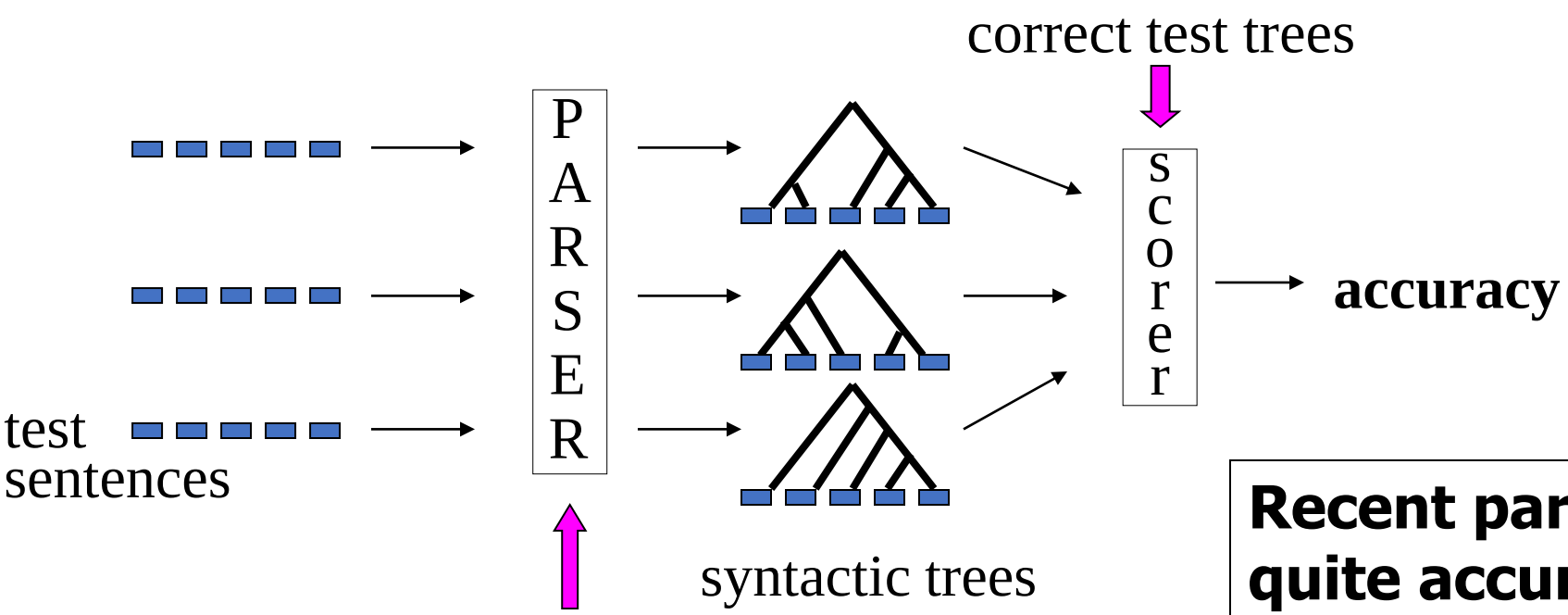
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Syntactic Parsing

Lê Thanh Hương
School of Information and Communication Technology
Email: huonglt@soict.hust.edu.vn

The parsing problem



Grammar

S → NP VP
NP → John, garbage
VP → laughed, walks

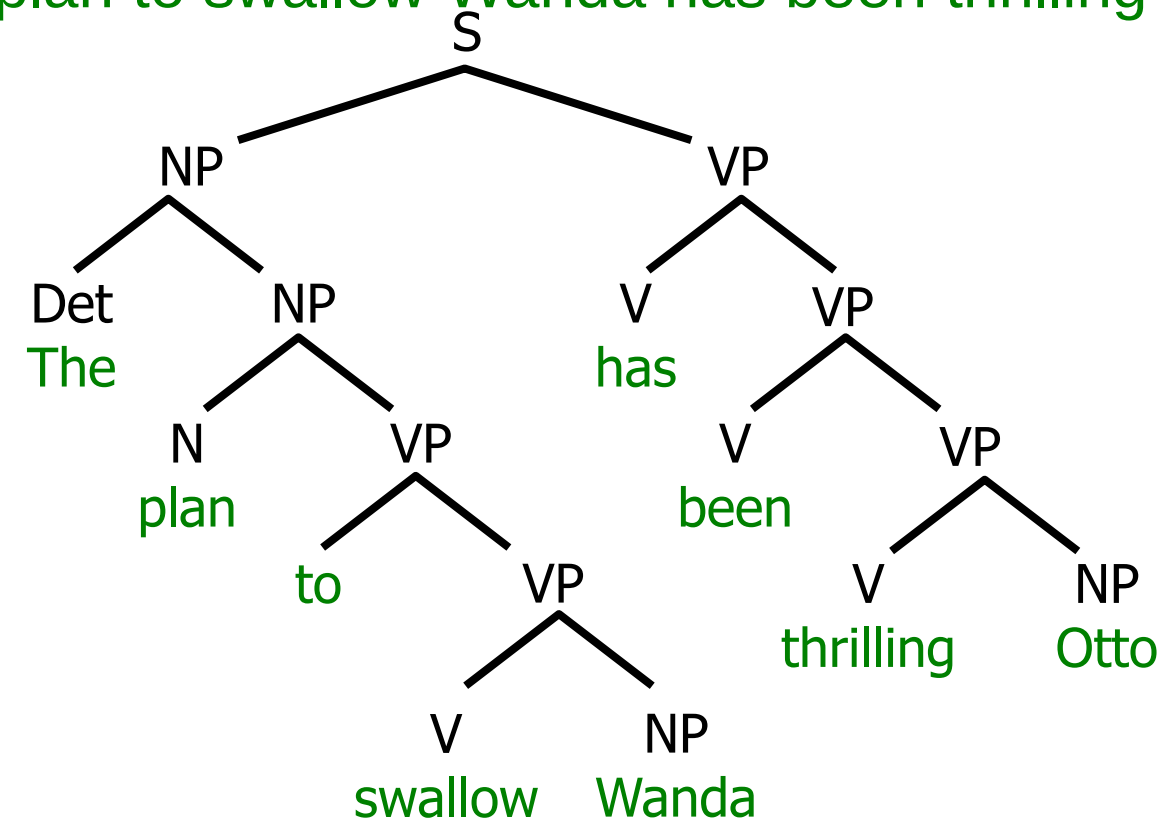
Recent parsers quite accurate

(Eisner, Collins, Charniak, etc.)

Syntactic Structure

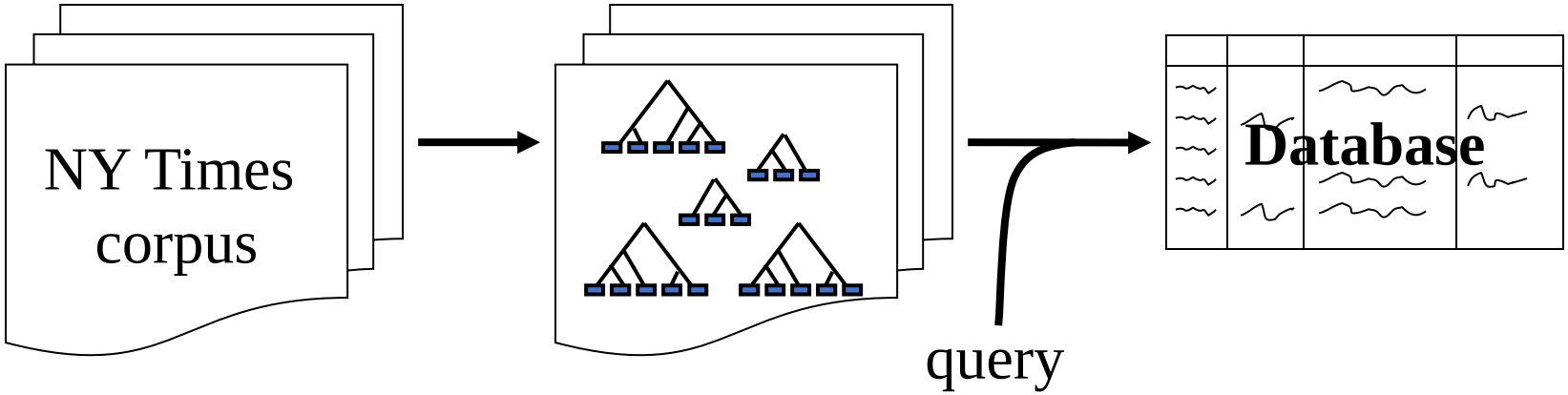
A parse tree represent the syntactic structure of a sentence.

The plan to swallow Wanda has been thrilling Otto.



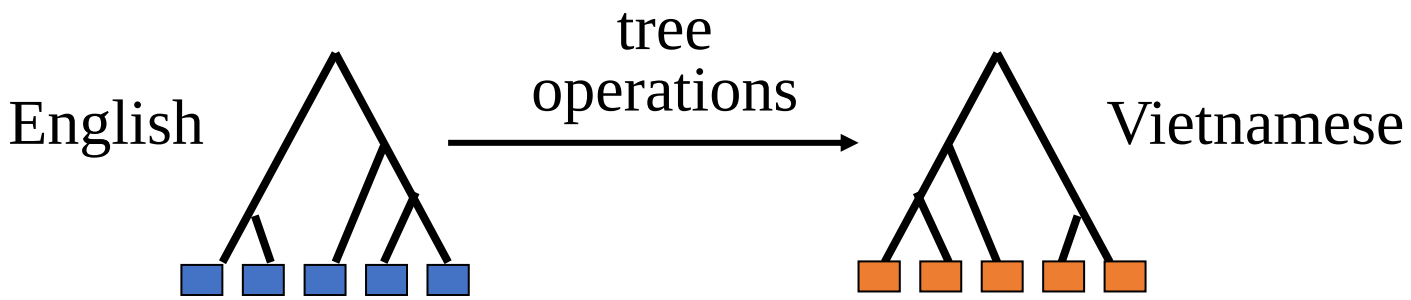
Some applications

- Grammar checking
- Question answering/chatbot
- Text summarization
- Information Extraction



Some applications

- Machine translation (Alshawi 1996, Wu 1997, ...)



- Speech recognition using parsing (Chelba et al 1998)

  Put the file in the folder.

  Put the file and the folder.

Some definitions

- A **grammar** is a formal specification of the structures allowable in a language.
- A **parsing algorithm** is a method for determining the structure of a sentence with respect to a grammar.
- A **parser** is a program that determines the structure of sentences.

A Simple Grammar

- A grammar is described as a set of **rewrite rules**.
- Symbols that cannot be decomposed are **terminal symbols**.
- Symbols that can be decomposed are **nonterminal symbols**.

Consider grammar G:

$S \rightarrow NP\ VP$

$NP \rightarrow$ John, garbage

$VP \rightarrow$ laughed, walks

G can produce the following sentences:

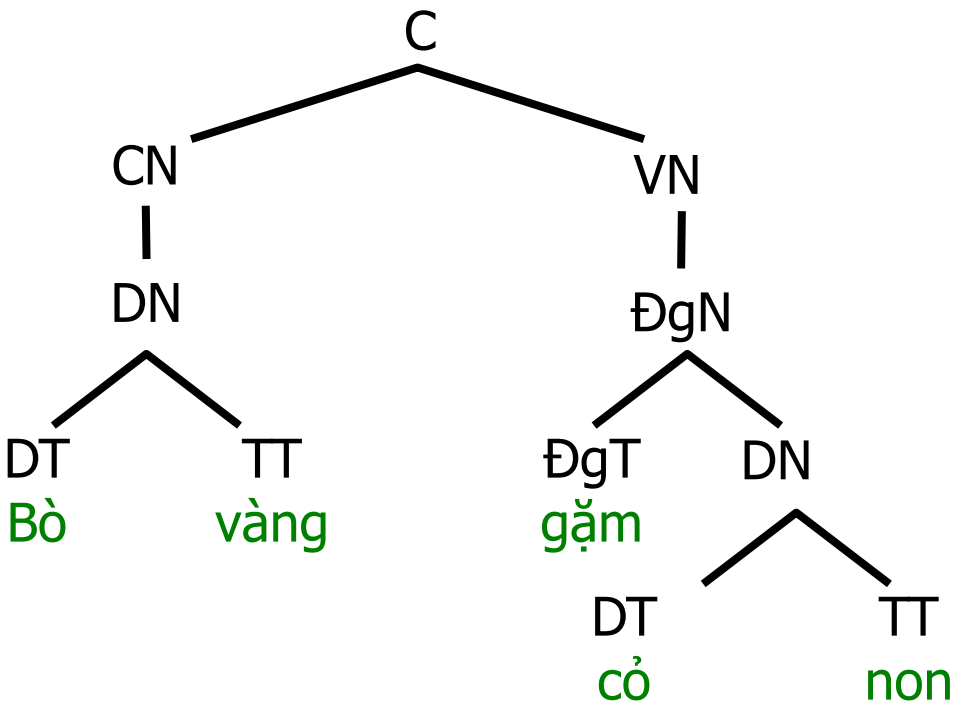
John laughed. John walks.

Garbage laughed. Garbage walks.

Syntactic Structure

Sentence “Bò vàng gặm cỏ non”

- Syntactic rules:
  - C → CN VN
  - CN → DN
  - VN → ĐgN
  - ĐgN → ĐgT DN
  - DN → DT TT



Context-Free Grammar

… also known as a phrase structure grammar

- G = <T,N,P,S,R>
  - T - set of terminals
  - N - set of nonterminals
  - P - preterminals which always rewrite as terminals, P $\subset$ N
  - S - start symbol
  - R: X $\rightarrow \gamma$, X is a nonterminal; γ is a sequence of terminals and nonterminals (may be empty)
  - A grammar G generates a language L
- A **recognizer:** returns yes or no
- A **parser:** returns a set of parse trees

Example

- G1 = ({a,b}, {X}, X, {X→ε, X→aXb})

  Define L(G1)

X-> aXb -> ab

X → aXb → aaXbb -> ….→$a^n b^n$

- G2 = ({a,b}, {X}, X, {X→ε, X→aXb, X→XX})

  Define L(G2)

X-aXXb-aanbnambmb

Context-Free Grammar

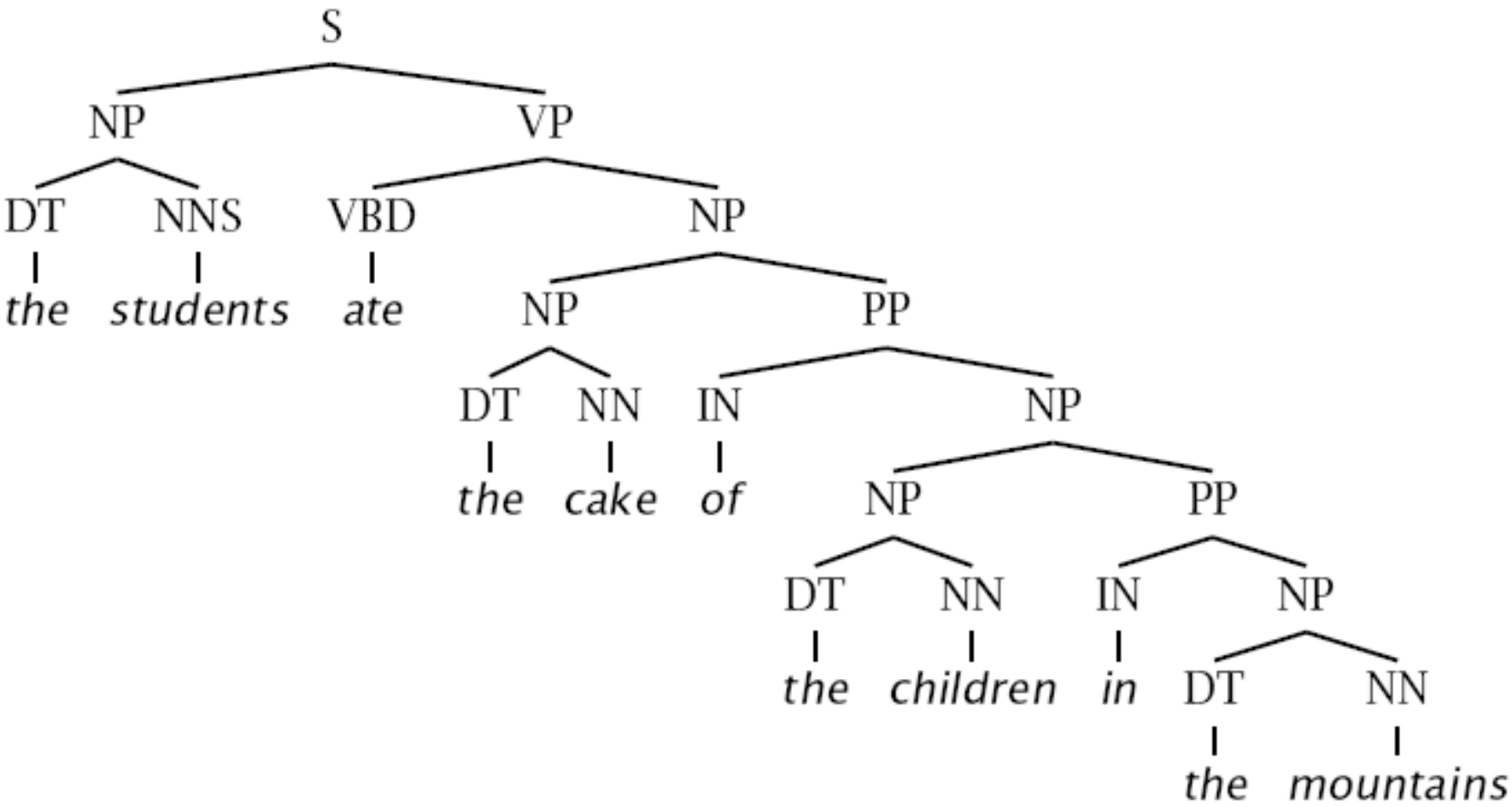
$$
\begin{array}{lcl}
\text{S} & \rightarrow & \text{NP VP} \\
\text{NP} & \rightarrow & \left\{\begin{array}{l}\text{DT NNS}\\ \text{DT NN}\\ \text{NP PP}\end{array}\right\} \\
\text{VP} & \rightarrow & \left\{\begin{array}{l}\text{VP PP}\\ \text{VBD}\\ \text{VBD NP}\end{array}\right\} \\
\text{PP} & \rightarrow & \text{IN NP}
\end{array}
$$

$$
\begin{array}{lcl}
\text{DT} & \rightarrow & \textit{the} \\
\text{NNS} & \rightarrow & \left\{\begin{array}{l}\textit{children}\\ \textit{students}\\ \textit{mountains}\end{array}\right\} \\
\text{VBD} & \rightarrow & \left\{\begin{array}{l}\textit{slept}\\ \textit{ate}\\ \textit{saw}\end{array}\right\} \\
\text{IN} & \rightarrow & \left\{\begin{array}{l}\textit{in}\\ \textit{of}\end{array}\right\} \\
\text{NN} & \rightarrow & \textit{cake}
\end{array}
$$

Application of grammar rewrite rules

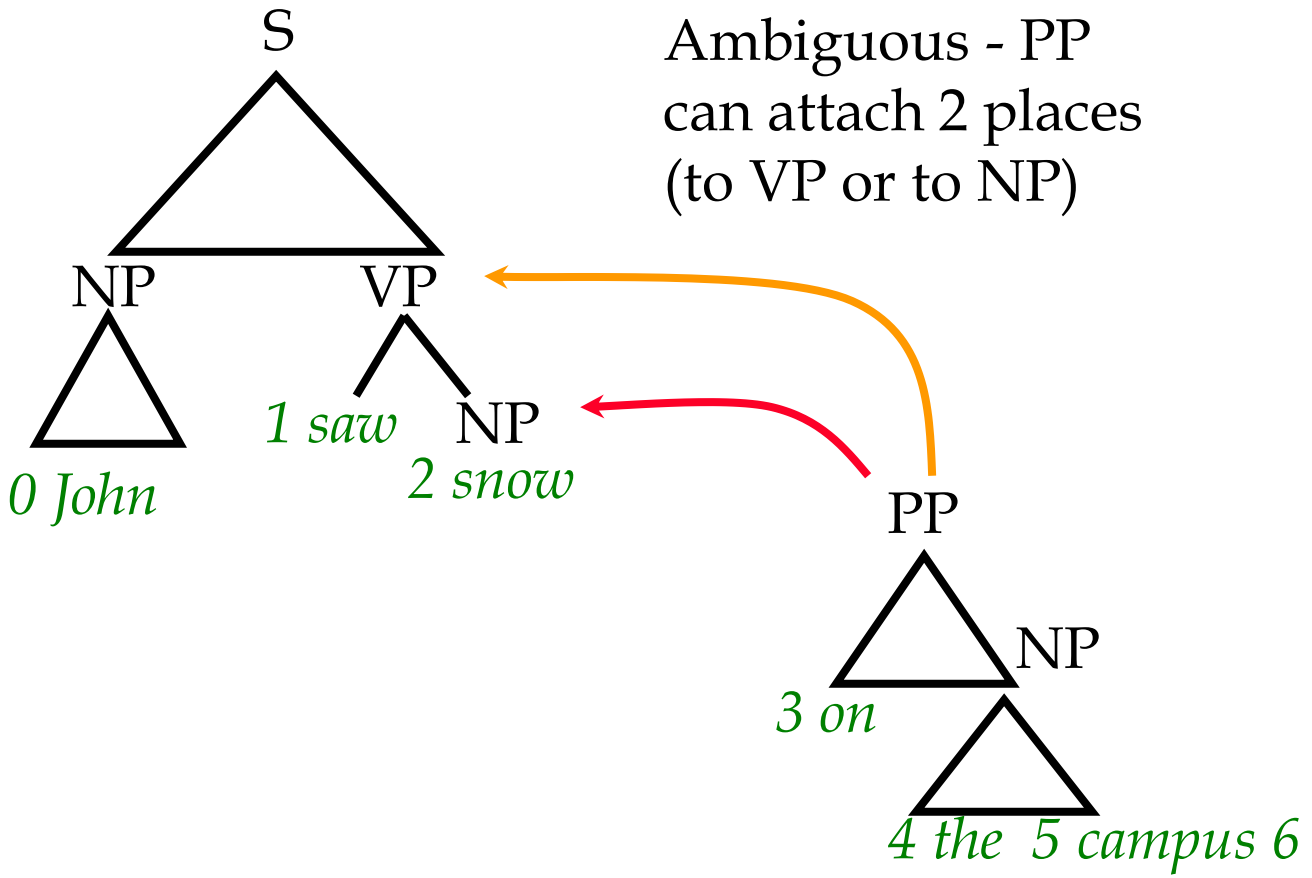
- S
  - → NP VP
  - → DT NNS VBD
  - → *The children slept*
- S
  - → NP VP
  - → DT NNS VBD NP
  - → DT NNS VBD DT NN
  - → *The children ate the cake*

Phrase structure is recursive



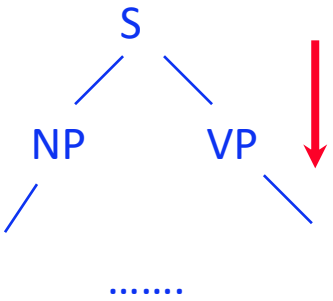
Natural language grammars are ambiguous

John saw snow on the campus



Ambiguous - PP can attach 2 places (to VP or to NP)

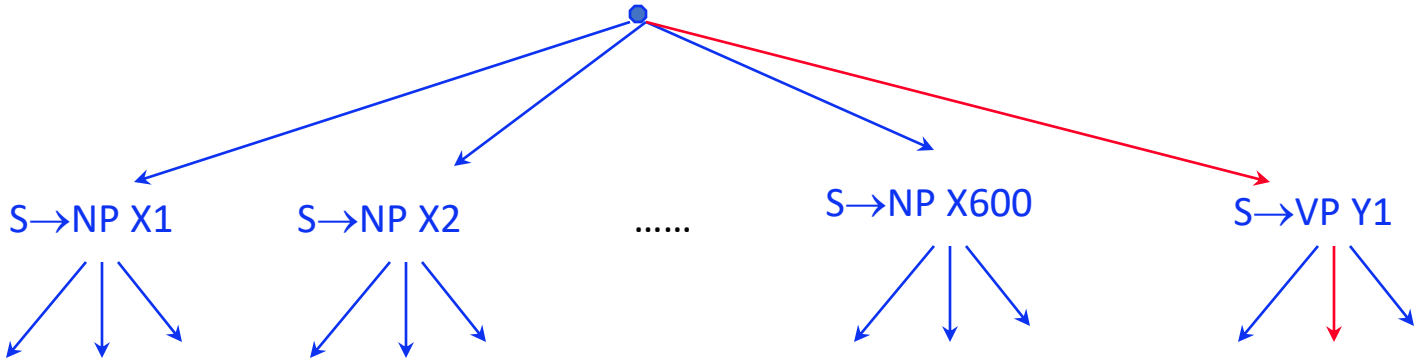
Top-down parsing



- Goal directed
- Starts with a list of constituents to be built (S, NP,VP,…)
- Rewrite the goals in the goal list by
  - matching one against the LHS of the grammar rules
  - expanding it with the RHS, attempting to match the sentence to be derived.
- If a goal can be rewritten in several ways → choose a rule to apply (search problem)
- Can use depth-first or breadth-first search

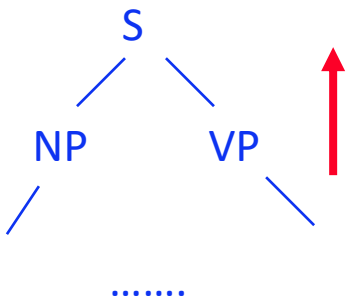
Problems with top-down parsing

- Left recursive rules
- A top-down parser will do badly if there are many different rules for the same LHS.



- Useless work: expands things that are possible top-down
- Top-down parsers do well if there is useful grammar-driven control
- Top-down is hopeless for rewriting parts of speech (preterminals) with words (terminals). In practice that is always done bottom-up as lexical lookup.
- Repeated work: anywhere there is common substructure

Bottom-up parsing



- Data directed
- Start with the string to be parsed
- If a sequence in the goal list matches the RHS of a rule $\rightarrow$ replace it by the LHS of the rule.
- Finish when the goal list = {S}.
- If the RHS of several rules match the goal list, then there is a choice of which rule to apply (search problem)
- Can use depth-first or breadth-first search

Problems with bottom-up parsing

- Inefficient when there is great lexical ambiguity
- Repeated work: anywhere there is common substructure
- Both TD (LL) and BU (LR) parsers frequently do work exponential in the sentence length.

CKY algorithm (a recognizer)

- Input: string of n words, a set of grammar rewrite rules in Chomsky normal form (CNF)
- Output: yes/no
- Data structure: n x n table (chart table)
  - rows labeled 0 to n-1
  - columns labeled 1 to n
  - cell [i,j] lists constituents found between i and j

Chomsky normal form

- All context free grammars that do not contain ε are possible to generate from a grammar in which all rules being in the form $A \rightarrow BC$ or $A \rightarrow a$, with $A,B,C \in N$ and $a \in T$
- E.g.: Find the Chomsky normal form for the grammar G with T = {a,b}, N ={S,A,B}, R is as follow:
  - $S \rightarrow bA|aB$ $\quad C \rightarrow b$ $\quad S \rightarrow CA$
  - $A \rightarrow bAA|aS|a$
  - $B \rightarrow aBB|bS|b$

CKY algorithm (bottom-up)

- for i := 1 to n
  - Add to [i-1,i] all categories for the ith 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.

Context-free grammar

1. Start→ S
2. S → NP VP
3. NP → Det Noun
4. NP → Name
5. NP → Name PP
6. PP → Prep NP
7. VP → V NP
8. VP → V NP PP
9. V → *ate*
10. Name → *John*
11. Name → *ice-cream, snow*
12. Noun → *ice-cream, pizza*
13. Noun → *table, guy, campus*
14. Det → *the*
15. Prep → *on*

Combination rule

- Cell[i,j] holds X iff
  - There exists a rule X→YZ;
  - Cell[i,k] holds Y & Cell[k,j] holds Z, with k between i and j, inclusively;

- Example: NP → DT [0,1] NN[1,2]

CKY must use binary branching rules

- Change VP→V NP PP to:
  - 8.a. VP→V Arguments
  - 8.b. Arguments → NP PP

1. Start→ S
2. S → NP VP
3. NP → DT NN
4. NP → Name
5. NP → NN PP
6. NP → DT NP
7. PP → Prep NP
8. VP → V NP

9.a. VP→V Arguments

9.b. Arguments → NP PP

“ The guy ate the ice-cream on the table”

	1	2	3	4	5	6	7	8
0	DT							
1		NN						
2			VBD					
3				DT				
4					NN			
5						Prep		
6							DT	
7								NN

Now apply ‘paste’ operation

	1	2	3	4	5	6	7	8
0	DT	NP						
1		NN						
2			VBD					
3				DT				
4					NN			
5						IN		
6							DT	
7								NN

Ambiguity!

5. NP → NN PP
9.a. VP→V Arguments
9.b. Arguments → NP PP

	1	2	3	4	5	6	7	8
0	DT	NP			S			S
1		NN						
2			VBD		VP			VP
3				DT	NP			Args
4					NN			NP
5						IN		PP
6							DT	NP
7								NN

Exercise

0 The 1 old 2 man 3 goes 4 too 5 fast 6

NP → DT NN

NP → DT JJ NN

NP → DT NP

V → VB | VBZ | VBD

VP → V

VP → V AdjP

VP → VP AdjP

AdjP → RB JJ

S → NP VP

DT → the

JJ → old

NN → man

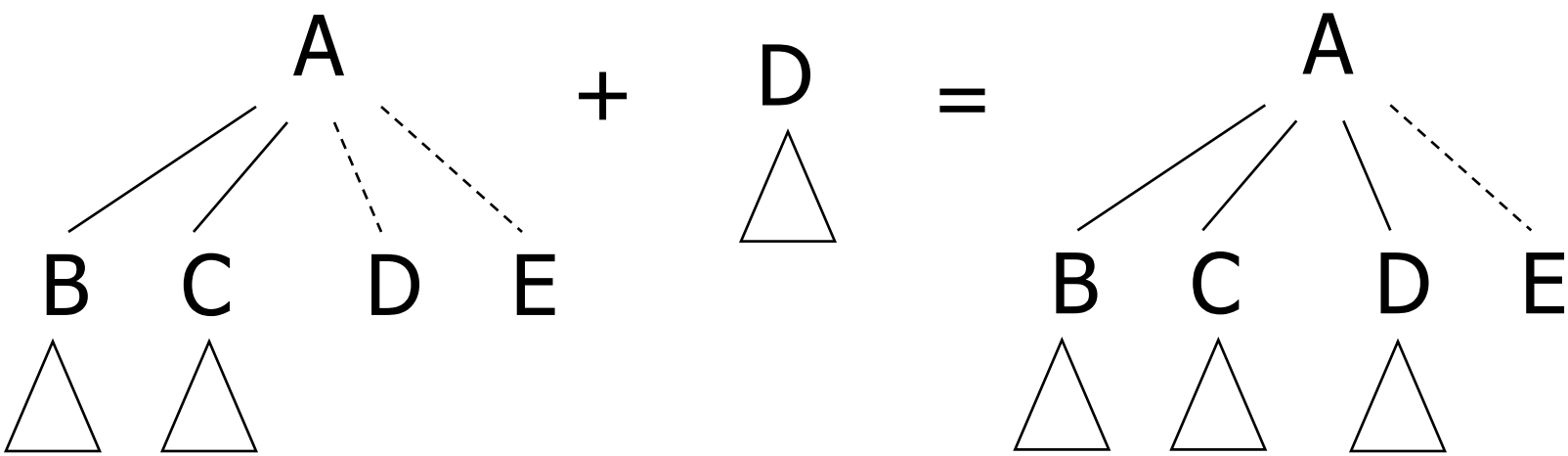
VBZ → goes

RB → too

JJ → fast

Earley's Algorithm (top-down)

- Finds constituents and <u>partial</u> constituents in input
  - A → B C . D E is partial: only the first half of the A



A → B C . D E A → B C D . E

- Proceeds <u>incrementally</u>, left-to-right

Example Grammar

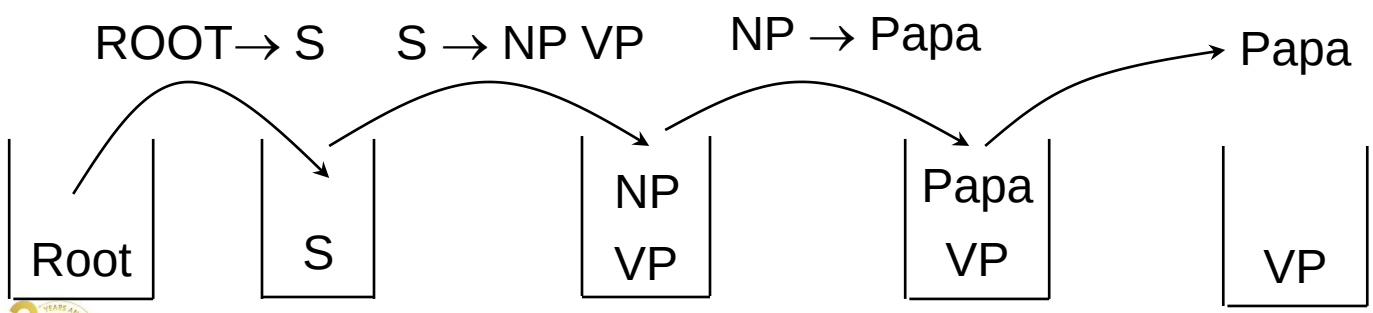
ROOT	$\rightarrow$ S	NP	$\rightarrow$ Papa
S	$\rightarrow$ NP VP	N	$\rightarrow$ caviar
NP	$\rightarrow$ Det N	N	$\rightarrow$ spoon
NP	$\rightarrow$ NP PP	V	$\rightarrow$ ate
VP	$\rightarrow$ VP PP	P	$\rightarrow$ with
VP	$\rightarrow$ V NP	Det	$\rightarrow$ the
PP	$\rightarrow$ P NP	Det	$\rightarrow$ a

0 Papa 1 ate 2 the 3 caviar 4 with 5 a 6 spoon 7

Recursive Descent (Đệ quy)

ROOT → S	VP → VP PP	NP → Papa	V → ate
S → NP VP	VP → V NP	N → caviar	P → with
NP → Det N	PP → P NP	N → spoon	Det → the
NP → NP PP			Det → a

- 0 ROOT → . S 0
  - 0 S → . NP VP 0
    - 0 NP → . Papa 0
    - 0 NP → Papa . 1
  - 0 S → NP . VP 1



Goal stack

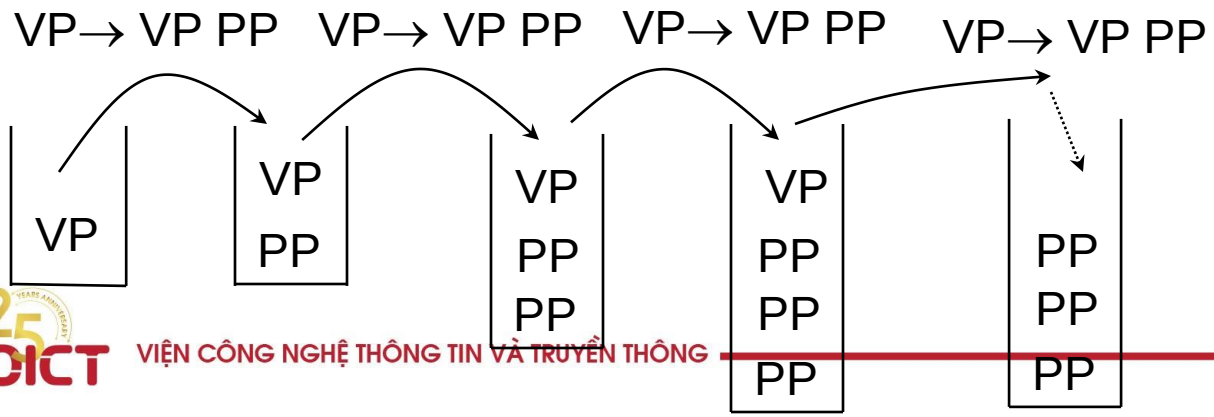
0 Papa 1 ate 2 the 3 caviar 4 with 5 a 6 spoon 7

Recursive Descent

ROOT → S	VP → VP PP	NP → Papa	V → ate
S → NP VP	VP → V NP	N → caviar	P → with
NP → Det N	PP → P NP	N → spoon	Det → the
NP → NP PP			Det → a

- 0 S → NP . VP 1
  - 1 VP → . VP PP 1
    - 1 VP → . VP PP 1
      - 1 VP → . VP PP 1

        1 VP → . VP PP 1 stack overflowed



0 Papa 1 ate 2 the 3 caviar 4 with 5 a 6 spoon 7

First Try: Recursive Descent

ROOT → S	VP → V NP	NP → Papa	V → ate
S → NP VP	VP → VP PP	N → caviar	P → with
NP → Det N	PP → P NP	N → spoon	Det → the
NP → NP PP			Det → a

- 0 ROOT → . S 0
  - 0 S → . NP VP 0
    - 0 NP → . Papa 0
    - 0 NP → Papa . 1
  - 0 S → NP . VP 1
    - 1 VP → . V NP 1 — after dot = nonterminal, so recursively look for it ("**predict**")
      - 1 V → . ate 1 — after dot = terminal, so look for it in the input ("**scan**")
      - 1 V → ate . 2 — after dot = nothing, so parent's subgoal is completed ("**attach**")
    - 1 VP → V . NP 2 — **predict** (next subgoal)
      - 2 NP → . ... 2 — do some more parsing and eventually ...
      - 2 NP → ... . 7 — we complete the parent's NP subgoal, so **attach**
    - 1 VP → V NP . 7 — **attach** again
  - 0 S → NP VP . 7 — **attach** again

0 Papa 1 ate 2 the 3 caviar 4 with 5 a 6 spoon 7

First Try: Recursive Descent

ROOT → S	VP → V NP	NP → Papa	V → ate
S → NP VP	VP → VP PP	N → caviar	P → with
NP → Det N	PP → P NP	N → spoon	Det → the
NP → NP PP			Det → a

- 0 ROOT → . S 0
  - 0 S → . NP VP 0
    - 0 NP → . Papa 0
    - 0 NP → Papa . 1
  - 0 S → NP . VP 1
    - 1 VP → . V NP 1
      - 1 V → . ate 1
      - 1 V → ate . 2
    - 1 VP → V . NP 2
      - 2 NP → 2
      - 2 NP → 7
    - 1 VP → V NP . 7
  - 0 S → NP VP . 7

implement by function calls:
S() calls NP() and VP(), which is recursive

must backtrack to try predicting a different VP rule here instead

0 Papa 1 ate 2 the 3 caviar 4 with 5 a 6 spoon 7

First Try: Recursive Descent

ROOT → S	VP → V NP	NP → Papa	V → ate
S → NP VP	VP → VP PP	N → caviar	P → with
NP → Det N	PP → P NP	N → spoon	Det → the
NP → NP PP			Det → a

- 0 ROOT → . S 0
  - 0 S → . NP VP 0
    - 0 NP → . Papa 0
    - 0 NP → Papa . 1
  - 0 S → NP . VP 1
    - 1 VP → . VP PP 1
      - 1 VP → . V NP 1
        - 1 V → . ate 1
        - 1 V → ate . 2
      - 1 VP → V . NP 2
        - 2 NP → 2
        - 2 NP → 4

we'd better backtrack here too!

do some more parsing and eventually ...

... the correct NP is from 2 to 4 this time

0 Papa 1 ate 2 the 3 caviar 4 with 5 a 6 spoon 7

First Try: Recursive Descent

ROOT → S	VP → V NP	NP → Papa	V → ate
S → NP VP	VP → VP PP	N → caviar	P → with
NP → Det N	PP → P NP	N → spoon	Det → the
NP → NP PP			Det → a

- 0 ROOT → . S 0
  - 0 S → . NP VP 0
    - 0 NP → . Papa 0
    - 0 NP → Papa . 1
  - 0 S → NP . VP 1
    - 1 VP → . VP PP 1
      - 1 VP → . VP PP 1
        - 1 VP → . VP PP 1

          1 VP → . VP PP 1

          1 VP → . VP PP 1

stack overflowed

no fix after all

– must transform grammar to eliminate left-recursive rules

Earley's Algorithm

- Earley's algorithm resembles recursive descent, but solves the left-recursion problem. No recursive function calls.
- Use a parse table as we did in CKY, so we can look up anything we've discovered so far. **"Dynamic programming."**

Operation of the Algorithm

- Process a hypothesis according to what follows the dot as in recursive descent:
  - If a word, **scan** input and see if it matches
  - If a nonterminal, **predict** ways to match it (reduce #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

0
0 ROOT . S

initialize

This stands for (0, ROOT → . S)

0
0 ROOT . S
0 S . NP VP

predict the kind of S we are looking for

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	
	1 P . with	

scan: success!

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	**scan**: failure

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	
0 Det . a	1 V . ate	2 Det . a	
	1 P . with		

scan: success!

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	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			

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	

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

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

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

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

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

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

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

0 The 1 old 2 man 3 goes 4 too 5 fast 6

NP → DT NN

NP → DT JJ NN

NP → DT NP

VP → V

VP → V AdjP

VP → VP AdjP

AdjP → RB JJ

S → NP VP

DT → the

JJ → old

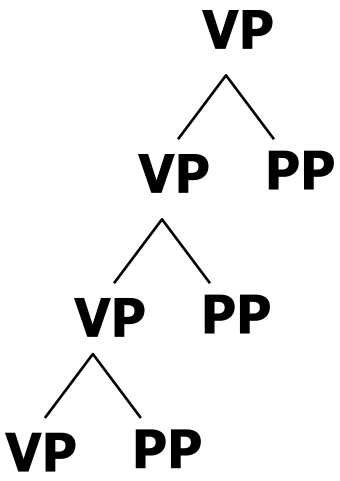
NN → man

V → goes

RB → too

JJ → fast

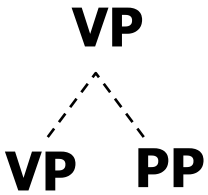
Left Recursion Kills Pure Top-Down Parsing …



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

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

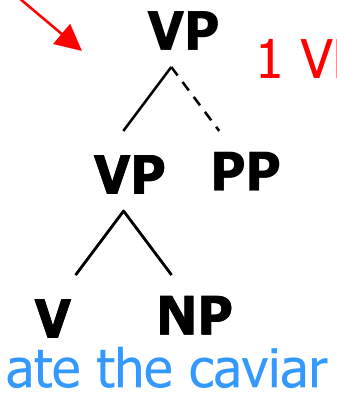
… but Earley's Alg is Ok!



1 VP → . VP PP

(in column 1)

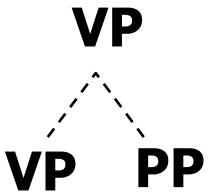
attach



1 VP → VP . PP

(in column 4)

… but Earley's Alg is Ok!

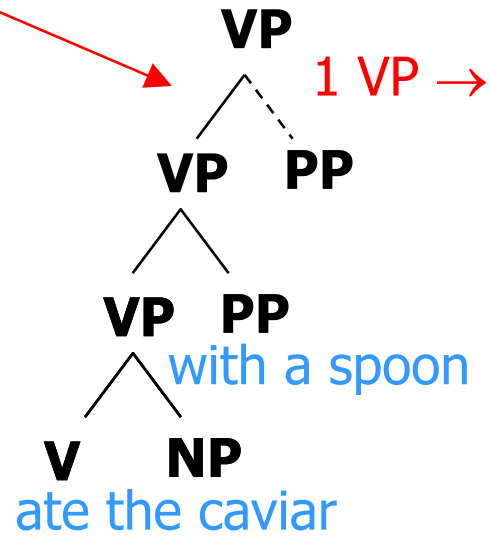


1 VP → . VP PP

can be reused

(in column 1)

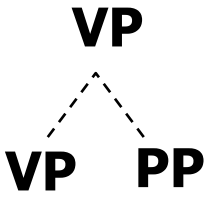
attach



1 VP → VP . PP

(in column 7)

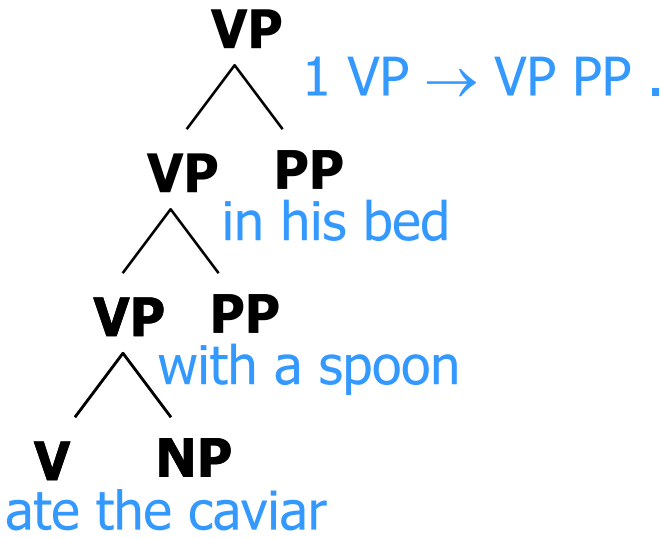
… but Earley's Alg is Ok!



1 VP → . VP PP

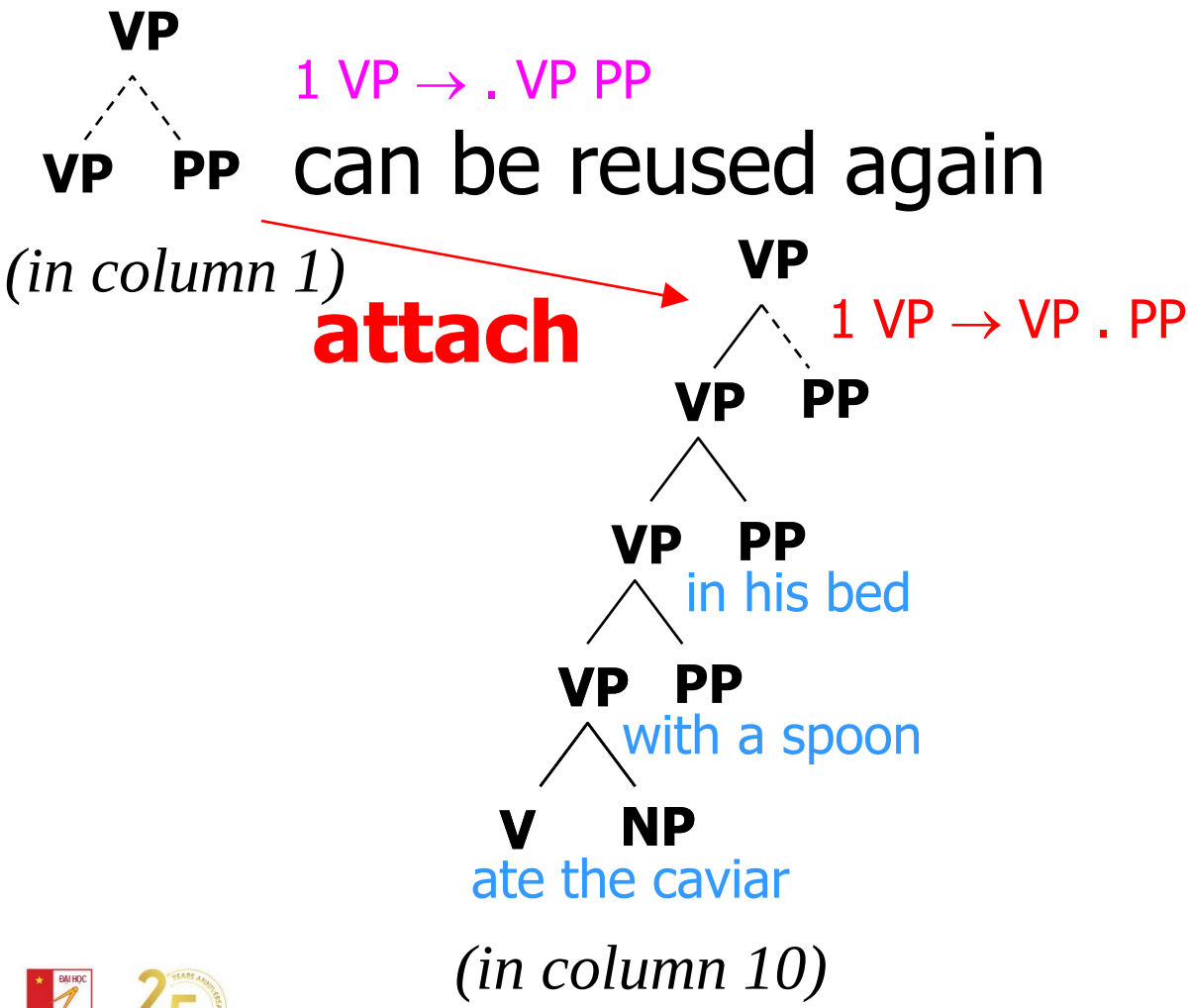
can be reused

(in column 1)



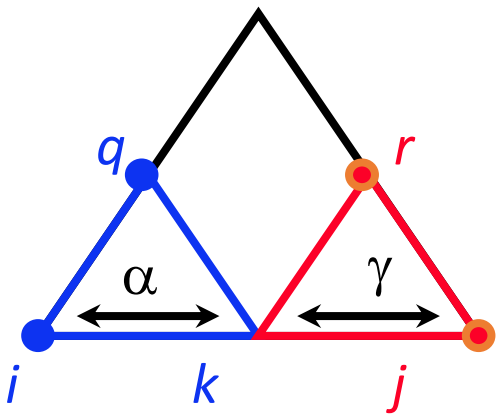
(in column 10)

… but Earley's Alg is Ok!



Recovering parses

[*s,i*] in state set *j*



Use simple queue algorithm, based on useful items

- Any item of form in final state set is useful
- *If* $s=[A \rightarrow \alpha \bullet B, i]$ is in state set k & useful
- *then* $q=[A \rightarrow \alpha B \bullet, k]$ & item $r= [B \rightarrow \gamma \bullet, j]$ are useful

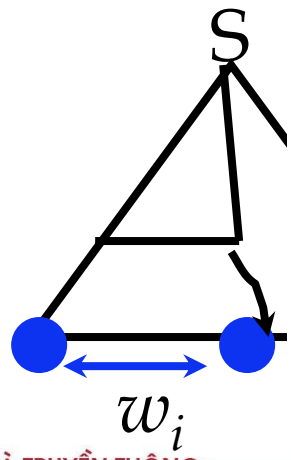
[*s,i*] : an item with a dotted rule *s* & return pointer *i.*

Mark all items in state set *Sn* in the form *Start* $\rightarrow \alpha S \bullet$, 0
for j=n downto 0 do
    for i=0 to *j do*
        for every marked [*s,i*] in state set *j do*
            for k=i to j do
            if $[q,i] \in Sk$ & $[r,k] \in Sj$ & $s= q \otimes r$ *then*
            mark [*q,i*] and [*r,k*]

Advantages

- Earley algorithm does some *top-down* filtering: any item (state, or triple) added to a given State Set must be compatible with *some* derivation on the left, e.g.,

  $S \Rightarrow^* w_i$ where w_i is the sentence seen so far



Disadvantages

- Explicit representation of rules: wastes time building them.
- Does filtering on *left* but not on the *right*

Lookahead filter for nonterminals *A:*

FIRST(A)= $\{x | A \Rightarrow x\delta\}$, *x= 1 token*

e.g., *FIRST(S)= who, did, the,* etc.

Other parsing methods

- Different methods correspond to different ways of search “phrase space”
- Phrase space: X[*i, j*] if phrase of type *X* spans input from *i* to *j.*

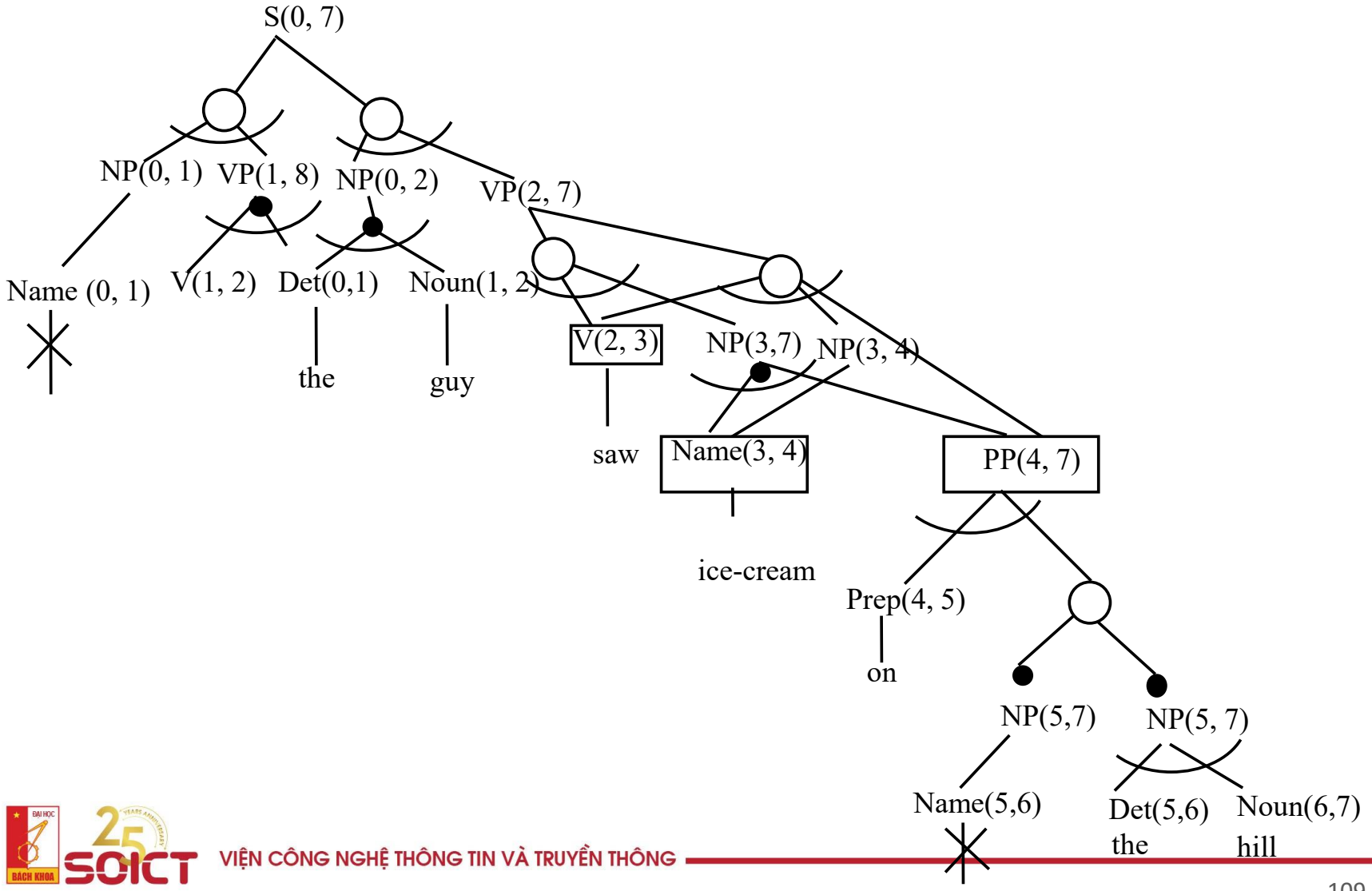
  Example:

  $_0$ *John* $_1$ *ate* $_2$ *ice-cream* $_3$ *on* $_4$ *the* $_5$ *table* $_6$

  PP[3,6]; S[0,6]; …
- Represent search as and-or tree
  - Disjuncts (or) = alternative parse paths
  - Conjuncts (and) = right-hand side of a rule, eg, an S *is a* NP VP

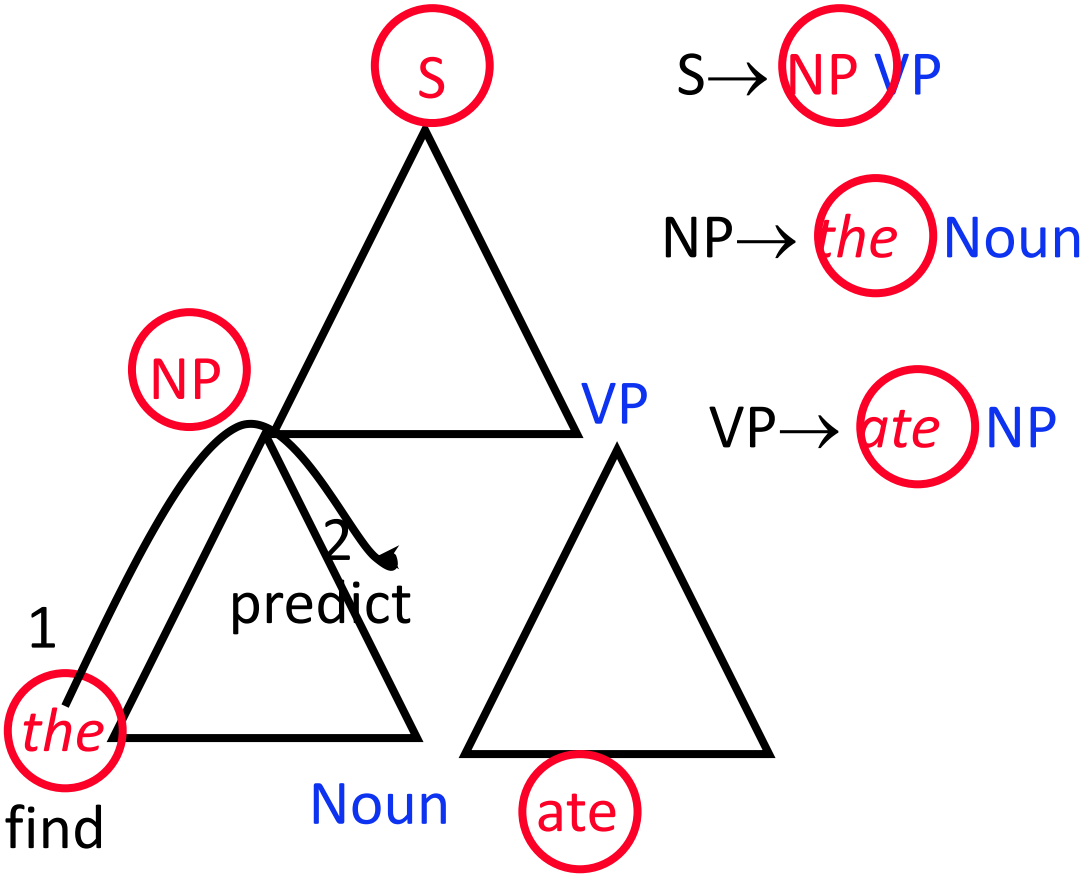
Parsing as search

0 the 1 guy 2 saw 3 ice-cream 4 on 5 the 6 hill 7



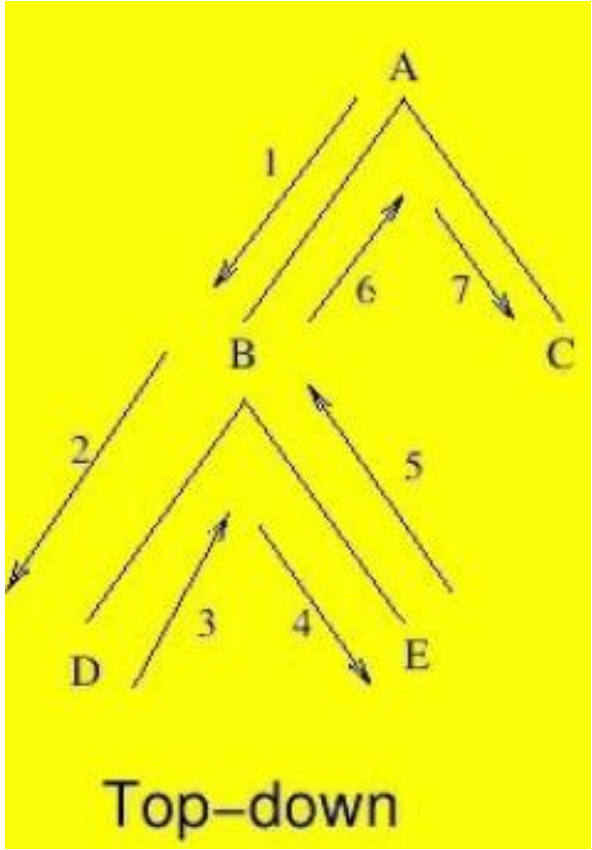
Left-corner parsing

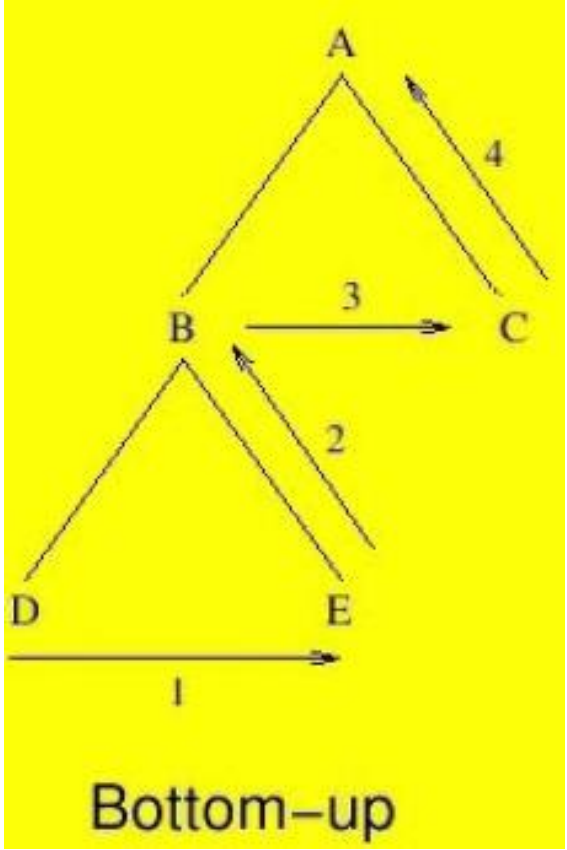
- Looks bottom-up for the *first* symbol (left-corner) of a phrase; and then tries to *confirm* the rest of the phrase top-down
- Tries to combine best features of b-u and t-d

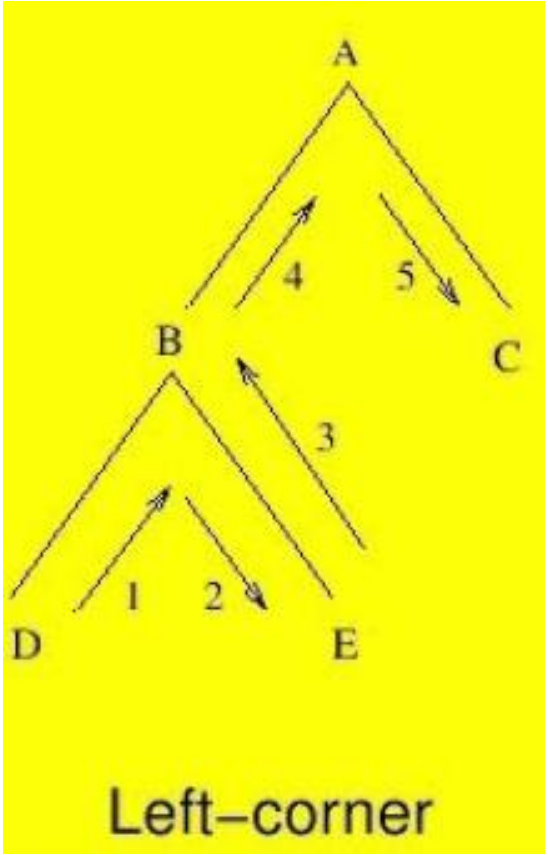


This works well in a head-first language like English. German, Dutch, Japanese are head-final languages

Left-corner parsing







Top down doesn’t care about the input text

Bottom up doesn’t care about what should be built

Left-corner parsing

- Rules:
  - S→ NP VP
  - NP → NN | DT NN | DT NNS | NNP
  - VP → V | V NP
  - V → VBZ | VBP | VBD

- DT → the | a
- NNS → children
- NN → table | song | plant
- NNP → Kate
- VBZ → sings
- VBP → sing | plant
- VBD → died

- Input:
  - Kate sings
  - Kate sings a song
  - The chidren sing a song
  - The plant died

Problems with left-corner parsing

- Some rules with the same “left-corner”
- Ambiguity: cases with 2 possibilities:
  - The new structure can be used to complete a considering structure
  - The new structure can be the starting point of a new larger structure

Context-free grammar

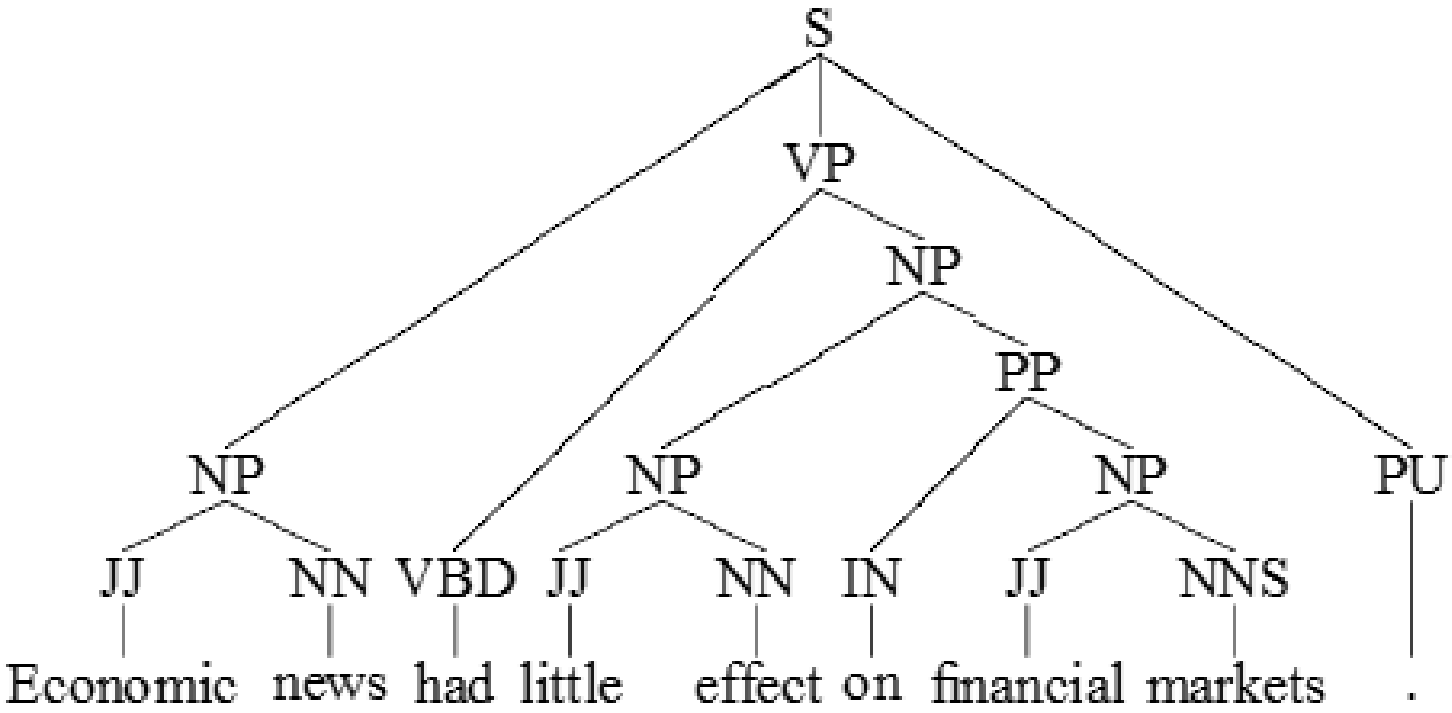


Figure 1: Constituent structure for English sentence from the Penn Treebank

Dependency grammar

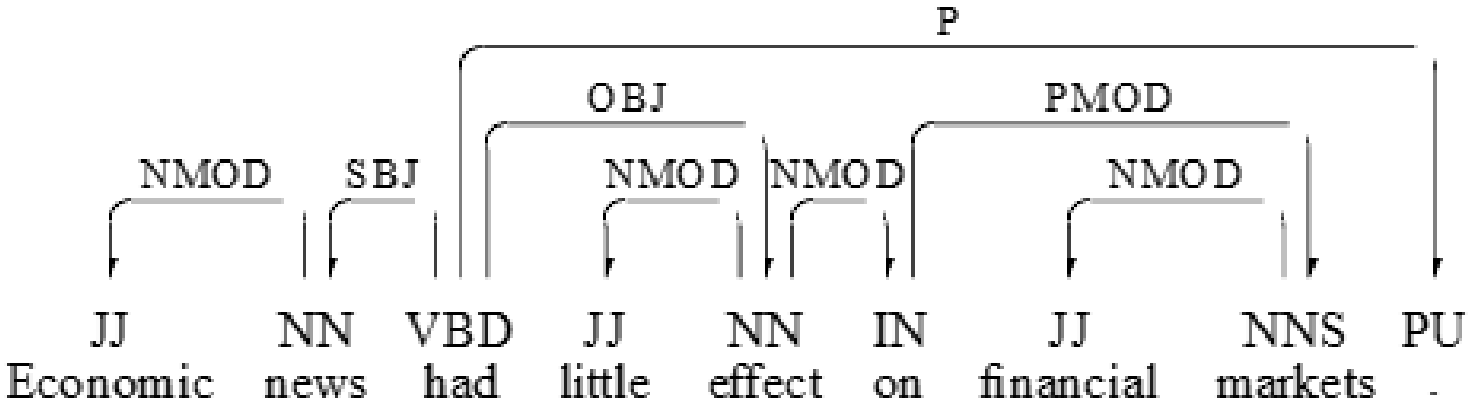


Figure 2: Dependency structure for English sentence from the Penn Treebank