN
- | DT ADJP JJ NN
- | DT ADJP JJ NN NN

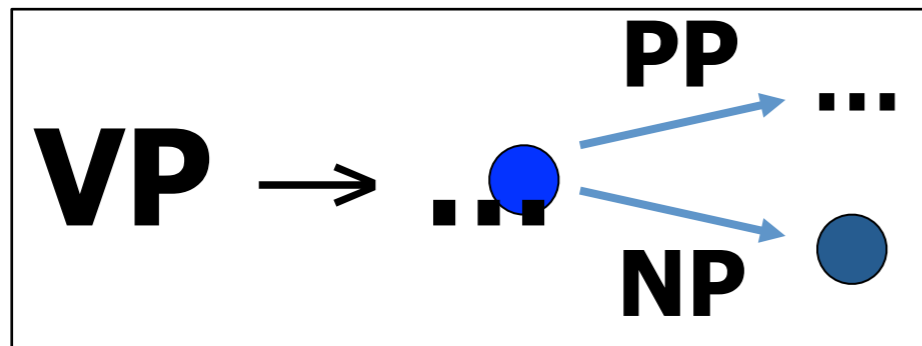
**etc.**

regular  
expression



# Earley's Algorithm on DFAs

- What does Earley's algorithm now look like?

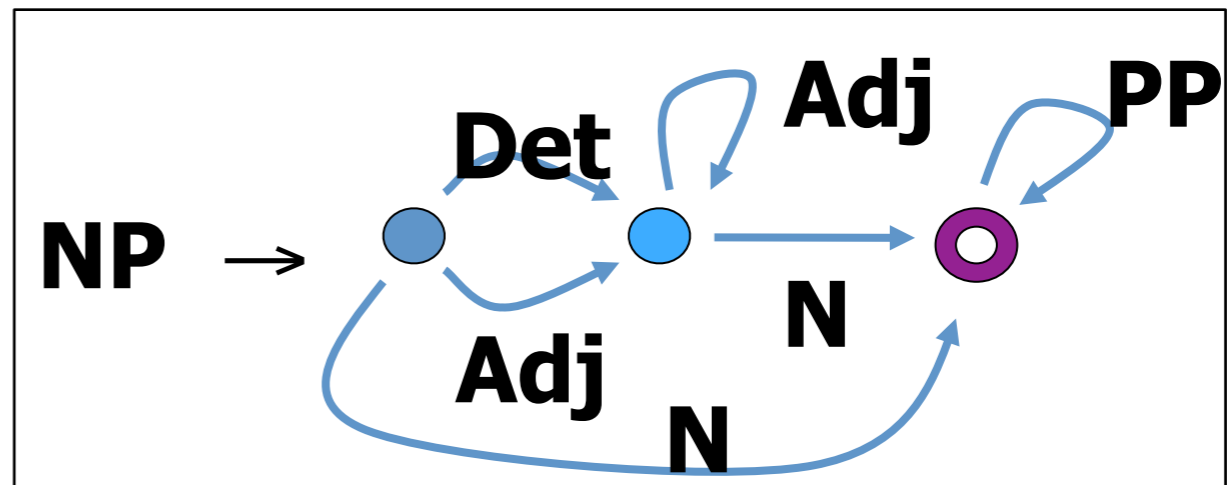
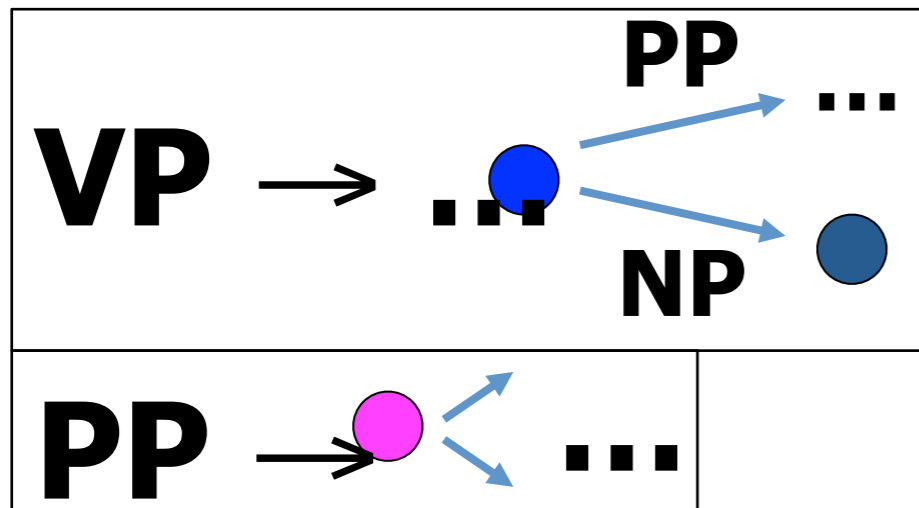


Column 4
...
(2, ●)

**predict**

# Earley's Algorithm on DFAs

- What does Earley's algorithm now look like?

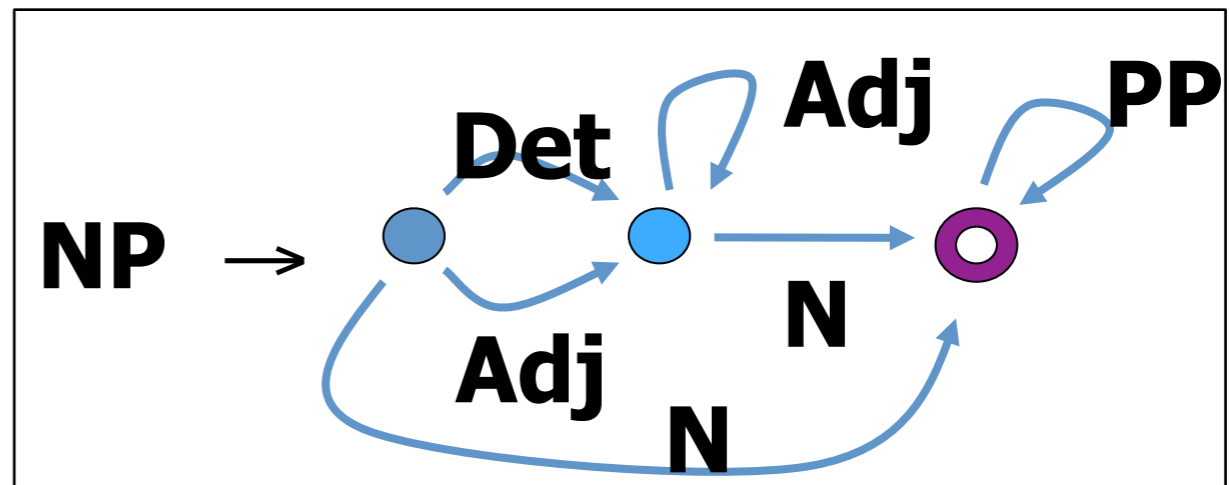
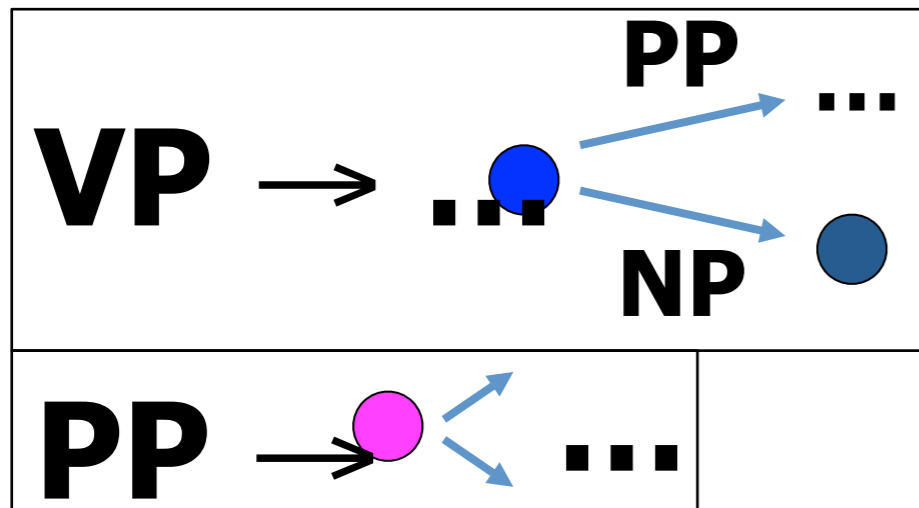


Column 4
...
(2, ●)
(4, ●)
(4, ●)

**predict**

# Earley's Algorithm on DFAs

- What does Earley's algorithm now look like?

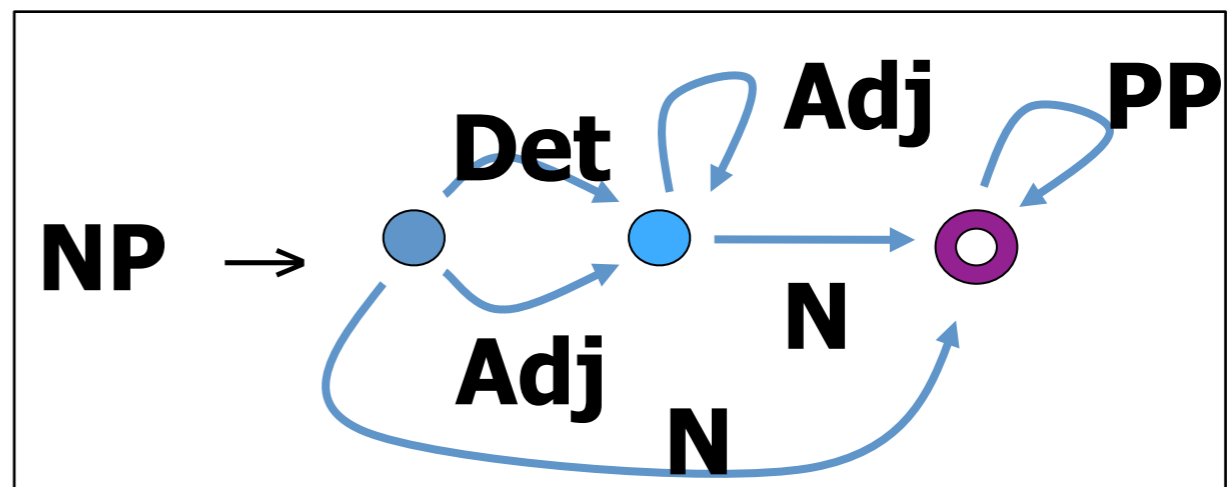
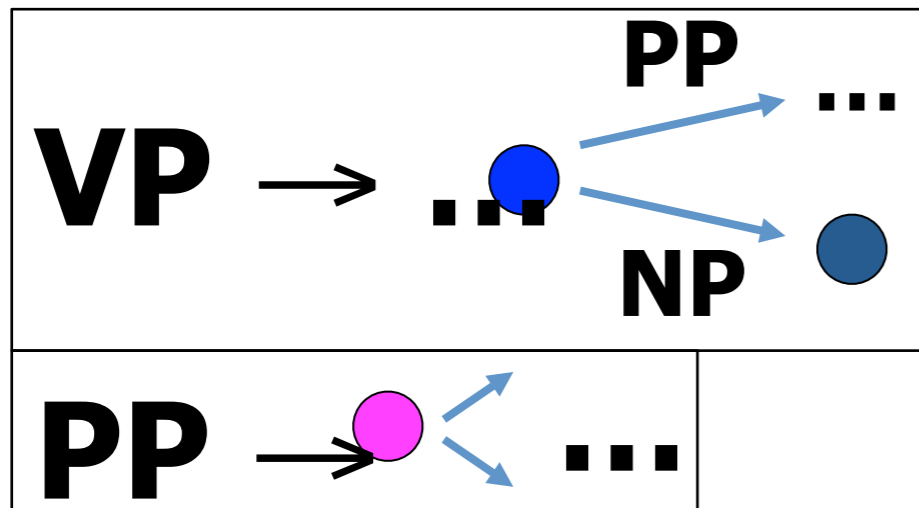


Column 4	Column 5	...	Column 7
...	...		
(2, ●)			(4, ●)
(4, ●)			
(4, ●) ----- (4, ●)			

**predict  
or attach?**

# Earley's Algorithm on DFAs

- What does Earley's algorithm now look like?



Column 4	Column 5	...	Column 7
...	...		
(2, ●)			(4, ○)
(4, ●)			
(4, ●) ----- (4, ●)			

**predict  
or attach?  
Both!**



# Pruning for Speed

# Pruning for Speed

- **Heuristically throw away** constituents that probably won't make it into best complete parse.

# Pruning for Speed

- **Heuristically throw away** constituents that probably won't make it into best complete parse.
- Use **probabilities** to decide which ones.

# Pruning for Speed

- **Heuristically throw away** constituents that probably won't make it into best complete parse.
- Use **probabilities** to decide which ones.
  - So probs are useful for speed as well as accuracy!

# Pruning for Speed

- **Heuristically throw away** constituents that probably won't make it into best complete parse.
- Use **probabilities** to decide which ones.
  - So probs are useful for speed as well as accuracy!
- **Both safe and unsafe methods exist**

# Pruning for Speed

- **Heuristically throw away** constituents that probably won't make it into best complete parse.
- Use **probabilities** to decide which ones.
  - So probs are useful for speed as well as accuracy!
- **Both safe and unsafe methods exist**
  - Throw  $x$  away if  $p(x) < 10^{-200}$   
(and lower this threshold if we don't get a parse)

# Pruning for Speed

- **Heuristically throw away** constituents that probably won't make it into best complete parse.
- Use **probabilities** to decide which ones.
  - So probs are useful for speed as well as accuracy!
- **Both safe and unsafe methods exist**
  - Throw  $x$  away if  $p(x) < 10^{-200}$   
(and lower this threshold if we don't get a parse)
  - Throw  $x$  away if  $p(x) < 100 * p(y)$   
for some  $y$  that spans the same set of words

# Pruning for Speed

- **Heuristically throw away** constituents that probably won't make it into best complete parse.
- Use **probabilities** to decide which ones.
  - So probs are useful for speed as well as accuracy!
- **Both safe and unsafe methods exist**
  - Throw  $x$  away if  $p(x) < 10^{-200}$   
(and lower this threshold if we don't get a parse)
  - Throw  $x$  away if  $p(x) < 100 * p(y)$   
for some  $y$  that spans the same set of words
  - Throw  $x$  away if  $p(x) * q(x)$  is small, where  $q(x)$  is an estimate of probability of all rules needed to combine  $x$  with the other words in the sentence



# Agenda (“Best-First”) Parsing

# Agenda (“Best-First”) Parsing

- Explore best options first
  - Should get some good parses early on – grab one & go!

# Agenda (“Best-First”) Parsing

- Explore best options first
  - Should get some good parses early on – grab one & go!
- Prioritize constits (and dotted constits)
  - Whenever we build something, give it a priority
    - How likely do we think it is to make it into the highest-prob parse?
  - usually related to log prob. of that constit
  - might also hack in the constit’s context, length, etc.
  - if priorities are defined carefully, obtain an A\* algorithm

# Agenda (“Best-First”) Parsing

- Explore best options first
  - Should get some good parses early on – grab one & go!
- Prioritize constits (and dotted constits)
  - Whenever we build something, give it a priority
    - How likely do we think it is to make it into the highest-prob parse?
  - usually related to log prob. of that constit
  - might also hack in the constit’s context, length, etc.
  - if priorities are defined carefully, obtain an A\* algorithm
- Put each constit on a priority queue (heap)

# Agenda (“Best-First”) Parsing

- Explore best options first
  - Should get some good parses early on – grab one & go!
- Prioritize constits (and dotted constits)
  - Whenever we build something, give it a priority
    - How likely do we think it is to make it into the highest-prob parse?
  - usually related to log prob. of that constit
  - might also hack in the constit’s context, length, etc.
  - if priorities are defined carefully, obtain an A\* algorithm
- Put each constit on a priority queue (heap)
- Repeatedly pop and process best constituent.
  - CKY style: combine w/ previously popped neighbors.
  - Earley style: scan/predict/attach as usual. What else?

# Preprocessing

# Preprocessing

- First “tag” the input with parts of speech:
  - Guess the correct preterminal for each word, using faster methods we’ll learn later
  - Now only allow one part of speech per word
  - This eliminates a lot of crazy constituents!
  - But if you tagged wrong you could be hosed
- Raise the stakes:
  - What if tag says not just “verb” but “transitive verb”? Or “verb with a direct object and 2 PPs attached”? (“supertagging”)

# Preprocessing

- First “tag” the input with parts of speech:
  - Guess the correct preterminal for each word, using faster methods we’ll learn later
  - Now only allow one part of speech per word
  - This eliminates a lot of crazy constituents!
  - But if you tagged wrong you could be hosed
- Raise the stakes:
  - What if tag says not just “verb” but “transitive verb”? Or “verb with a direct object and 2 PPs attached”? (“supertagging”)
- Safer to allow a few possible tags per word, not just one  
...



# Center-Embedding

```
if x
then
  if y
  then
    if a
    then b
    endif
  else b
  endif
else b
endif
```

# Center-Embedding

if x

then

if y

then

if a

then b

endif

else b

endif

else b

endif

STATEMENT  $\rightarrow$  if EXPR then

STATEMENT

endif

# Center-Embedding

if x

then

if y

then

if a

then b

endif

else b

endif

else b

endif

STATEMENT  $\rightarrow$  if EXPR then

STATEMENT

endif

STATEMENT  $\rightarrow$  if EXPR then STATEMENT

else STATEMENT endif

# Center-Embedding

if x

then

if y

then

if a

then b

endif

else b

endif

else b

endif

STATEMENT  $\rightarrow$  if EXPR then

STATEMENT

endif

STATEMENT  $\rightarrow$  if EXPR then STATEMENT

else STATEMENT endif

But not:

STATEMENT  $\rightarrow$  if EXPR then

STATEMENT

# Center-Embedding

# Center-Embedding

- This is the rat that ate the malt.

# Center-Embedding

- This is the rat that ate the malt.
- This is the malt that the rat ate.

# Center-Embedding

- This is the rat that ate the malt.
- This is the malt that the rat ate.



# Center-Embedding

- This is the rat that ate the malt.
- This is the malt that the rat ate.
- This is the cat that bit the rat that ate the malt.

# Center-Embedding

- This is the rat that ate the malt.
- This is the malt that the rat ate.
  
- This is the cat that bit the rat that ate the malt.
- This is the malt that the rat that the cat bit ate.

# Center-Embedding

- This is the rat that ate the malt.
- This is the malt that the rat ate.
  
- This is the cat that bit the rat that ate the malt.
- This is the malt that the rat that the cat bit ate.

# Center-Embedding

- This is the rat that ate the malt.
- This is the malt that the rat ate.
  
- This is the cat that bit the rat that ate the malt.
- This is the malt that the rat that the cat bit ate.
  
- This is the dog that chased the cat that bit the rat that ate the malt.

# Center-Embedding

- This is the rat that ate the malt.
- This is the malt that the rat ate.
  
- This is the cat that bit the rat that ate the malt.
- This is the malt that the rat that the cat bit ate.
  
- This is the dog that chased the cat that bit the rat that ate the malt.
- This is the malt that [the rat that [the cat that [the dog chased] bit] ate].

# More Center-Embedding

[What did you disguise  
[those handshakes that you  
greeted  
[the people we bought  
[the bench  
[Billy was read to]  
on]  
with]  
with]  
for]?

# More Center-Embedding

[What did you disguise  
[those handshakes that you  
greeted  
[the people we bought  
[the bench  
[Billy was read to]  
on]  
with]  
with]  
for]?

[Which mantelpiece did you put  
[the idol I sacrificed  
[the fellow we sold  
[the bridge you threw  
[the bench  
[Billy was read to]

# More Center-Embedding

[What did you disguise  
[those handshakes that you  
greeted  
[the people we bought  
[the bench  
[Billy was read to]  
on]  
with]  
with]  
for]?

[Which mantelpiece did you put  
[the idol I sacrificed  
[the fellow we sold  
[the bridge you threw  
[the bench  
[Billy was read to]  
on]



# More Center-Embedding

[What did you disguise  
[those handshakes that you  
greeted  
[the people we bought  
[the bench  
[Billy was read to]  
on]  
with]  
with]  
for]?

[Which mantelpiece did you put  
[the idol I sacrificed  
[the fellow we sold  
[the bridge you threw  
[the bench  
[Billy was read to]  
on]  
off]

# More Center-Embedding

[What did you disguise  
[those handshakes that you  
greeted  
[the people we bought  
[the bench  
[Billy was read to]  
on]  
with]  
with]  
for]?

[Which mantelpiece did you put  
[the idol I sacrificed  
[the fellow we sold  
[the bridge you threw  
[the bench  
[Billy was read to]  
on]  
off]  
to]

# More Center-Embedding

[What did you disguise  
[those handshakes that you  
greeted  
[the people we bought  
[the bench  
[Billy was read to]  
on]  
with]  
with]  
for]?

[Which mantelpiece did you put  
[the idol I sacrificed  
[the fellow we sold  
[the bridge you threw  
[the bench  
[Billy was read to]  
on]  
off]  
to]  
to]

# More Center-Embedding

[What did you disguise  
[those handshakes that you  
greeted  
[the people we bought  
[the bench  
[Billy was read to]  
on]  
with]  
with]  
for]?

[Which mantelpiece did you put  
[the idol I sacrificed  
[the fellow we sold  
[the bridge you threw  
[the bench  
[Billy was read to]  
on]  
off]  
to]  
to]  
on]?

# More Center-Embedding

[What did you disguise  
[those handshakes that you  
greeted

[the people we bought  
[the bench

[Billy was read to]

on]

with]

with]

for]?

[Which mantelpiece did you put  
[the idol I sacrificed

[the fellow we sold

[the bridge you threw

[the bench

[Billy was read to]

on]

off]

to]

on]?

**Take that,**

**English teachers!**

# Center Recursion vs. Tail Recursion



[What did you disguise  
[those handshakes that you  
greeted  
[the people we bought  
[the bench  
[Billy was read to]  
on]  
with]  
with]  
for]?



[For what did you disguise  
[those handshakes with which  
you greeted  
[the people with which we bought  
[the bench on which  
[Billy was read to]?

“pied piping” –  
NP moves leftward,  
preposition follows along

# Disallow Center-Embedding?

# Disallow Center-Embedding?

- Center-embedding seems to be in the grammar, but people have trouble processing more than 1 level of it.



# Disallow Center-Embedding?

- Center-embedding seems to be in the grammar, but people have trouble processing more than 1 level of it.
- You can limit # levels of center-embedding via features: e.g.,  
 $S[S\_DEPTH=n+1] \rightarrow A S[S\_DEPTH=n] B$

# Disallow Center-Embedding?

- Center-embedding seems to be in the grammar, but people have trouble processing more than 1 level of it.
- You can limit # levels of center-embedding via features: e.g.,  
 $S[S\_DEPTH=n+1] \rightarrow A S[S\_DEPTH=n] B$
- If a CFG limits # levels of embedding, then it can be compiled into a finite-state machine – we don't need a stack at all!
  - Finite-state recognizers run in linear time.
  - However, it's tricky to turn them into parsers for the original CFG from which the recognizer was compiled.

# Overview

- Treebanks and evaluation
- Lexicalized parsing (with heads)
  - Examples: Collins

# Treebanks

- \* Pure Grammar Induction Approaches tend not to produce the parse trees that people want
  
- \* Solution
  - ∅ Give a some example of parse trees that we want
  - ∅ Make a learning tool learn a grammar
  
- \* Treebank
  - ∅ A collection of such example parses
  - ∅ **PennTreebank** is most widely used

# Treebanks

- Penn Treebank
  - Trees are represented via **bracketing**
  - Fairly **flat structures** for Noun Phrases  
(NP Arizona real estate loans)
  - Tagged with **grammatical and semantic functions**  
(-SBJ , -LOC, ...)
  - Use empty nodes(\*) to indicate **understood subjects** and **extraction gaps**

( ( S ( NP-SBJ The move)  
   ( VP followed  
     ( NP ( NP a round )  
       ( PP of  
         ( NP ( NP similar increases )  
           ( PP by  
             ( NP other lenders ) )  
           ( PP against  
             ( NP Arizona real estate loans ) ) ) ) ) ) ) ) ) )  
       ' ( S-ADV ( NP-SBJ \* )  
         ( VP reflecting  
           ( NP a continuing decline )  
           ( PP-LOC in  
             ( NP that market ) ) ) ) ) ) ) )  
   . )

# Treebanks

- Many people have argued that **it is better to have linguists constructing treebanks** than grammars
- Because it is easier
  - to work out the correct parse of sentences
- than
  - to try to determine what **all possible manifestations** of a certain rule or grammatical construct are

# Parser Evaluation



# Evaluation

Ultimate goal is to build system for IE, QA, MT

People are rarely interested in syntactic analysis for its own sake

Evaluate the system for evaluate the parser

For Simplicity and modularization, and Convenience

Compare parses from a parser with the result of hand parsing of a sentence(gold standard)

What is objective criterion that we are trying to maximize?

# Evaluation

## Tree Accuracy (Exact match)

It is a very tough standard!!!

But in many ways it is a sensible one to use

## PARSEVAL Measures

For some purposes, partially correct parses can be useful

Originally for non-statistical parsers

Evaluate the component pieces of a parse

Measures : Precision, Recall, Crossing brackets

# Evaluation

## (Labeled) Precision

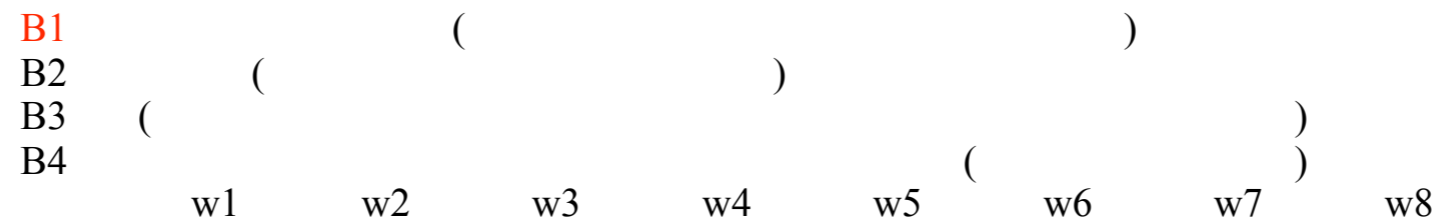
How many brackets in the parse match those in the correct tree (Gold standard)?

## (Labeled) Recall

How many of the brackets in the correct tree are in the parse?

## Crossing brackets

Average of how many constituents in one tree cross over constituent boundaries in the other tree



## Problems with PARSEVAL

Even vanilla PCFG performs quite well

It measures success at the level of individual decisions

You must make many consecutive decisions correctly to be correct on the entire tree.

## Problems with PARSEVAL (2)

### Behind story

The structure of Penn Treebank

Flat → Few brackets → Low Crossing brackets

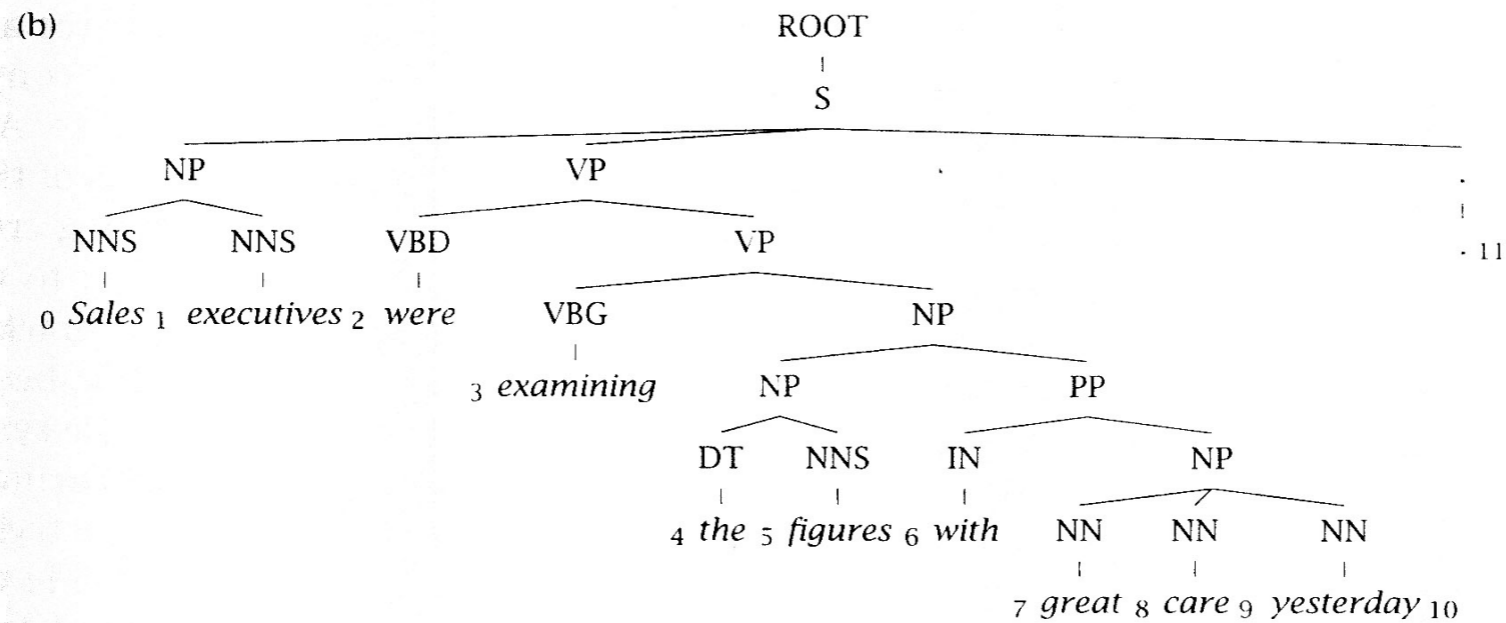
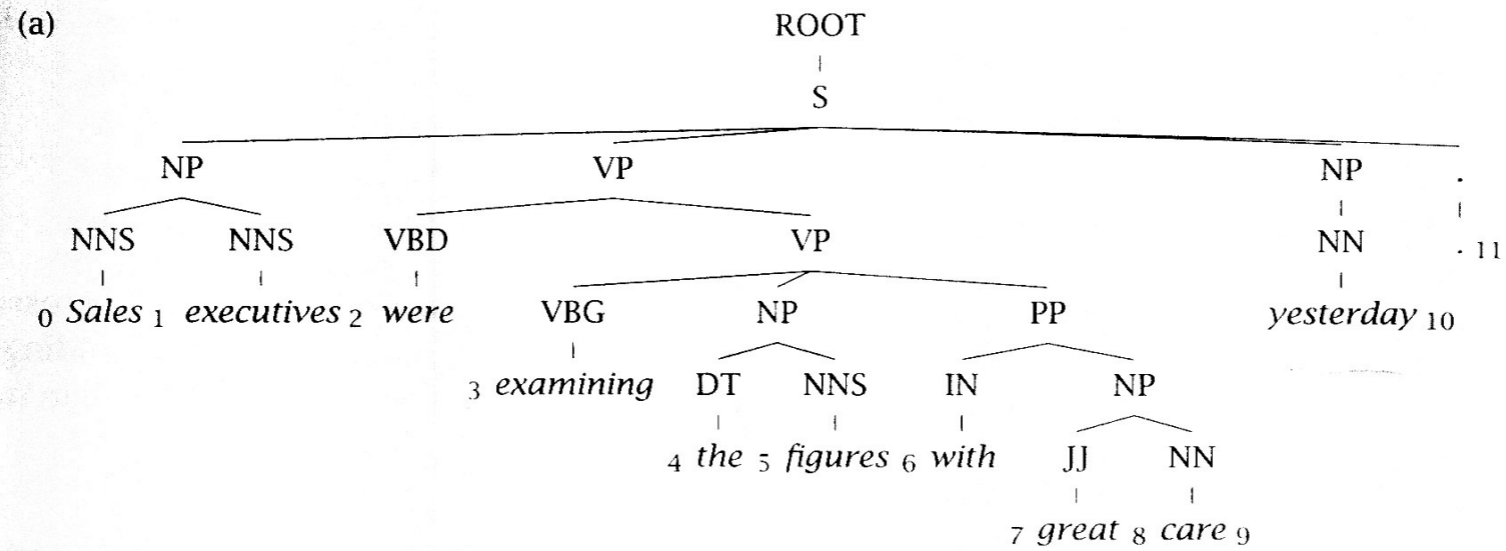
Troublesome brackets are avoided

→ High Precision/Recall

The errors in precision and recall are minimal

In some cases wrong PP attachment penalizes Precision, Recall and Crossing Bracket Accuracy minimally.

On the other hand, attaching low instead of high, then every node in the right-branching tree will be wrong: serious harm



(c) Brackets in gold standard tree (a.):

S-(0:11), NP-(0:2), VP-(2:9), VP-(3:9) NP-(4:6), PP-(6:9), NP-(7,9), \*NP-(9:10)

(d) Brackets in candidate parse (b.):

S-(0:11), NP-(0:2), VP-(2:10), VP-(3:10), NP-(4:10), NP-(4:6), PP-(6-10), NP-(7,10)

(e) Precision:  $3/8 = 37.5\%$  Crossing Brackets: 3  
 Recall:  $3/8 = 37.5\%$  Crossing Accuracy: 62%  
 Labeled Precision:  $3/8 = 37.5\%$  Tagging Accuracy:  $10/11 = 90.9\%$   
 Labeled Recall:  $3/8 = 37.5\%$

# Evaluation

Do PARSEVAL measures succeed in real tasks?

Many small parsing mistakes might not affect tasks of semantic interpretation

(Bonnema 1996,1997)

Tree Accuracy of the Parser : 62%

Correct Semantic Interpretations : 88%

(Hermajakob and Mooney 1997)

English to German translation

At the moment, people feel PARSEVAL measures are adequate for the comparing parsers

# Lexicalized Parsing



# Limitations of PCFGs

- PCFGs assume:
  - Place invariance
  - Context free:  $P(\text{rule})$  independent of
    - words outside span
    - *also, words with overlapping derivation*
  - Ancestor free:  $P(\text{rule})$  independent of
    - *Non-terminals above.*
- Lack of sensitivity to lexical information
- Lack of sensitivity to structural frequencies

# Lack of Lexical Dependency

Means that

$P(\text{VP} \rightarrow \text{V NP NP})$

is independent of the particular verb involved!

... but much more likely with ditransitive verbs (like ***gave***).

*He **gave** the boy a ball.*

*He **ran** to the store.*

# The Need for Lexical Dependency

Probabilities dependent on Lexical words

Problem 1 : Verb subcategorization

VP expansion is independent of the choice of verb

However ...

	verb			
	come	take	think	want
VP -> V	9.5%	2.6%	4.6%	5.7%
VP -> V NP	1.1%	32.1%	0.2%	13.9%
VP -> V PP	34.5%	3.1%	7.1%	0.3%
VP -> V SBAR	6.6%	0.3%	73.0%	0.2%
VP -> V S	2.2%	1.3%	4.8%	70.8%

Including actual words information when making decisions about tree structure is necessary

# Weakening the independence assumption of PCFG

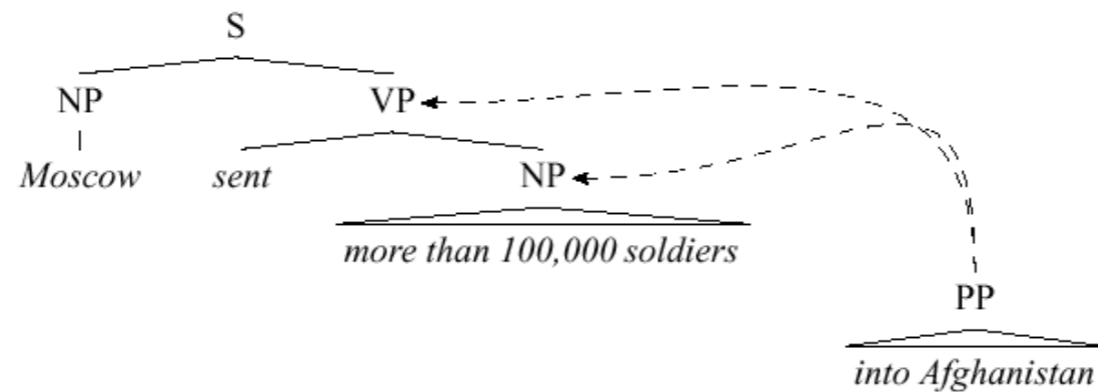
Probabilities dependent on Lexical words

Problem 2 : Phrasal Attachment

Lexical content of phrases provide information for decision

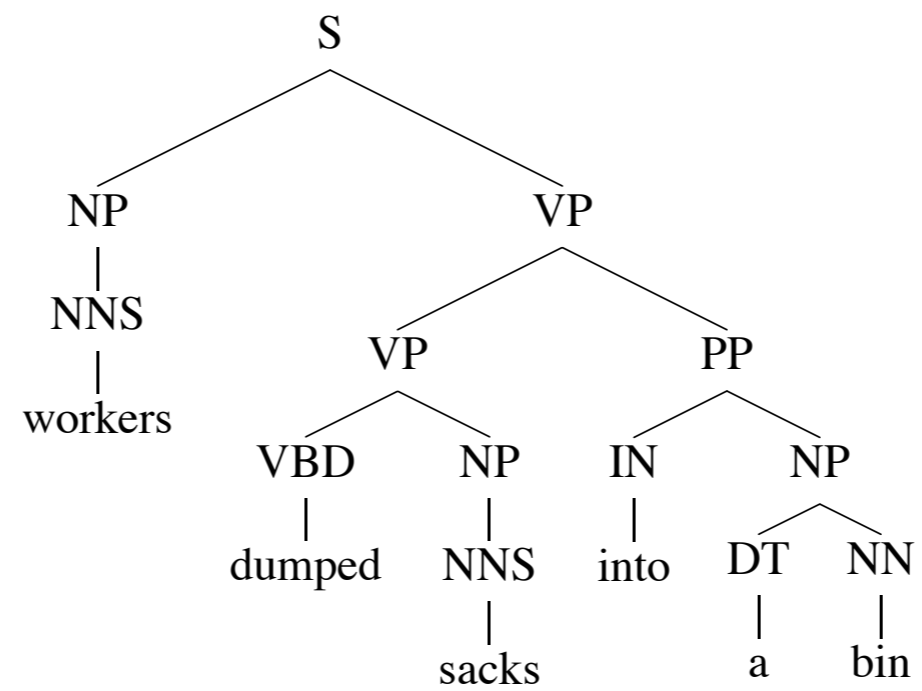
Syntactic category of the phrases provide very little information

Standard PCFG is worse than n-gram models

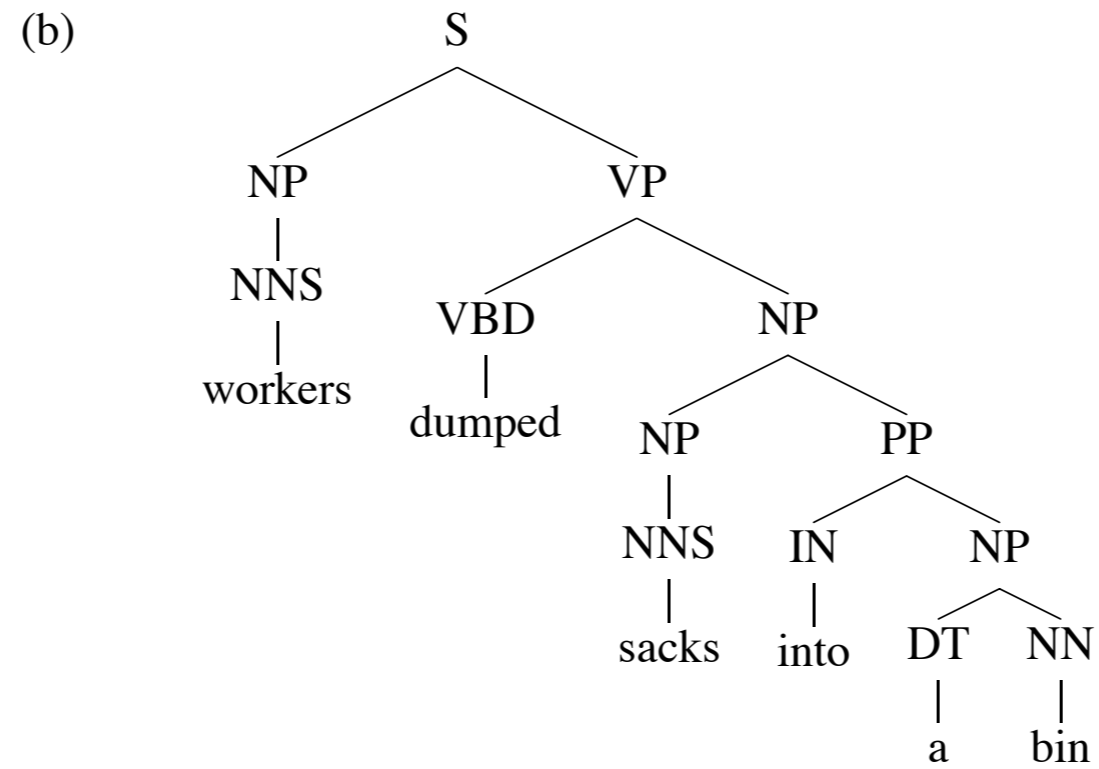


## Another case of PP attachment ambiguity

(a)



## Another case of PP attachment ambiguity



## Another case of PP attachment ambiguity

(a)

Rules
S → NP VP
NP → NNS
<b>VP → VP PP</b>
VP → VBD NP
NP → NNS
PP → IN NP
NP → DT NN
NNS → workers
VBD → dumped
NNS → sacks
IN → into
DT → a
NN → bin

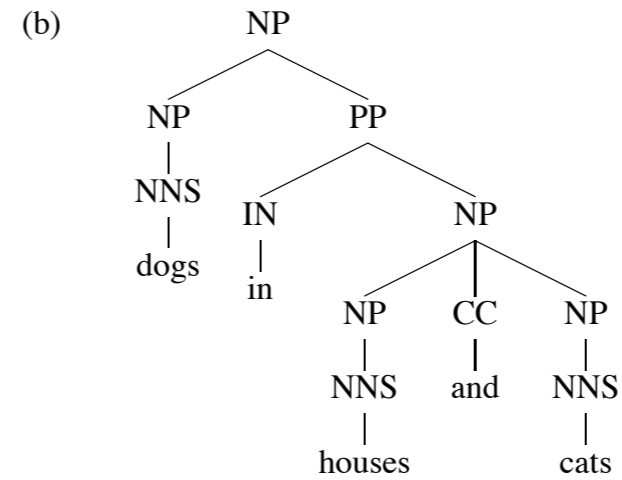
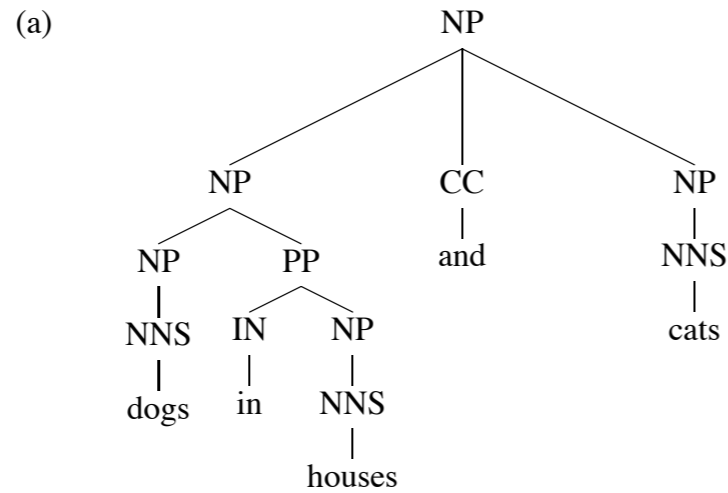
(b)

Rules
S → NP VP
NP → NNS
<b>NP → NP PP</b>
VP → VBD NP
NP → NNS
PP → IN NP
NP → DT NN
NNS → workers
VBD → dumped
NNS → sacks
IN → into
DT → a
NN → bin

If  $P(\text{NP} \rightarrow \text{NP PP} \mid \text{NP}) > P(\text{VP} \rightarrow \text{VP PP} \mid \text{VP})$  then (b) is more probable, else (a) is more probable.

**Attachment decision is completely independent of the words**

# A case of coordination ambiguity



(a)

Rules
NP → NP CC NP
NP → NP PP
NP → NNS
PP → IN NP
NP → NNS
NP → NNS
NNS → dogs
IN → in
NNS → houses
CC → and
NNS → cats

(b)

Rules
NP → NP CC NP
NP → NP PP
NP → NNS
PP → IN NP
NP → NNS
NP → NNS
NNS → dogs
IN → in
NNS → houses
CC → and
NNS → cats

**Here the two parses have identical rules, and therefore have identical probability under any assignment of PCFG rule probabilities**

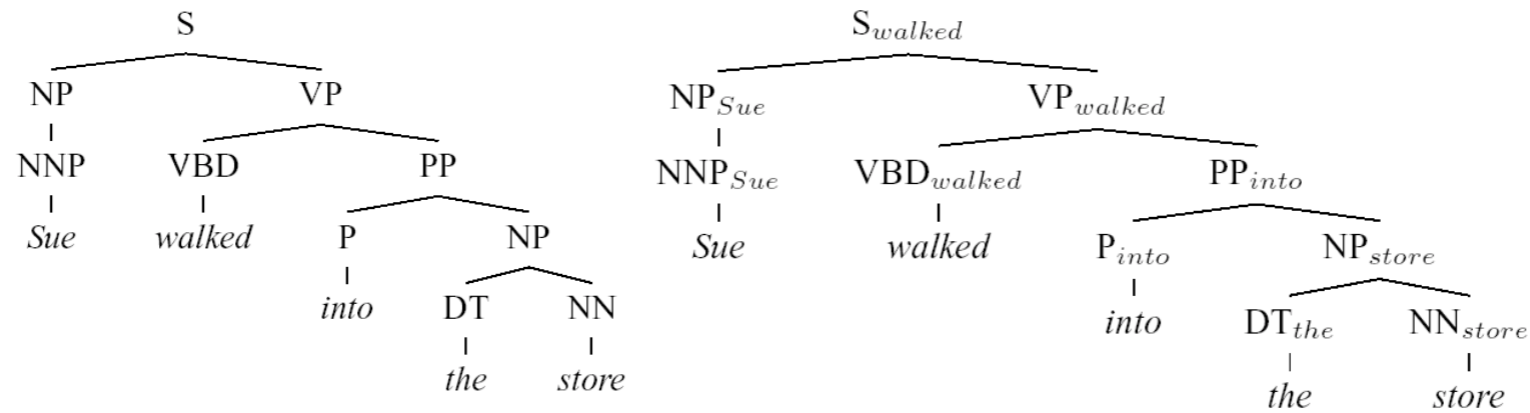


# Weakening the independence assumption of PCFG

Probabilities dependent on Lexical words

Solution

Lexicalize CFG : Each phrasal node with its **head word**



Background idea

Strong lexical dependencies between heads and their dependents

## Heads in Context-Free Rules

Add annotations specifying the “**head**” of each rule:

S	⇒	NP	<b>VP</b>
VP	⇒	<b>Vi</b>	
VP	⇒	<b>Vt</b>	NP
VP	⇒	<b>VP</b>	PP
NP	⇒	DT	<b>NN</b>
NP	⇒	<b>NP</b>	PP
PP	⇒	<b>IN</b>	NP

Vi	⇒	sleeps
Vt	⇒	saw
NN	⇒	man
NN	⇒	woman
NN	⇒	telescope
DT	⇒	the
IN	⇒	with
IN	⇒	in

Note: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase, DT=determiner, Vi=intransitive verb, Vt=transitive verb, NN=noun, IN=preposition

## More about heads

- Each context-free rule has one “special” child that is the head of the rule. e.g.,

S	⇒	NP	<b>VP</b>	(VP is the head)
VP	⇒	<b>Vt</b>	NP	(Vt is the head)
NP	⇒	DT	NN	<b>NN</b> (NN is the head)

- A core idea in linguistics  
(X-bar Theory, Head-Driven Phrase Structure Grammar)
- Some intuitions:
  - The central sub-constituent of each rule.
  - The semantic predicate in each rule.

## Rules which recover heads: Example rules for NPs

**If** the rule contains NN, NNS, or NNP:

Choose the rightmost NN, NNS, or NNP

**Else If** the rule contains an NP: Choose the leftmost NP

**Else If** the rule contains a JJ: Choose the rightmost JJ

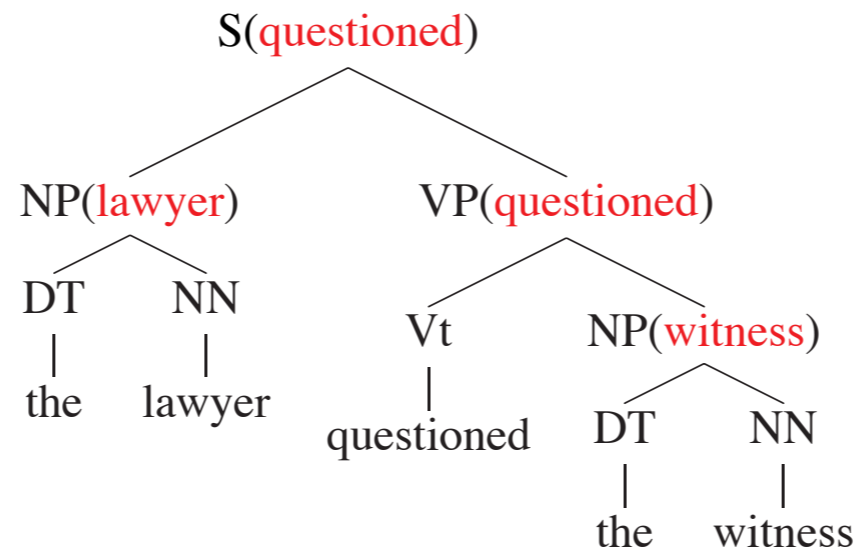
**Else If** the rule contains a CD: Choose the rightmost CD

**Else** Choose the rightmost child

e.g.,

NP	⇒	DT	NNP	NN
NP	⇒	DT	NN	NNP
NP	⇒	NP	PP	
NP	⇒	DT	JJ	
NP	⇒	DT		

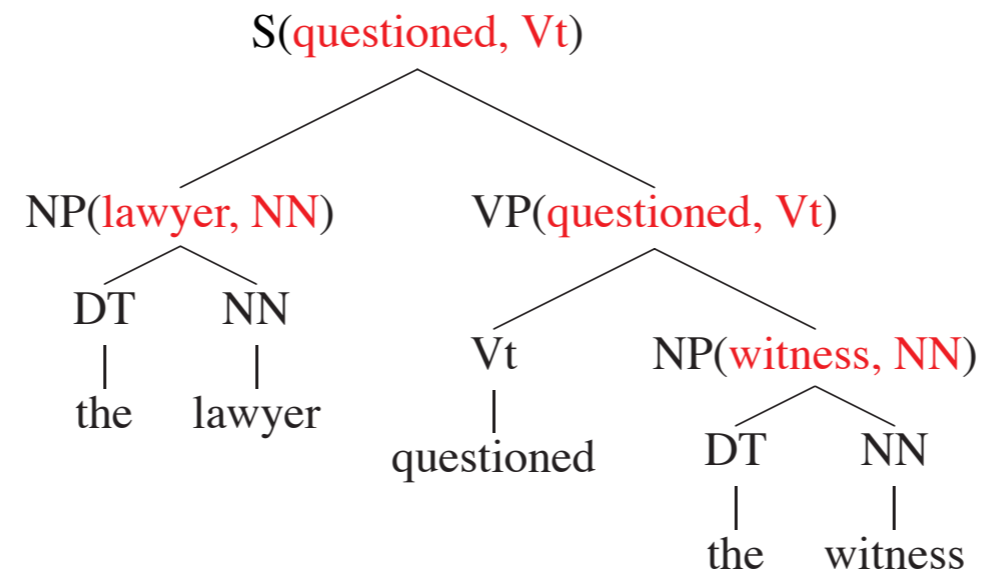
# Adding Headwords to Trees



- A constituent receives its **headword** from its **head child**.

S	⇒	NP	<b>VP</b>	(S receives headword from VP)
VP	⇒	<b>Vt</b>	NP	(VP receives headword from Vt)
NP	⇒	DT	<b>NN</b>	(NP receives headword from NN)

## Adding Headtags to Trees



- Also propagate **part-of-speech tags** up the trees

## Explosion of number of rules

New rules might look like:

VP[gave] → V[gave] NP[man] NP[book]

But this would be a massive explosion in number of rules (and parameters)

## Sparseness and the Penn Treebank

- The Penn Treebank – 1 million words of parsed English *WSJ* – has been a key resource (because of the widespread reliance on supervised learning)
- But 1 million words is like nothing:
  - 965,000 constituents, but only 66 WHADJP, of which only 6 aren't *how much* or *how many*, but there is an infinite space of these (*how clever/original/incompetent (at risk assessment and evaluation)*)
- Most of the probabilities that you would like to compute, you can't compute



## Sparseness and the Penn Treebank

- Most intelligent processing depends on bilexical statistics: likelihoods of relationships between pairs of words.
- Extremely sparse, even on topics central to the *WSJ*:
  - stocks plummeted 2 occurrences
  - stocks stabilized 1 occurrence
  - stocks skyrocketed 0 occurrences
  - #stocks discussed 0 occurrences
- So far there has been very modest success augmenting the Penn Treebank with extra unannotated materials or using semantic classes or clusters (cf. Charniak 1997, Charniak 2000) – as soon as there are more than tiny amounts of annotated training data.

**Lexicalized, Markov out from head**

## **Collins 1997: Markov model out from head**

- Charniak (1997) expands each phrase structure tree in a single step.
- This is good for capturing dependencies between child nodes
- But it is bad because of data sparseness
- A pure dependency, one child at a time, model is worse
- But one can do better by in between models, such as generating the children as a Markov process on both sides of the head (Collins 1997; Charniak 2000)

## Modeling Rule Productions as Markov Processes

- Step 1: generate category of head child
- 

$S(\text{told}, V[6])$



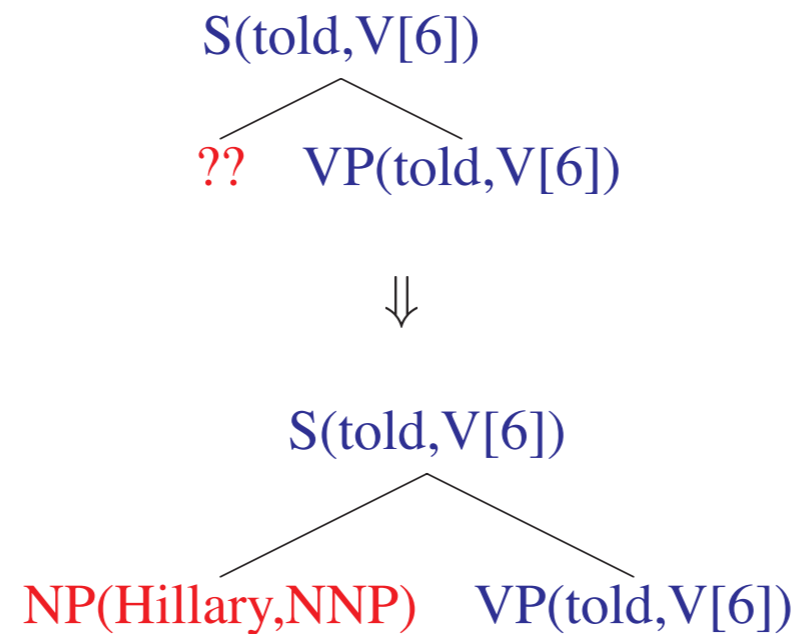
$S(\text{told}, V[6])$

|  
 $VP(\text{told}, V[6])$

$P_h(VP \mid S, \text{told}, V[6])$

## Modeling Rule Productions as Markov Processes

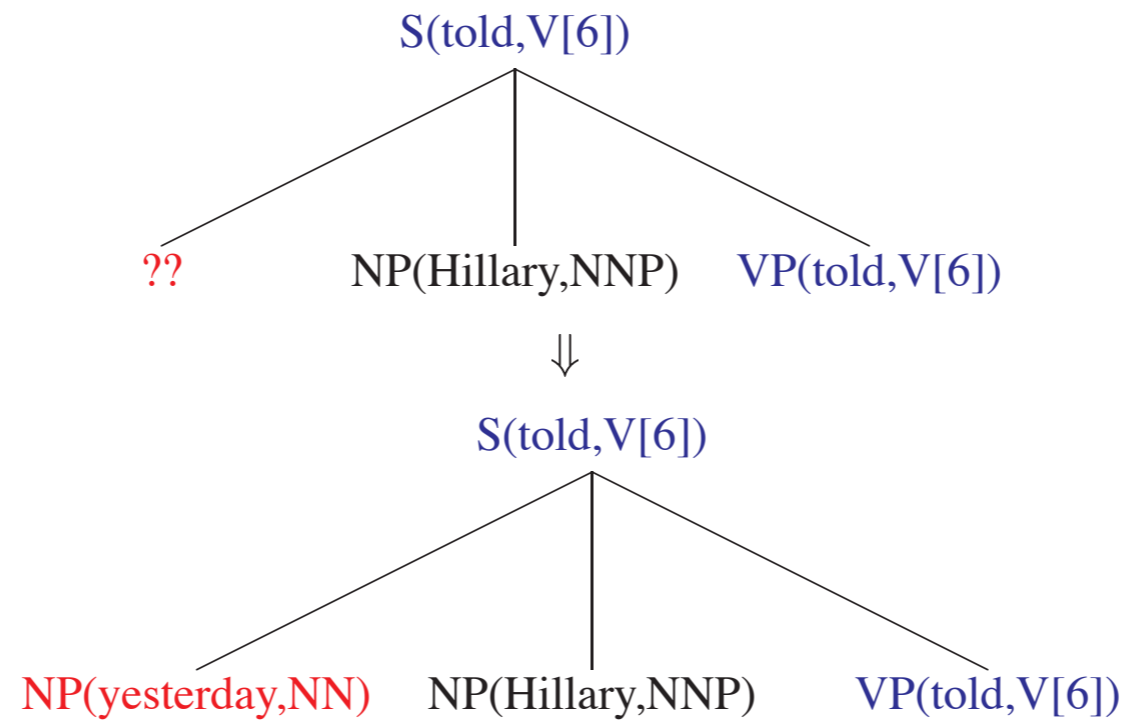
- Step 2: generate left modifiers in a Markov chain
- 



$$P_h(\text{VP} \mid \text{S, told, V[6]}) \times P_d(\text{NP(Hillary, NNP)} \mid \text{S, VP, told, V[6], LEFT})$$

## Modeling Rule Productions as Markov Processes

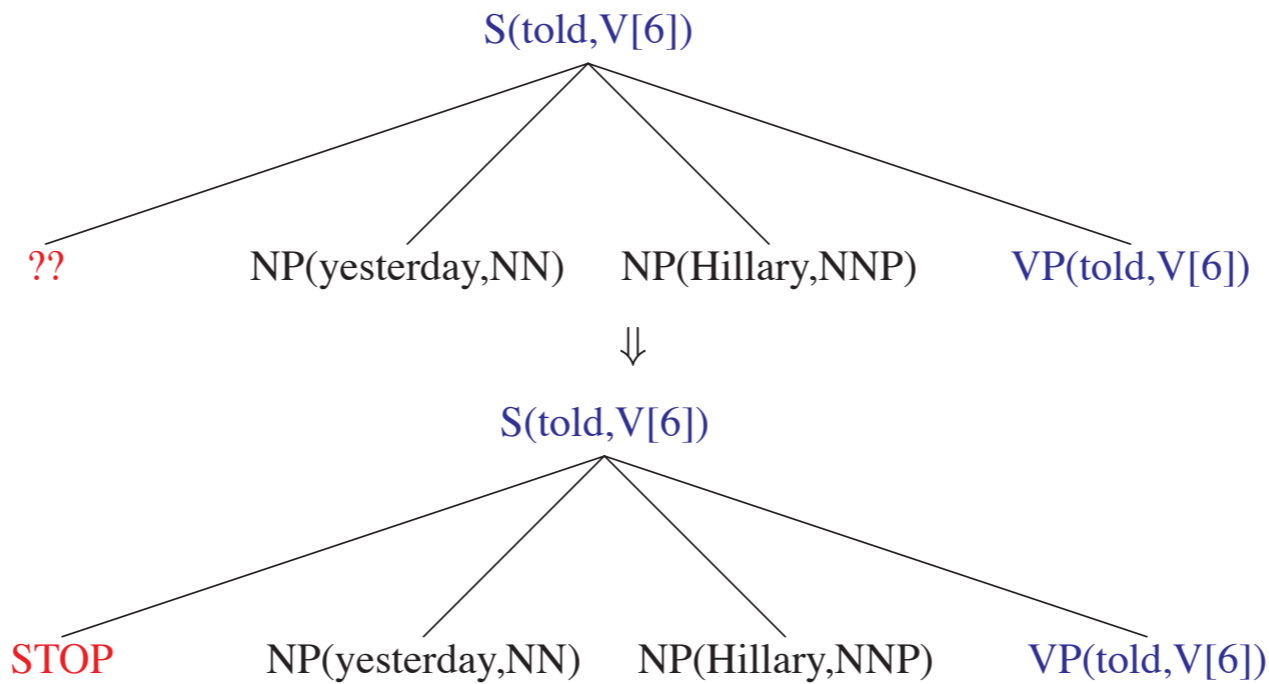
- Step 2: generate left modifiers in a Markov chain



$$P_h(VP \mid S, \text{told}, V[6]) \times P_d(NP(\text{Hillary}, \text{NNP}) \mid S, VP, \text{told}, V[6], \text{LEFT}) \times P_d(NP(\text{yesterday}, \text{NN}) \mid S, VP, \text{told}, V[6], \text{LEFT})$$

## Modeling Rule Productions as Markov Processes

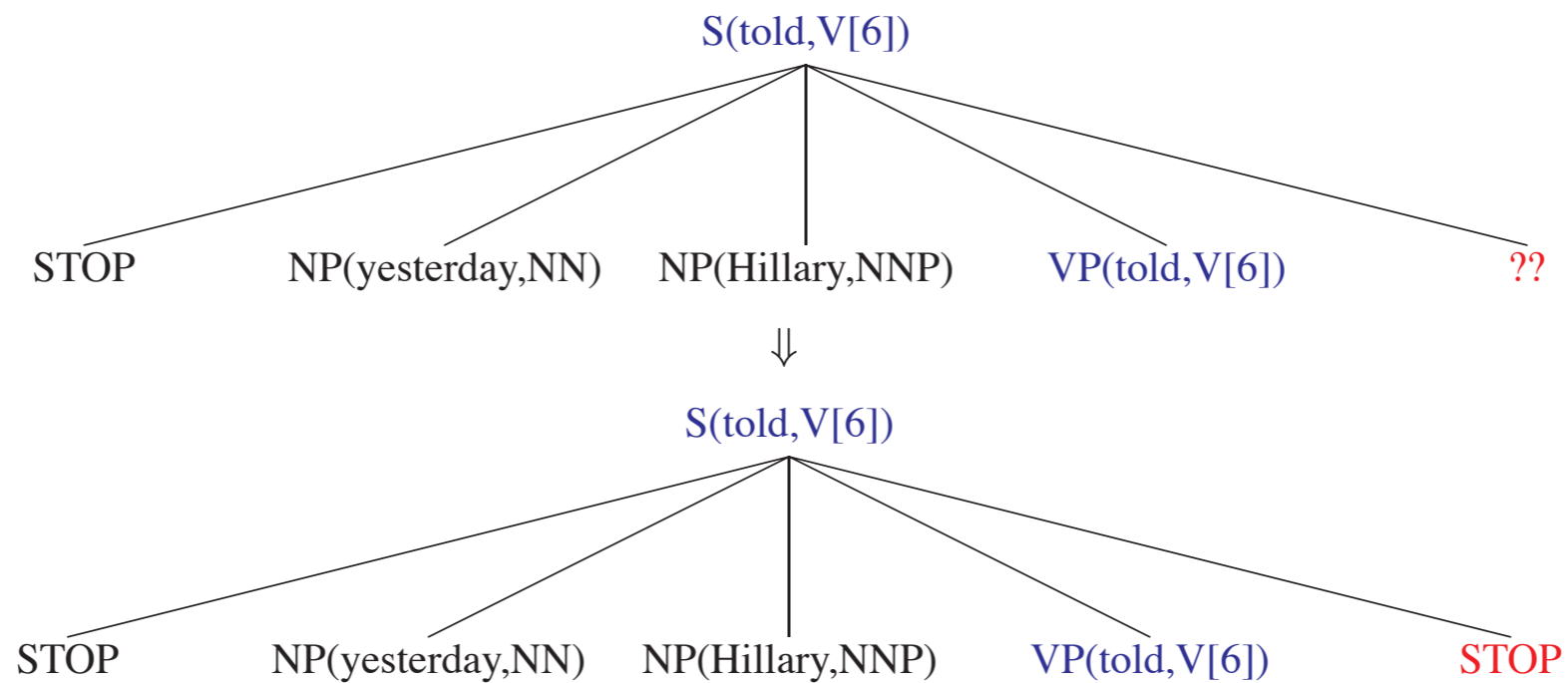
- Step 2: generate left modifiers in a Markov chain



$$P_h(VP \mid S, \text{told}, V[6]) \times P_d(NP(\text{Hillary}, NNP) \mid S, VP, \text{told}, V[6], \text{LEFT}) \times \\ P_d(NP(\text{yesterday}, NN) \mid S, VP, \text{told}, V[6], \text{LEFT}) \times P_d(\text{STOP} \mid S, VP, \text{told}, V[6], \text{LEFT})$$

## Modeling Rule Productions as Markov Processes

- Step 3: generate right modifiers in a Markov chain

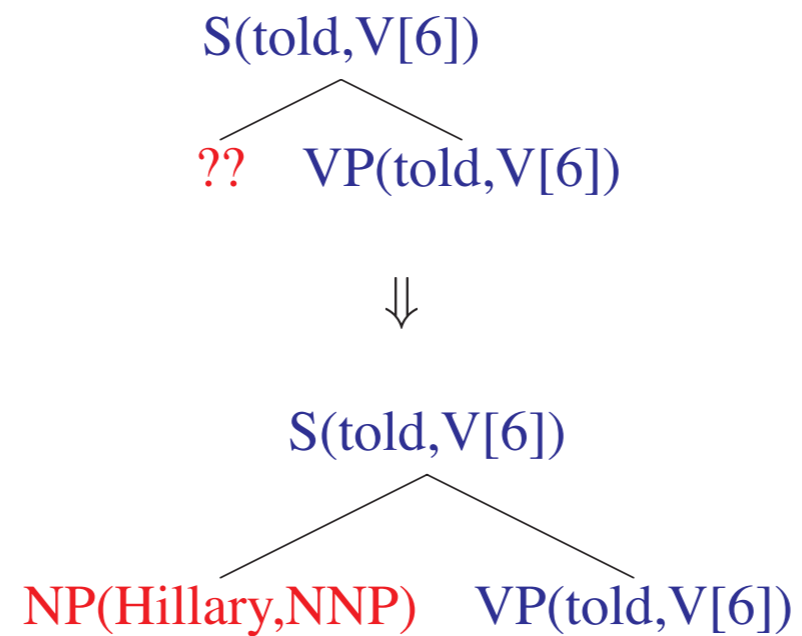


$$\begin{aligned}
 &P_h(\text{VP} \mid \text{S}, \text{told}, V[6]) \times P_d(\text{NP}(\text{Hillary}, \text{NNP}) \mid \text{S}, \text{VP}, \text{told}, V[6], \text{LEFT}) \times \\
 &P_d(\text{NP}(\text{yesterday}, \text{NN}) \mid \text{S}, \text{VP}, \text{told}, V[6], \text{LEFT}) \times P_d(\text{STOP} \mid \text{S}, \text{VP}, \text{told}, V[6], \text{LEFT}) \times \\
 &P_d(\text{STOP} \mid \text{S}, \text{VP}, \text{told}, V[6], \text{RIGHT})
 \end{aligned}$$



## A Refinement: Adding a Distance Variable

- $\Delta = 1$  if position is adjacent to the head.
- 



$$P_h(VP \mid S, \text{told}, V[6]) \times \\ P_d(NP(\text{Hillary}, NNP) \mid S, VP, \text{told}, V[6], \text{LEFT}, \Delta = 1)$$

## **Adding dependency on structure**

## Weakening the independence assumption of PCFG

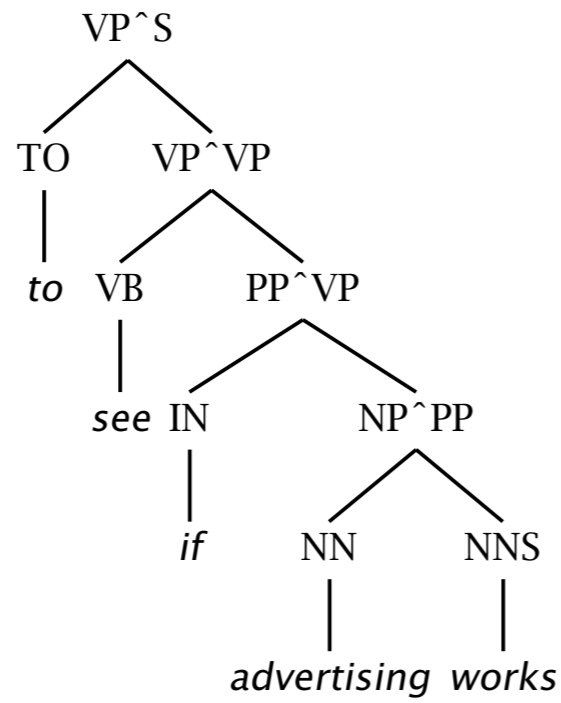
Probabilities dependent on structural context

PCFGs are also deficient on purely structural grounds too

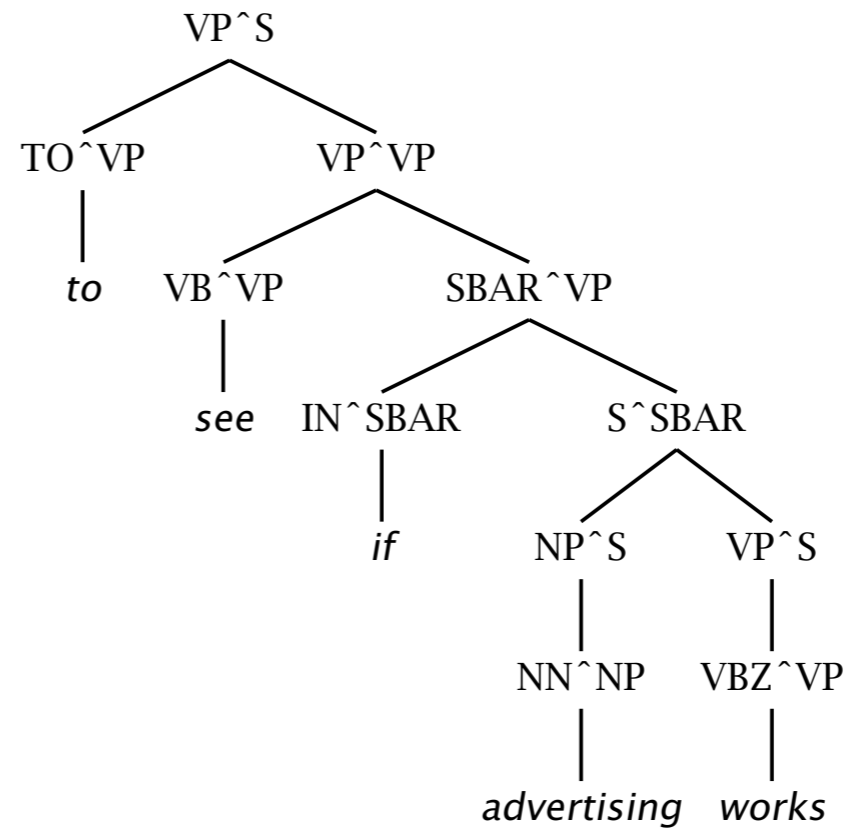
Really context independent?

Expansion	% as Subj	% as Obj
NP → PRP	13.7%	2.1%
NP → NNP	3.5%	0.9%
NP → DT NN	5.6%	4.6%
NP → NN	1.4%	2.8%
NP → NP SBAR	0.5%	2.6%
NP → NP PP	5.6%	14.1%

## Weakening the independence assumption of PCFG



(a)



(b)