Lecture 15: Context-Free Grammars and the CKY algorithm

Intro to NLP, CS585, Fall 2014

http://people.cs.umass.edu/~brenocon/inlp2014/

Brendan O'Connor

Includes material borrowed from Andrew McCallum, Noah Smith, Dan Klein, Chris Manning, Jurafsky&Martin

CMPSCI 585 In-class Exercise 10/28 (actually 10/30) Name: _____

Fill in the CYK dynamic programming table to parse the sentence below. In the bottom right corner, draw the two parse trees.

	0 she	1 eats	2 fish	3 with	4 chop-sticks	5
0	NP					
1						
2						
3						
4						

S → NP VP
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP

NP → she
NP → fish
NP → fork
NP → chopsticks
V → eats
V → fish
P → with

Two views of syntax

- Constituents: phrase structure

- Dependency structure

Parsing: applications

- Language modeling
  - John, who eats cookies, {love, loves} …
- Machine translation
- Information extraction
- Grammar checking (MS Word!)
- Question answering
- NL interfaces to databases
- Sentiment analysis
- ...

Constituency (phrase structure)

- Phrase structure organizes words into nested constituents.
- How do we know what is a constituent? (Not that linguists don't argue about some cases.)
  - Distribution: a constituent behaves as a unit that can appear in different places:
    - John talked [to the children] [about drugs].
    - John talked [about drugs] [to the children].
    - *John talked drugs to the children about
  - Substitution/expansion/pro-forms:
    - I sat [on the box/right on top of the box/there].
  - Coordination, regular internal structure, no intrusion, fragments, semantics, …

Ambiguity in parsing

- Syntactic ambiguity is widespread in language.
  - Attachment ambiguity
    - we ate sushi with chopsticks
    - I shot an elephant in my pajamas
  - Modifier scope
    - southern food store
  - etc.

Context-Free Grammars

- A generative formalism for constituency structures and text.
  - Generative view: produces a constit. tree and words in sentence.
  - Parsing view: given the words, what parse(s) could have generated it?
- Both boolean and probabilistic versions

Grammar (Production rules)

$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid TWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Lexicon

Figure 13.1 The $\mathscr{L}_1$ miniature English grammar and lexicon.

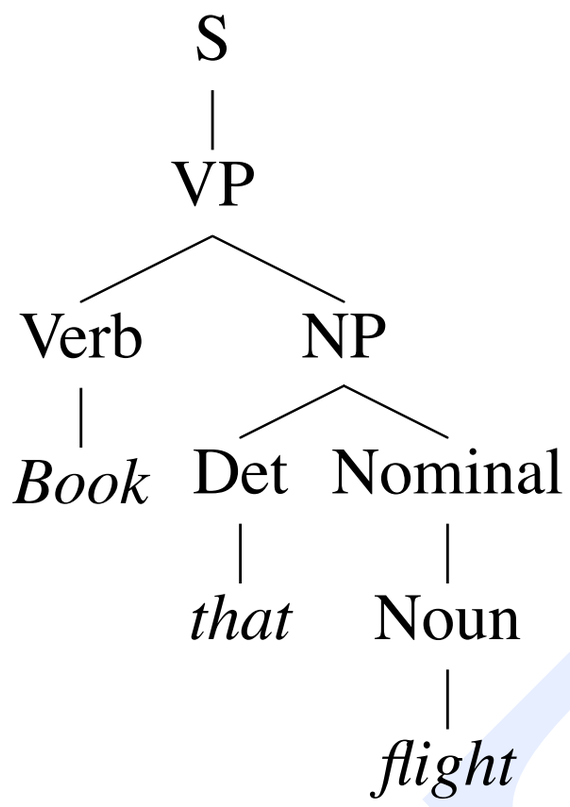
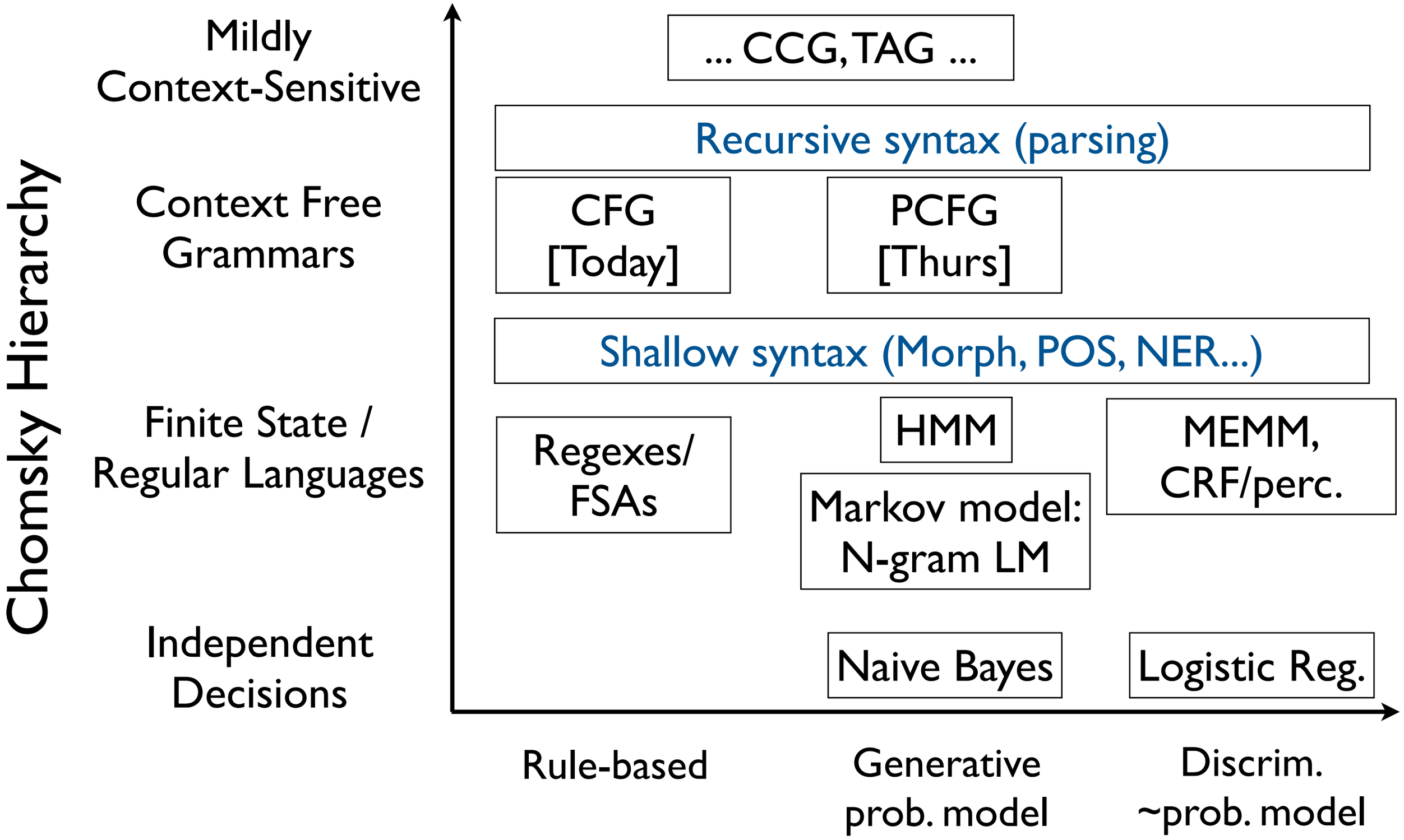


Figure 13.2 The parse tree for the sentence *Book that flight* according to grammar $\mathscr{L}_1$.

Context-Free Grammars

- Unlike programming language grammars: Massive ambiguity!
- Unlike finite state grammars: Potentially infinite recursion

Computation/Statistics in NLP (in this course)



Approaches to CFG parsing

- Top-down and Bottom-up search
- Shift-reduce: left-to-right

- [Today]
  Dynamic programming: CKY algorithm
  Exact search!

- Probabilistic/weighted variants of each of these:
  Find the best parse (e.g. most probable)

Binarized rules

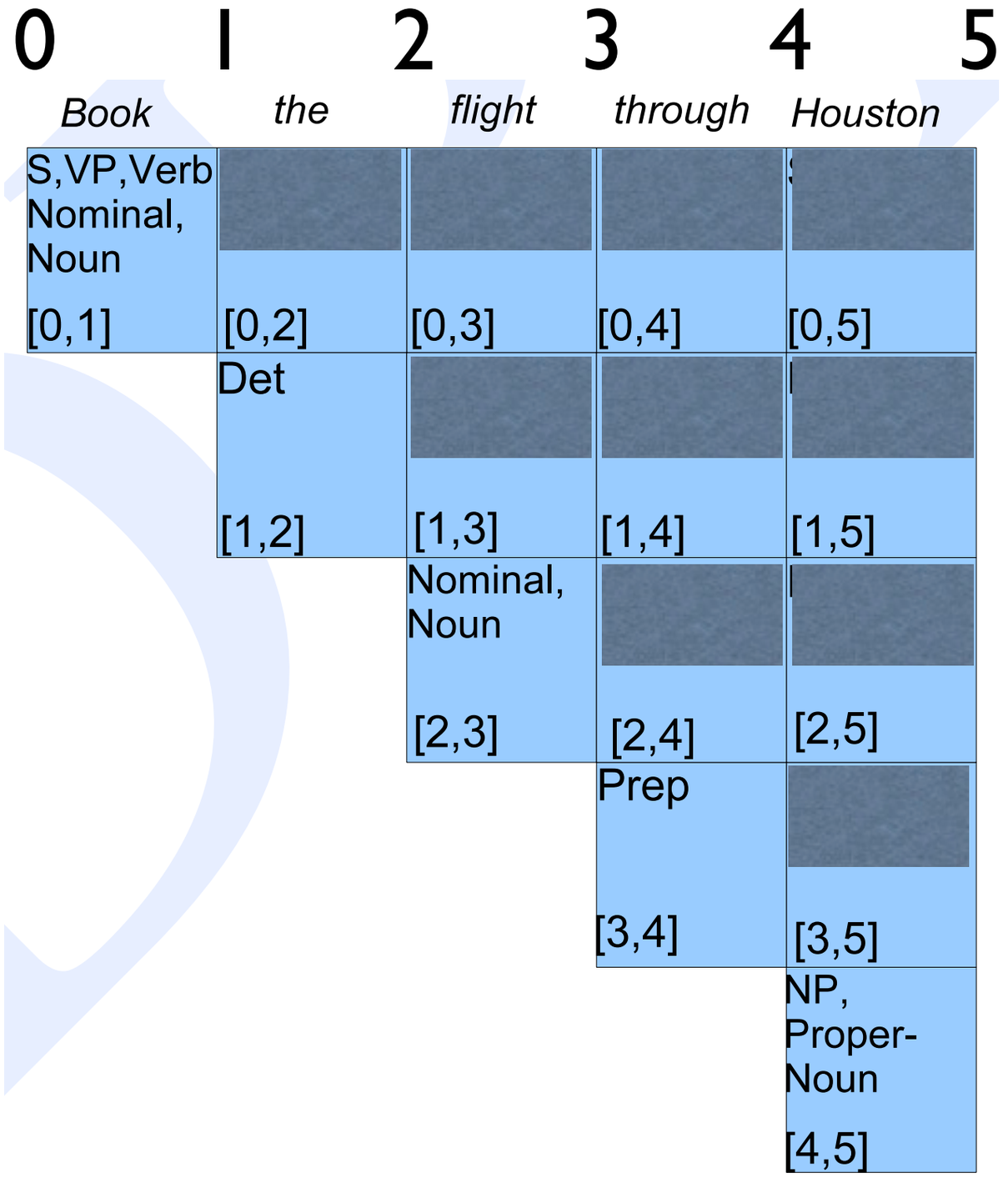
Necessary for CKY algorithm
Can convert to equivalent binarized grammar

$$S \rightarrow NP\ VP$$
$$S \rightarrow Aux\ NP\ VP$$

$$\left\| \begin{array}{l} S \rightarrow NP\ VP \\ S \rightarrow X1\ VP \\ X1 \rightarrow Aux\ NP \end{array} \right.$$

CKY recognizer

- Fill in all length-1 spans with possible nonterminals.
- Go bottom-up: progressively fill each cell with possible states, based on possible combinations below.
- If the top cell [0,5] can expand from ROOT, then accept!
- To get one of possible parses: trace backpointers
- Dynamic programming: what's below the cell does not matter



(J&M has a slightly different cell ordering. Both OK.)

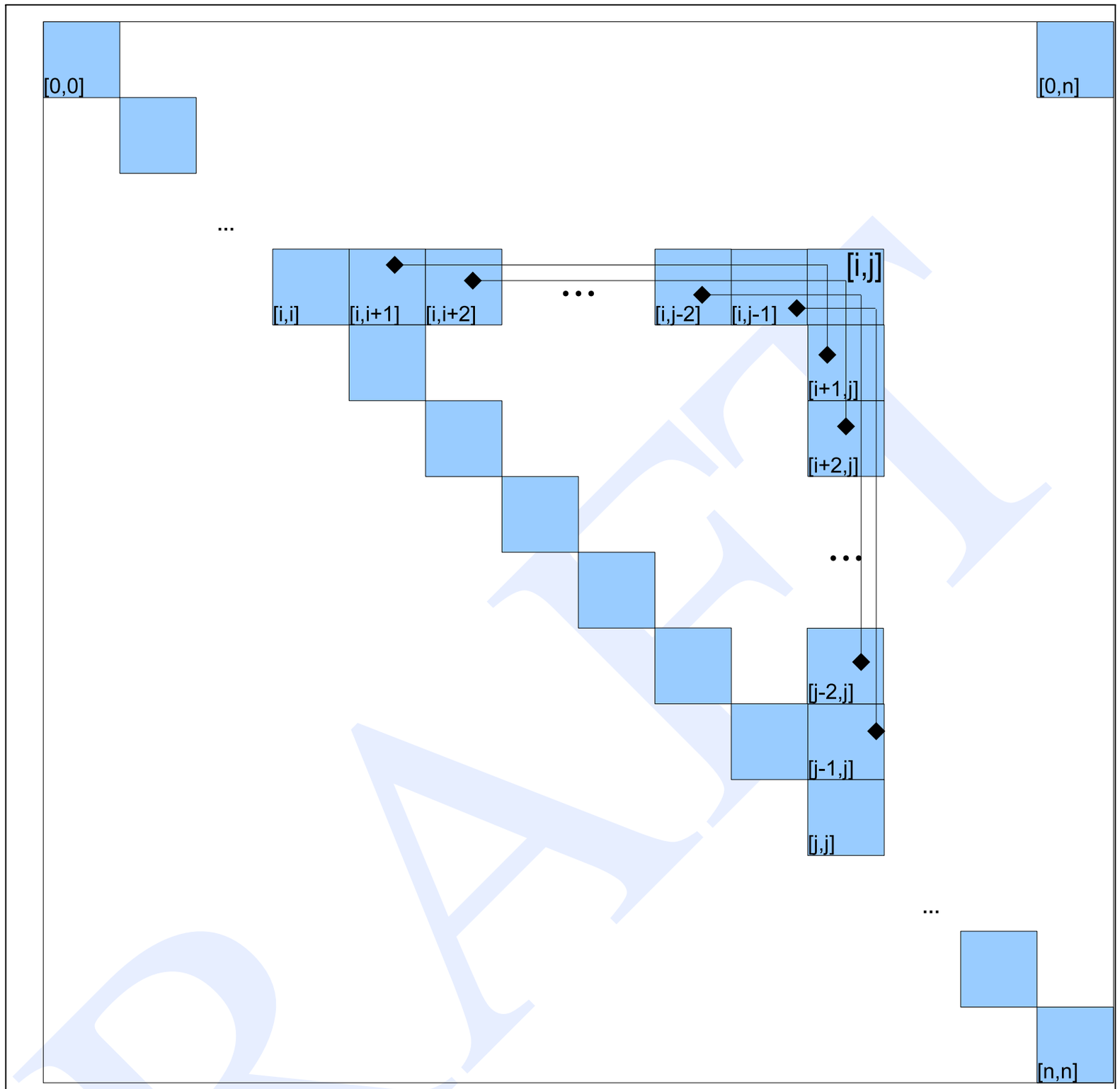
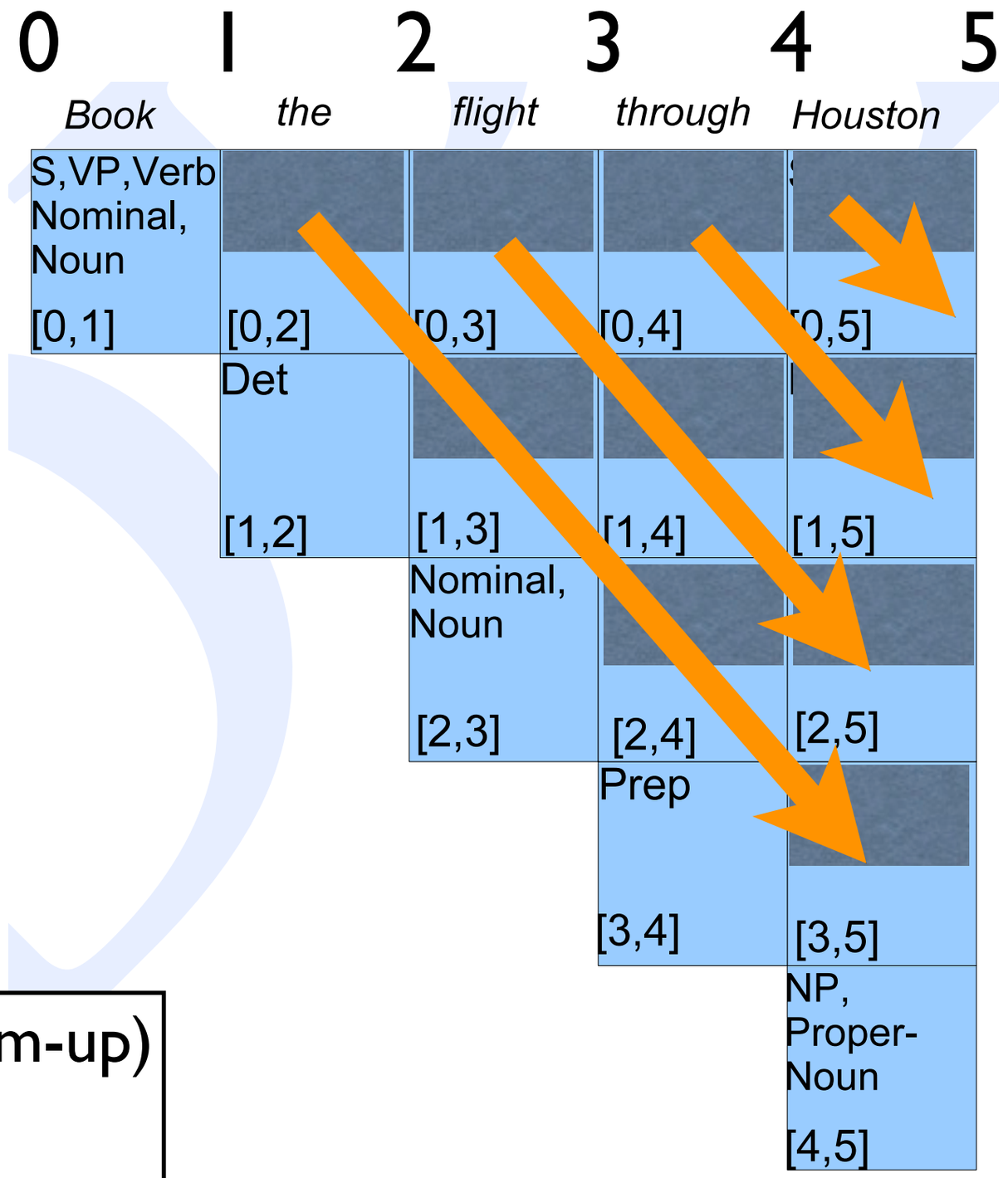


Figure 13.11 All the ways to fill the [i,j]th cell in the CKY table.

CKY recognizer

$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

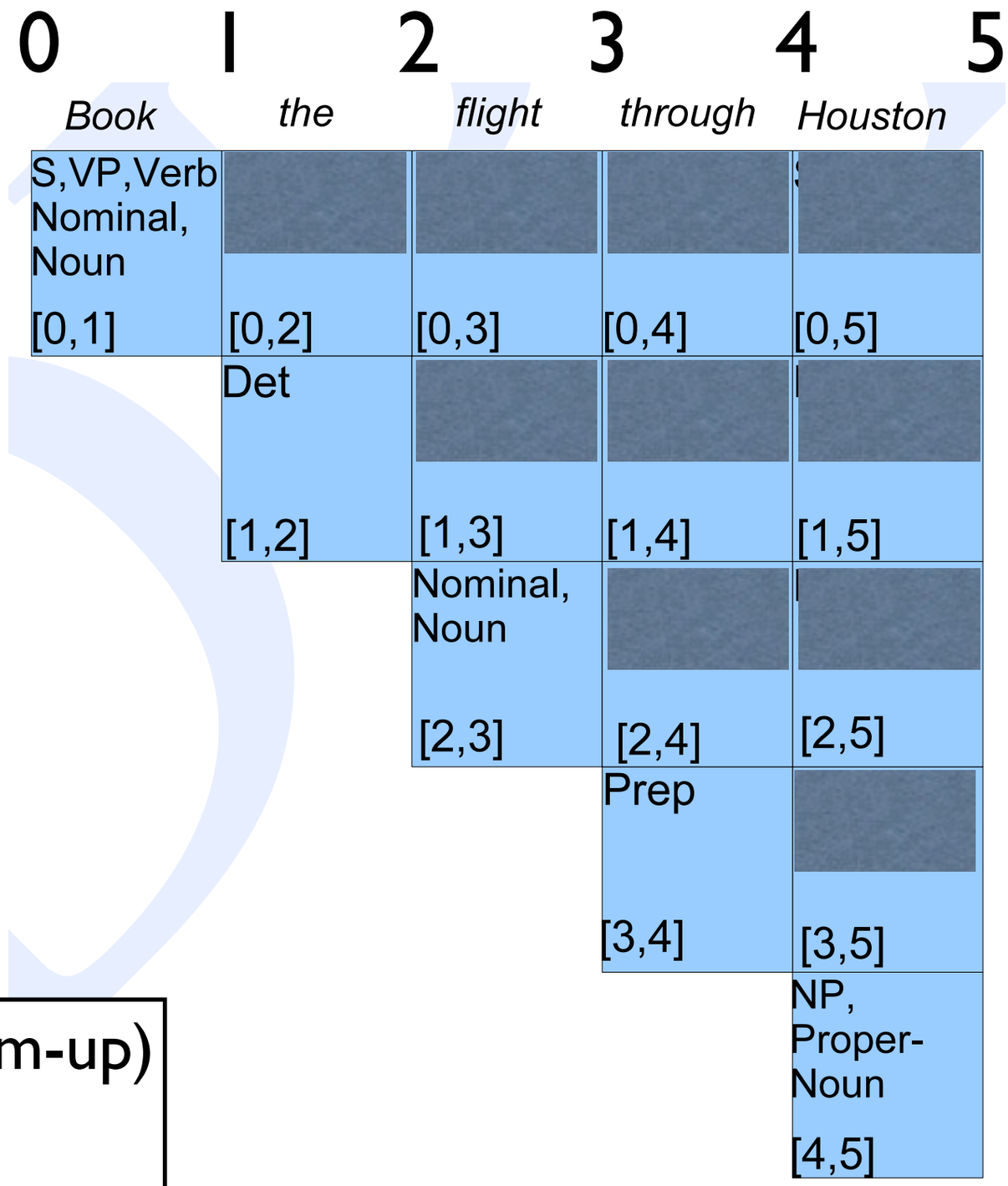


For cell [i,j] (loop through them bottom-up)
  For possible splitpoint k=(i+1)..(j-1):
    For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
        add A to cell [i,j]

(J&M has a slightly different cell ordering. Both OK.)

CKY recognizer

$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$



For cell [i,j] (loop through them bottom-up)
  For possible splitpoint k=(i+1)..(j-1):
    For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
        add A to cell [i,j]

(J&M has a slightly different cell ordering. Both OK.)

CKY recognizer

$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$
$S \rightarrow VP$
$NP \rightarrow Pronoun$
$NP \rightarrow Proper\text{-}Noun$
$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$
$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$
$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$

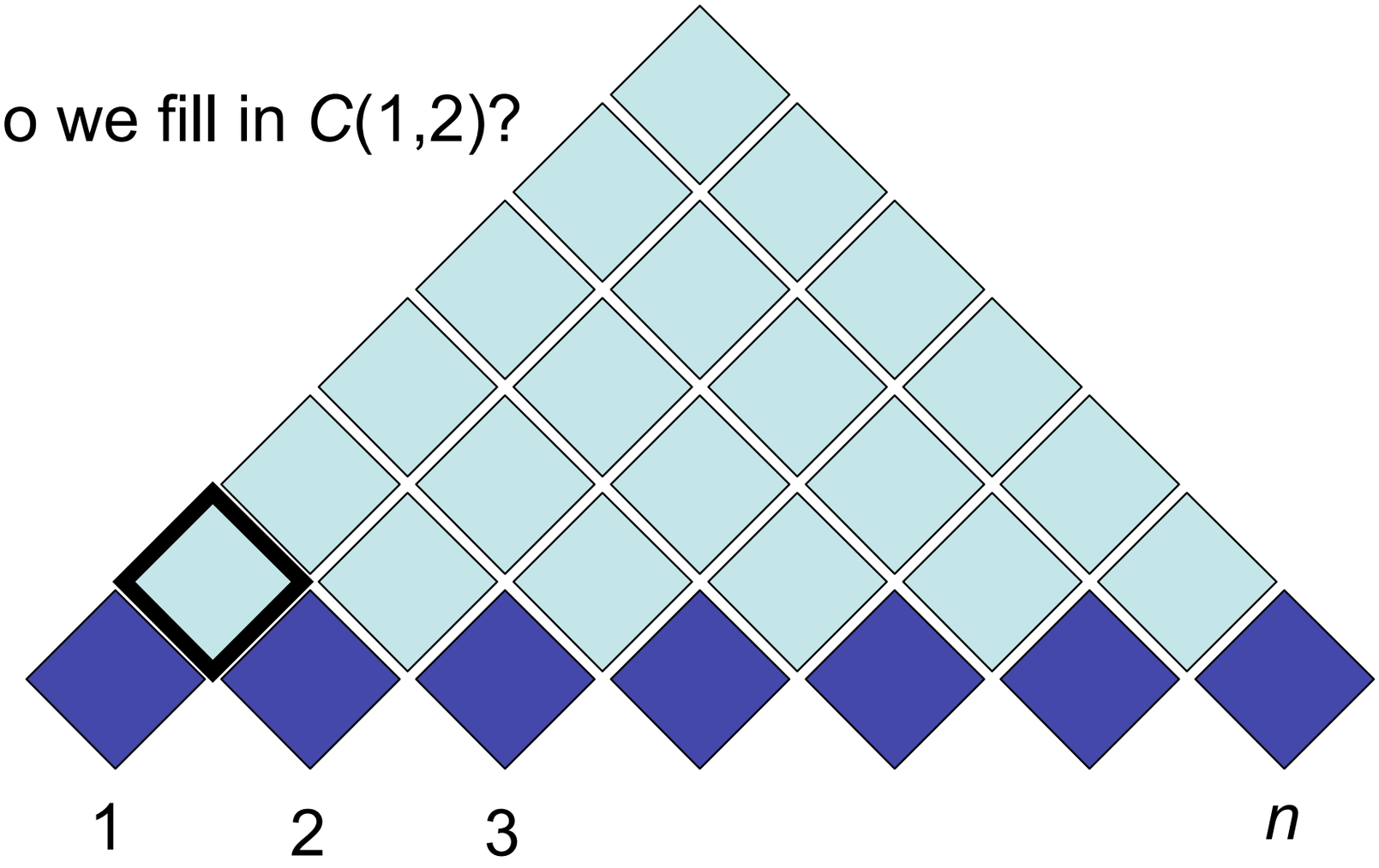
0	1	2	3	4	5
Book	*the*	*flight*	*through*	*Houston*	
S,VP,Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S, VP [0,5]	
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]	
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]	
			Prep [3,4]	PP [3,5]	
				NP, Proper-Noun [4,5]	

For cell [i,j] (loop through them bottom-up)
  For possible splitpoint k=(i+1)..(j-1):
    For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
        add A to cell [i,j]

(J&M has a slightly different cell ordering. Both OK.)

For cell [i,j] (loop through them bottom-up)
  For possible splitpoint k=(i+1)..(j-1):
    For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
        add A to cell [i,j]

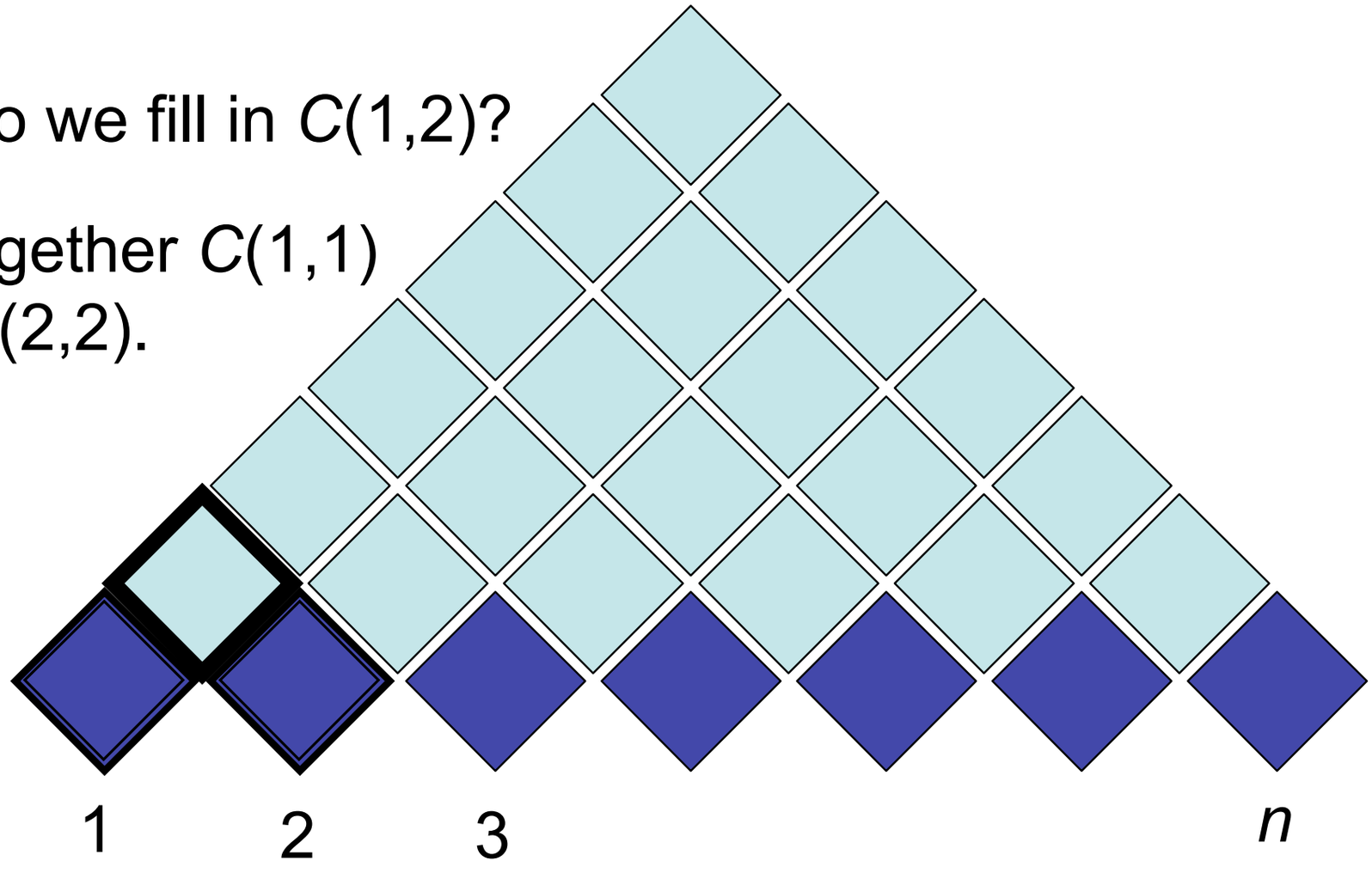
How do we fill in *C*(1,2)?



```
For cell [i,j]  (loop through them bottom-up)
    For possible splitpoint k=(i+1)..(j-1):
        For every B in [i,k] and C in [k,j],
            If exists rule A -> B C,
                add  A to cell [i,j]
```

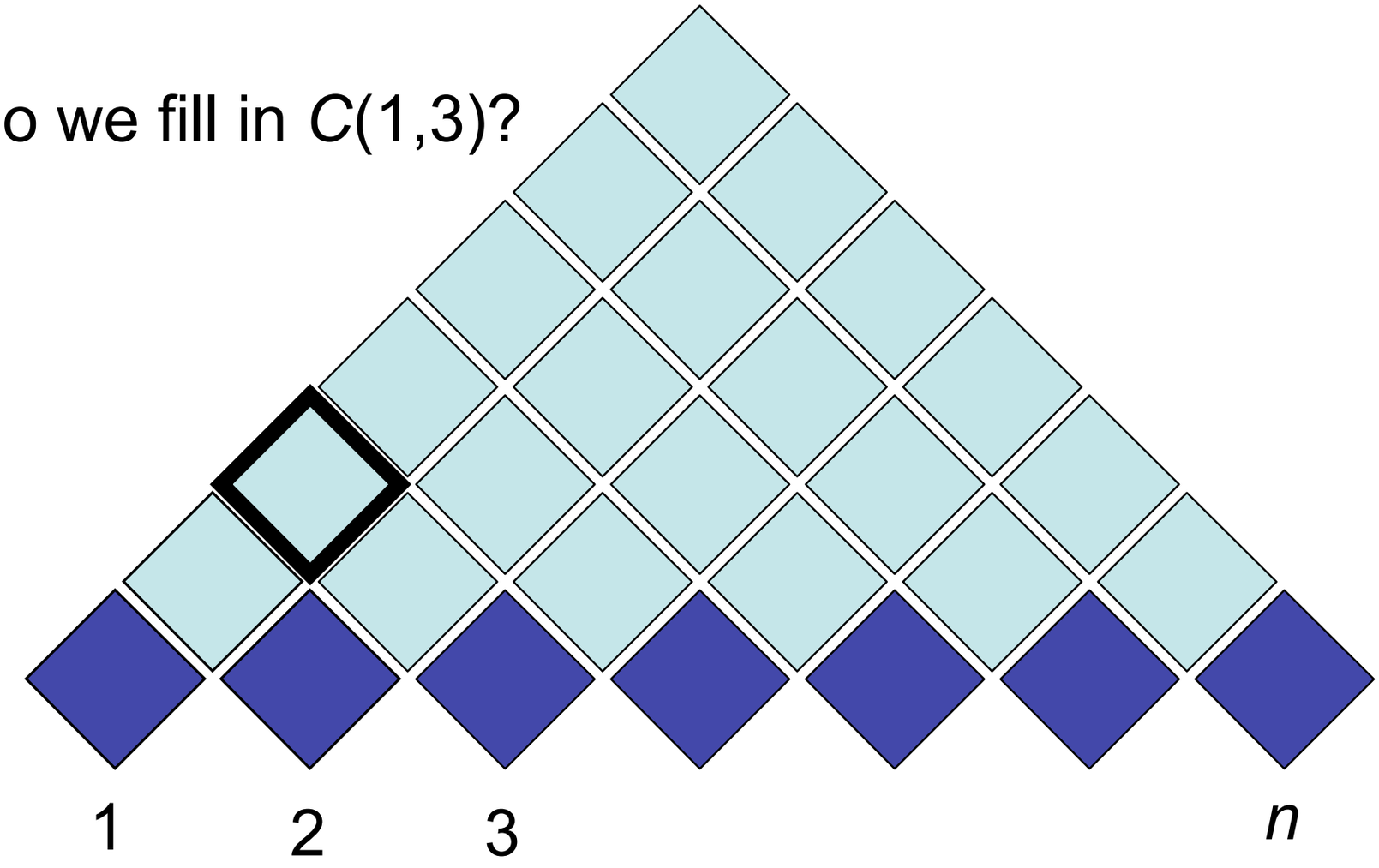
How do we fill in *C*(1,2)?

Put together *C*(1,1) and *C*(2,2).



For cell [i,j] (loop through them bottom-up)
  For possible splitpoint k=(i+1)..(j-1):
    For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
        add A to cell [i,j]

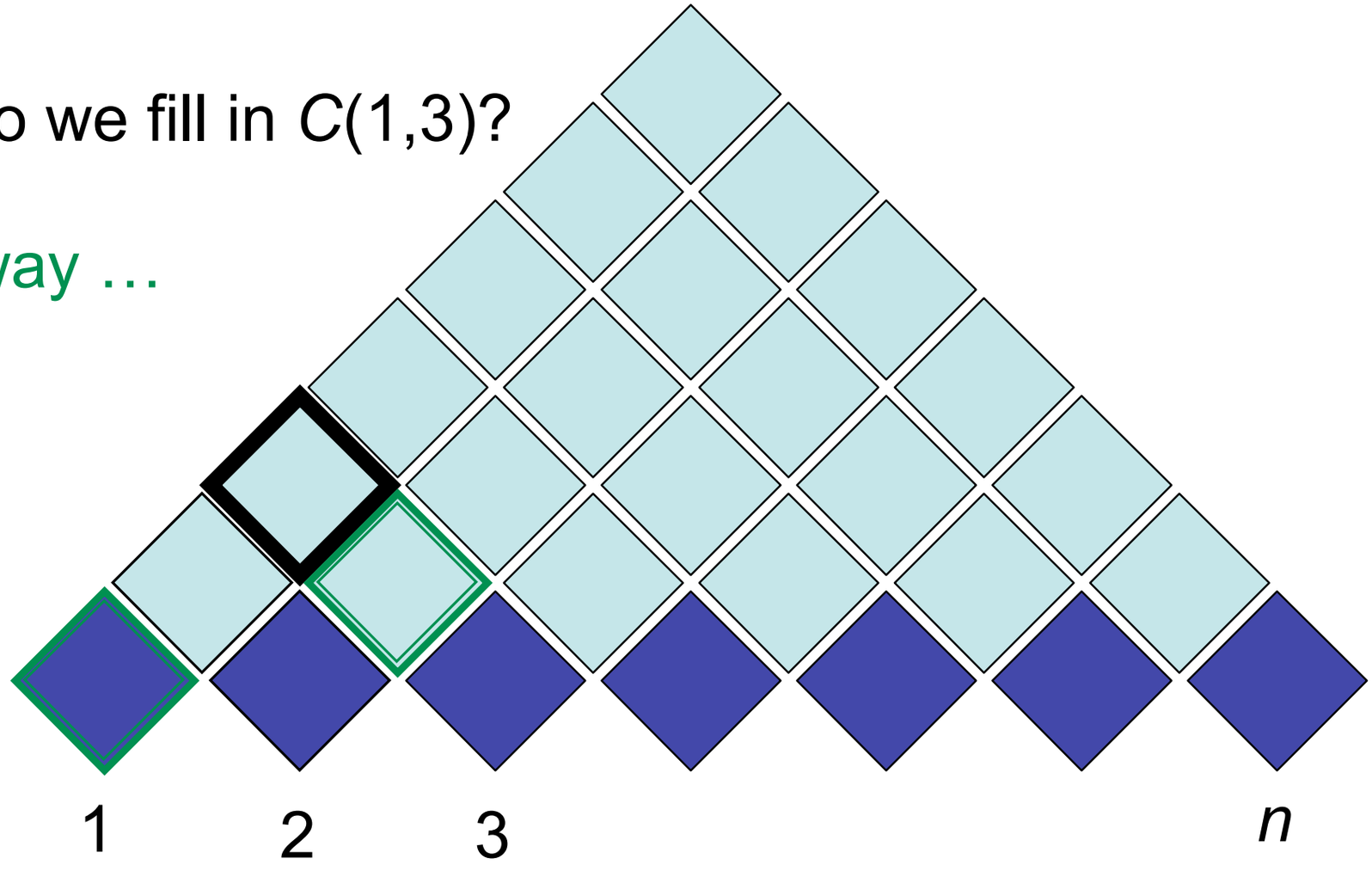
How do we fill in *C*(1,3)?



For cell [i,j] (loop through them bottom-up)
  For possible splitpoint k=(i+1)..(j-1):
    For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
        add A to cell [i,j]

How do we fill in *C*(1,3)?

One way …

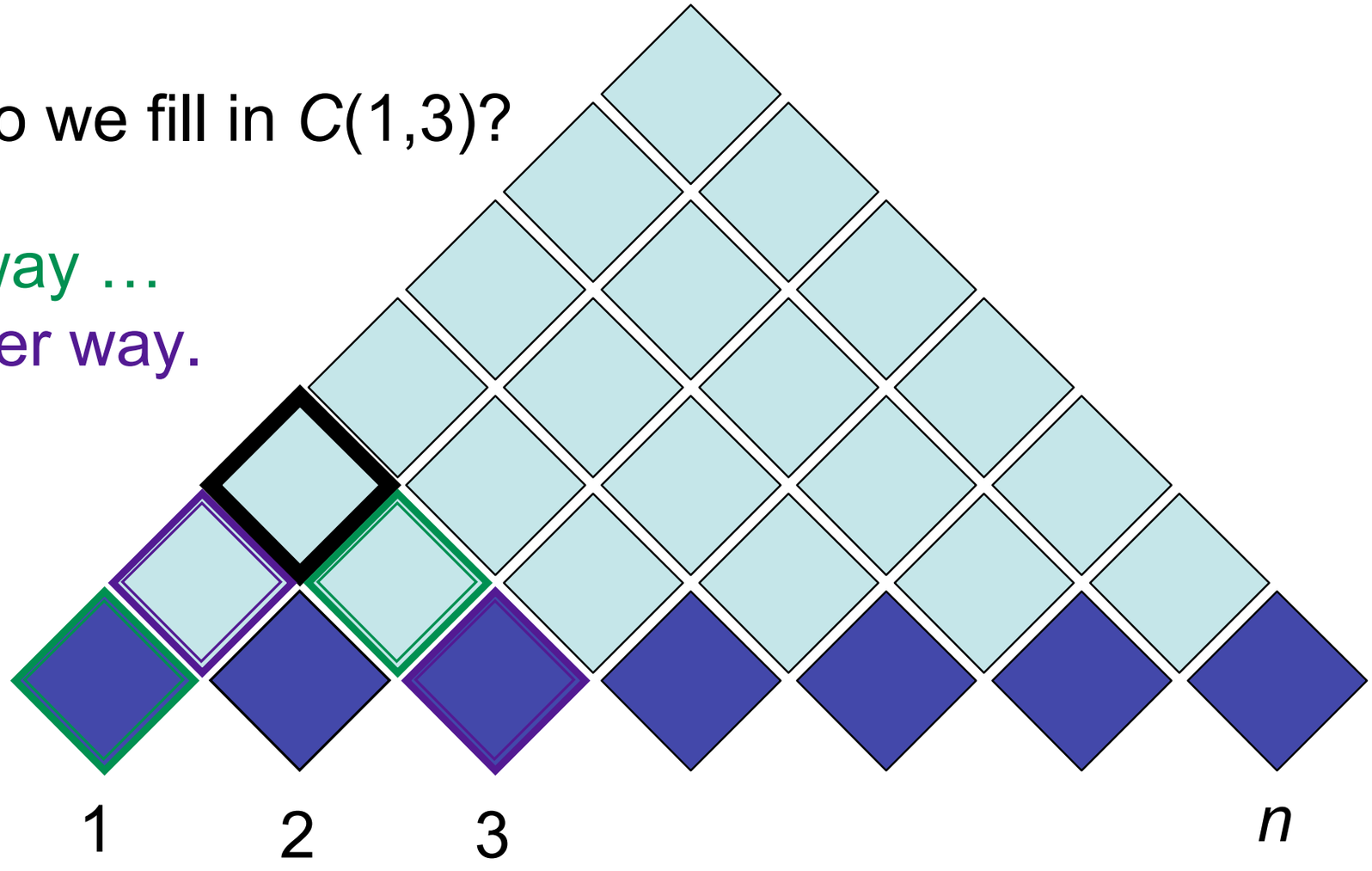


```
For cell [i,j]  (loop through them bottom-up)
    For possible splitpoint k=(i+1)..(j-1):
        For every B in [i,k] and C in [k,j],
            If exists rule A -> B C,
                add  A to cell [i,j]
```

How do we fill in *C*(1,3)?

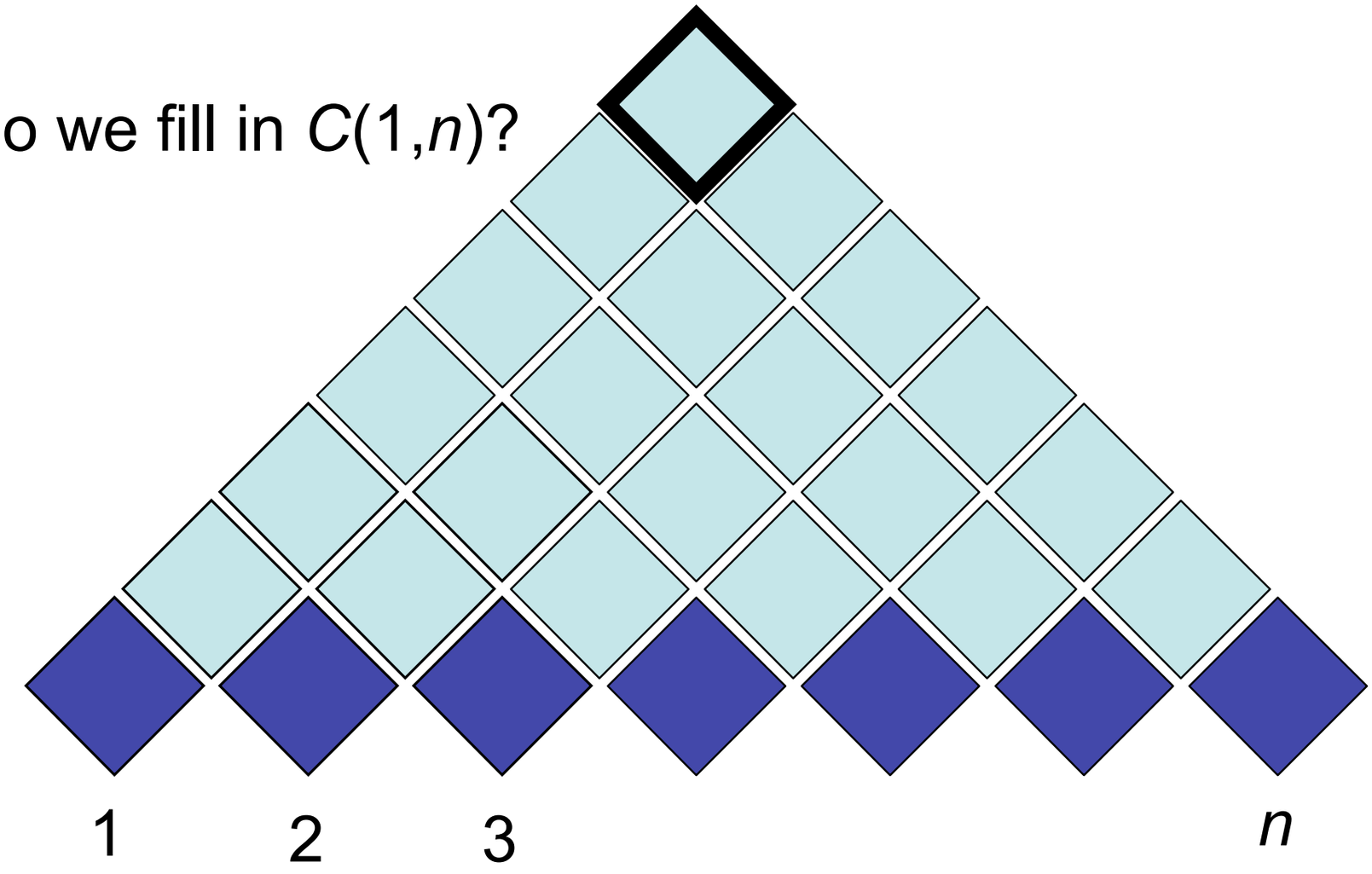
One way …

Another way.



```
For cell [i,j]  (loop through them bottom-up)
    For possible splitpoint k=(i+1)..(j-1):
        For every B in [i,k] and C in [k,j],
            If exists rule A -> B C,
                add  A to cell [i,j]
```

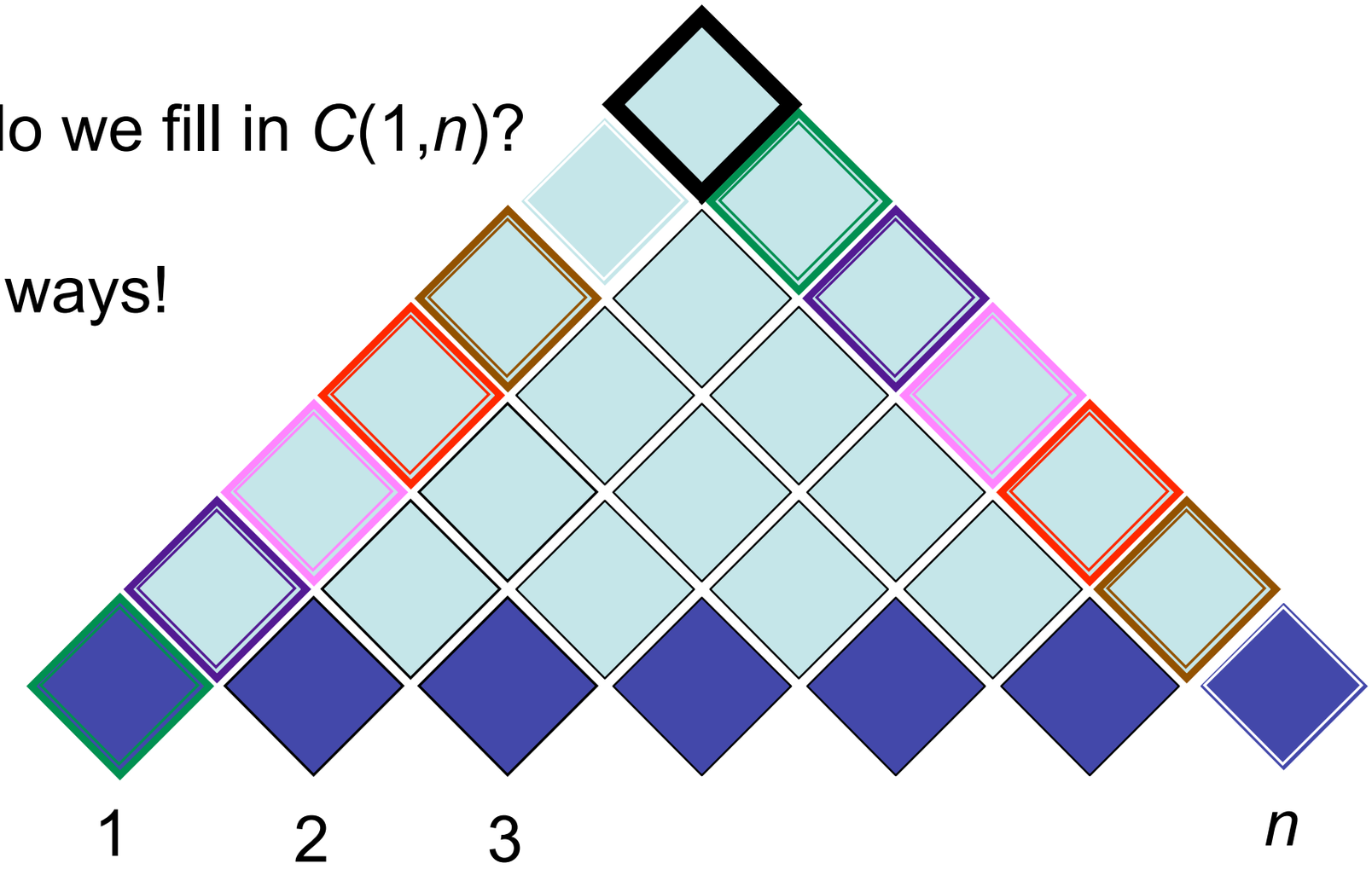
How do we fill in *C*(1,*n*)?



For cell [i,j] (loop through them bottom-up)
  For possible splitpoint k=(i+1)..(j-1):
    For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
        add A to cell [i,j]

How do we fill in *C*(1,*n*)?

n - 1 ways!

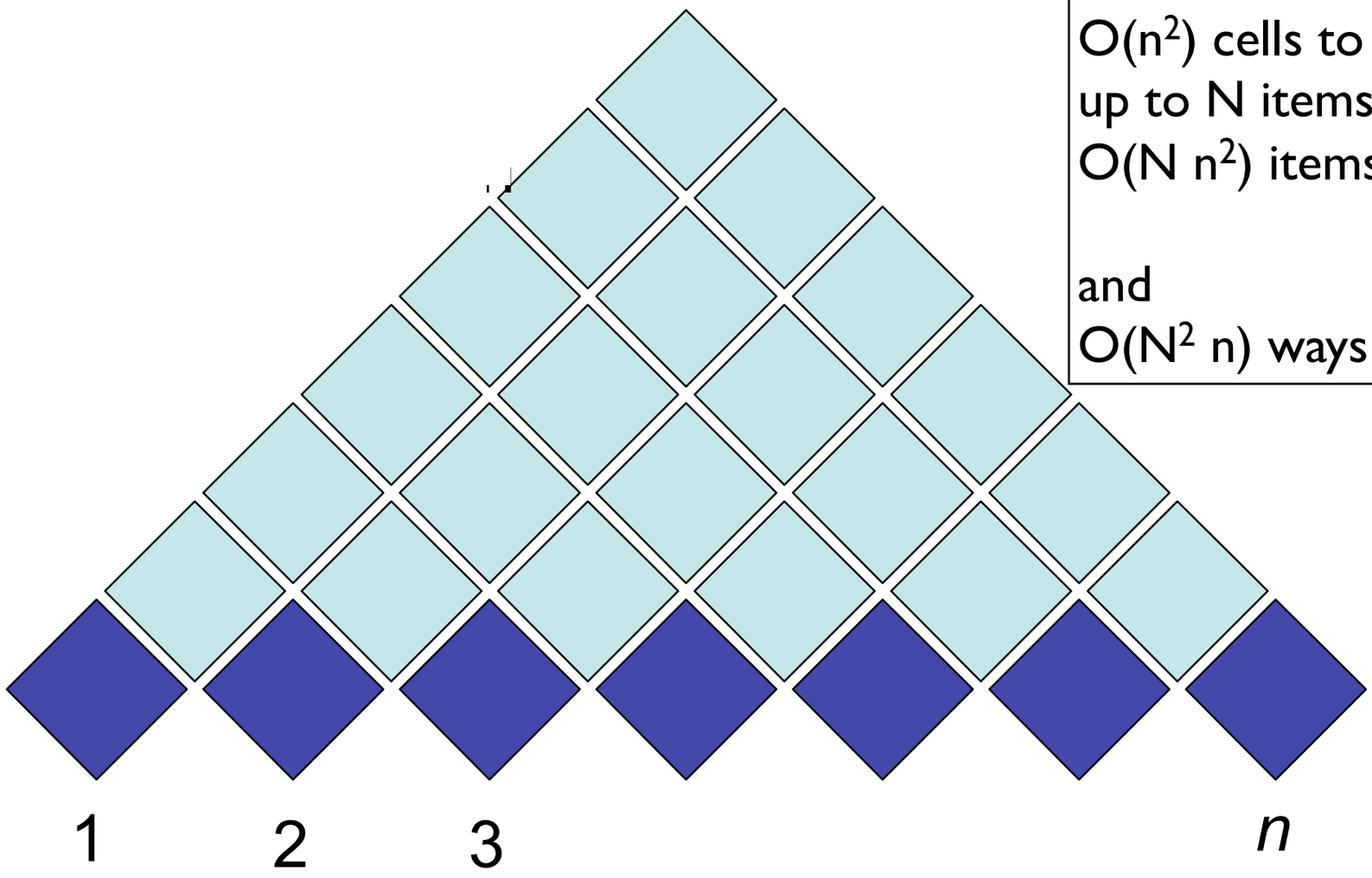


For cell [i,j] (loop through them bottom-up)
  For possible splitpoint k=(i+1)..(j-1):
    For every B in [i,k] and C in [k,j],
      If exists rule A -> B C,
        add A to cell [i,j]

Where N is the number of nonterminals in the grammar, and n is the length of the sentence

$O(n^2)$ cells to fill
up to N items per cell, thus
$O(N\ n^2)$ items to create

and
$O(N^2\ n)$ ways to fill a cell



Example with cost weights per expansion (Weighted CKY, a.k.a. Viterbi parsing)

	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

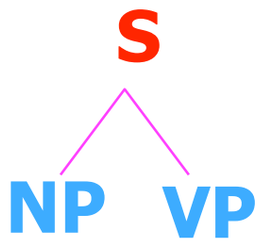
Follow backpointers ...

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**

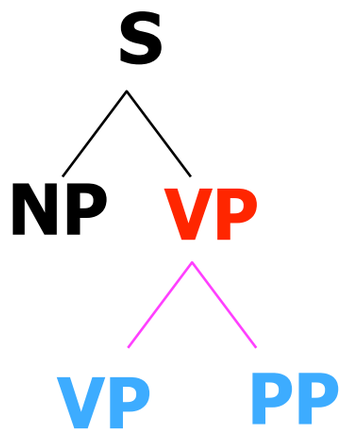
1 S → NP VP
6 S → Vst NP
2 S → S PP
1 VP → V NP
2 VP → VP PP
1 NP → Det N
2 NP → NP PP
3 NP → NP NP
0 PP → P NP



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

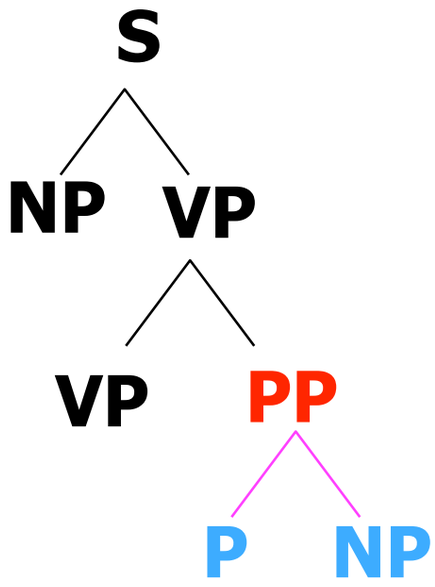
2 NP → NP PP

3 NP → NP NP

0 PP → P NP

time 1 flies 2 like 3 an 4 arrow 5

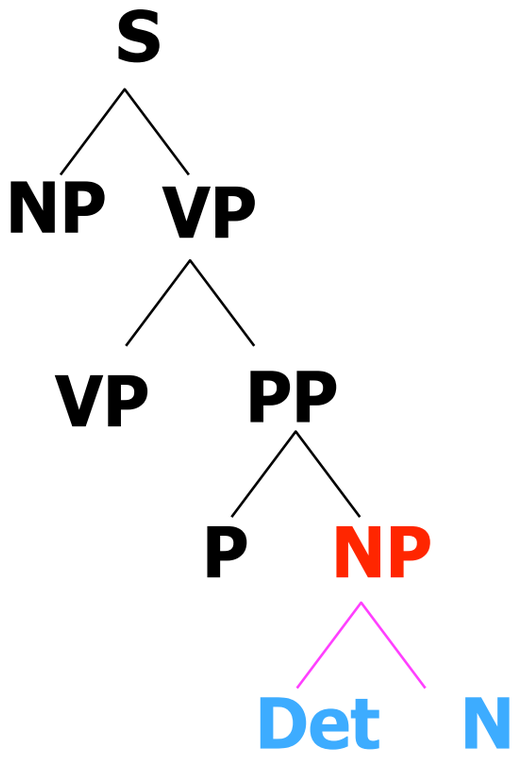
0	NP 3 Vst 3	NP 10 S 8 S 13	_	_	NP 24 S 22 S 27 NP 24 S 27 S 22 S 27
1		NP 4 VP 4	_	_	NP 18 S 21 VP 18
2			P 2 V 5	_	PP 12 VP 16
3				Det 1	NP 10
4					N 8



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