Context-Free Parsing: The CKY Algorithm and Probabilistic Parsing

Introduction to Natural Language Processing
Computer Science 585—Fall 2009
University of Massachusetts Amherst

David Smith

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
  - $X \rightarrow Y\ Z$ X, Y, Z nonterminals
  - $X \rightarrow a$ X nonterminal, a terminal
  - $S \rightarrow \varepsilon$ 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

$$
\begin{aligned}
S &\rightarrow TST \mid aB \\
T &\rightarrow B \mid S \\
B &\rightarrow b \mid \epsilon
\end{aligned}
$$

CNF Conversion

Original

$$
\begin{aligned}
S &\rightarrow TST \mid aB \\
T &\rightarrow B \mid S \\
B &\rightarrow b \mid \epsilon
\end{aligned}
$$

New start symbol

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow TST \mid aB \\
T &\rightarrow B \mid S \\
B &\rightarrow b \mid \epsilon
\end{aligned}
$$

CNF Conversion

Original

$$
\begin{aligned}
S &\rightarrow TST \mid aB \\
T &\rightarrow B \mid S \\
B &\rightarrow b \mid \epsilon
\end{aligned}
$$

Remove nulls

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow TST \mid aB \mid a \mid ST \mid TS \\
T &\rightarrow B \mid S \\
B &\rightarrow b \mid \epsilon
\end{aligned}
$$

New start symbol

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow TST \mid aB \\
T &\rightarrow B \mid S \\
B &\rightarrow b \mid \epsilon
\end{aligned}
$$

CNF Conversion

Original

$$
\begin{aligned}
S &\rightarrow TST \mid aB \\
T &\rightarrow B \mid S \\
B &\rightarrow b \mid \epsilon
\end{aligned}
$$

Remove nulls

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow TST \mid aB \mid a \mid ST \mid TS \\
T &\rightarrow B \mid S \\
B &\rightarrow b \mid \epsilon
\end{aligned}
$$

New start symbol

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow TST \mid aB \\
T &\rightarrow B \mid S \\
B &\rightarrow b \mid \epsilon
\end{aligned}
$$

Remove unary cycles

$$
\begin{aligned}
S_0 &\rightarrow TST \mid aB \mid a \mid ST \mid TS \\
S &\rightarrow TST \mid aB \mid a \mid ST \mid TS \\
T &\rightarrow b \mid TST \mid aB \mid a \mid ST \mid TS \\
B &\rightarrow b
\end{aligned}
$$

CNF Conversion

$$
\begin{aligned}
S_0 &\rightarrow TST \mid UB \mid a \mid ST \mid TS \\
S &\rightarrow TST \mid UB \mid a \mid ST \mid TS \\
T &\rightarrow b \mid TST \mid UB \mid a \mid ST \mid TS \\
B &\rightarrow b \\
U &\rightarrow a
\end{aligned}
$$

Make terminal rules unary

CNF Conversion

$$
\begin{aligned}
S_0 &\rightarrow TST \mid UB \mid a \mid ST \mid TS \\
S &\rightarrow TST \mid UB \mid a \mid ST \mid TS \\
T &\rightarrow b \mid TST \mid UB \mid a \mid ST \mid TS \\
B &\rightarrow b \\
U &\rightarrow a
\end{aligned}
$$

Make terminal rules unary

Make non-terminal rules binary

$$
\begin{aligned}
S_0 &\rightarrow TZ \mid UB \mid a \mid ST \mid TS \\
S &\rightarrow TZ \mid UB \mid a \mid ST \mid TS \\
T &\rightarrow b \mid TZ \mid UB \mid a \mid ST \mid TS \\
B &\rightarrow b \\
U &\rightarrow a \\
Z &\rightarrow ST
\end{aligned}
$$

CKY Algorithm

- Input: string of n words (m in M&S)
- 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^{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^{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^{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^{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^{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

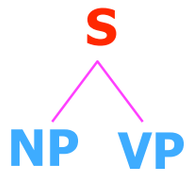
0 PP → P NP

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 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

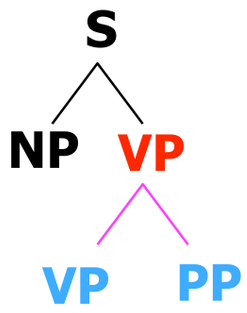
1 S → NP VP
6 S → Vst NP
2 S → S PP
1 VP → V NP
2 VP → VP PP
1 NP → Det N
2 NP → NP PP
3 NP → NP NP
0 PP → P NP



	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

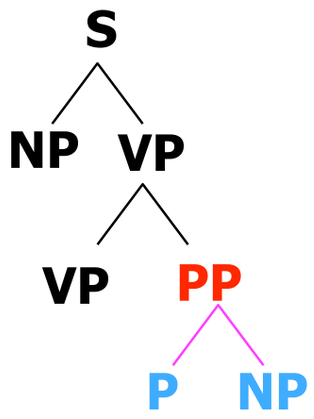
time	1 flies	2 like	3 an	4 arrow	5
0	NP 3 Vst 3	NP 10 S 8 S 13	–	–	NP 24 S 22 S 27 NP 24 S 27 S 22 S 27
1		NP 4 VP 4	–	–	NP 18 S 21 VP 18
2			P 2 V 5	–	PP 12 VP 16
3				Det 1	NP 10
4					N 8



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**



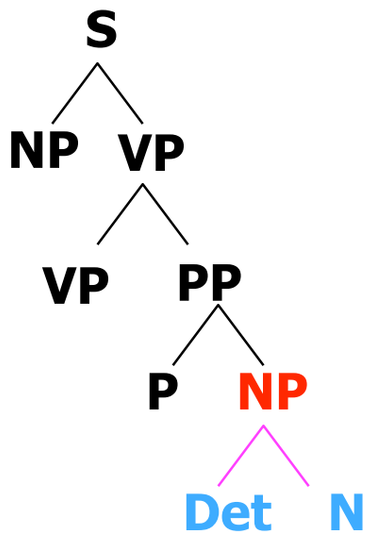
1 S → NP VP
6 S → Vst NP
2 S → S PP

1 VP → V NP
2 VP → VP PP

1 NP → Det N
2 NP → NP PP
3 NP → NP NP

0 PP → P NP

	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

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

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

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

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

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

Parsing as Deduction

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

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

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

$$score(constit(A, i, i+1)) = \max_{A,W,i} score(word(W, i)) \cdot score(A \rightarrow W)$$

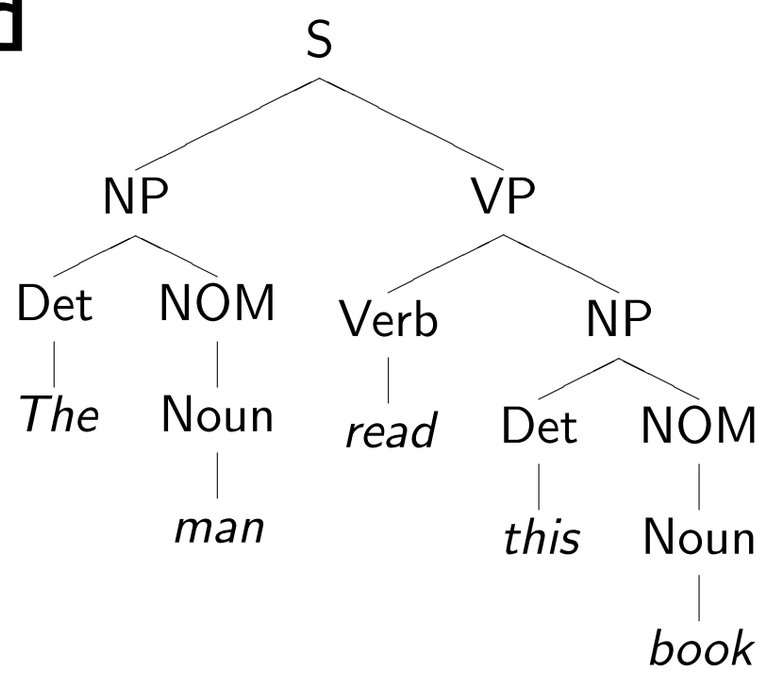
And how about the inside algorithm?

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?