CFG Parsing (3/7)

CS 690N, Spring 2017

Advanced Natural Language Processing

http://people.cs.umass.edu/~brenocon/anlp2017/

Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

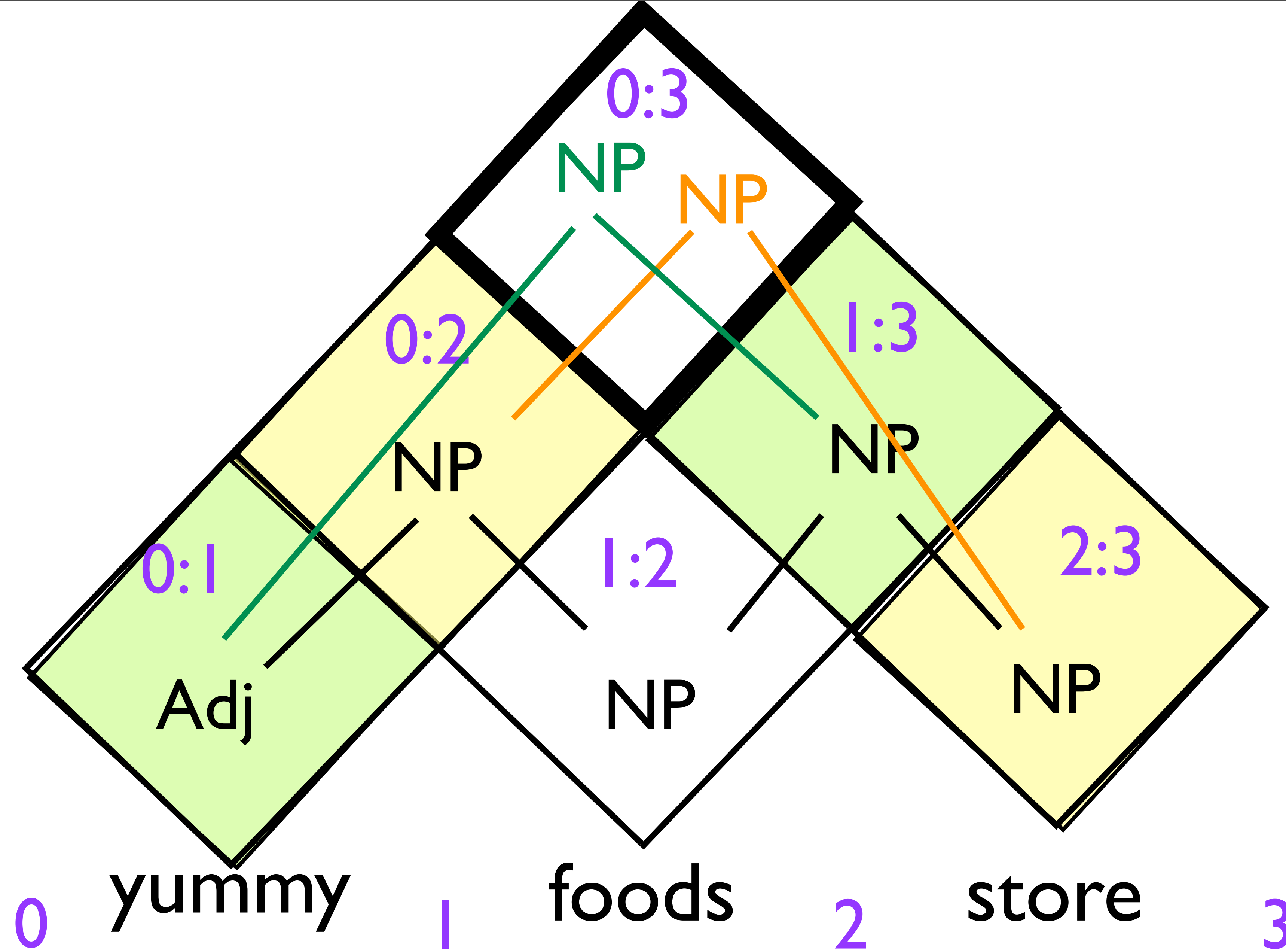
- Types of (P)CFG parsing algorithms
  - Top-down
  - Left-to-right
  - Bottom-up: CKY algorithm

- Naive approach: Number of parses is Catalan number in length!

$$C_n = \frac{(2n)!}{(n+1)!n!}$$

CKY

Grammar
Adj -> yummy
NP -> foods
NP -> store
NP -> NP NP
NP -> Adj 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] (*Recognizer*)
        ... or ...
        add (A,B,C, k) to cell [i,j] (*Parser*)

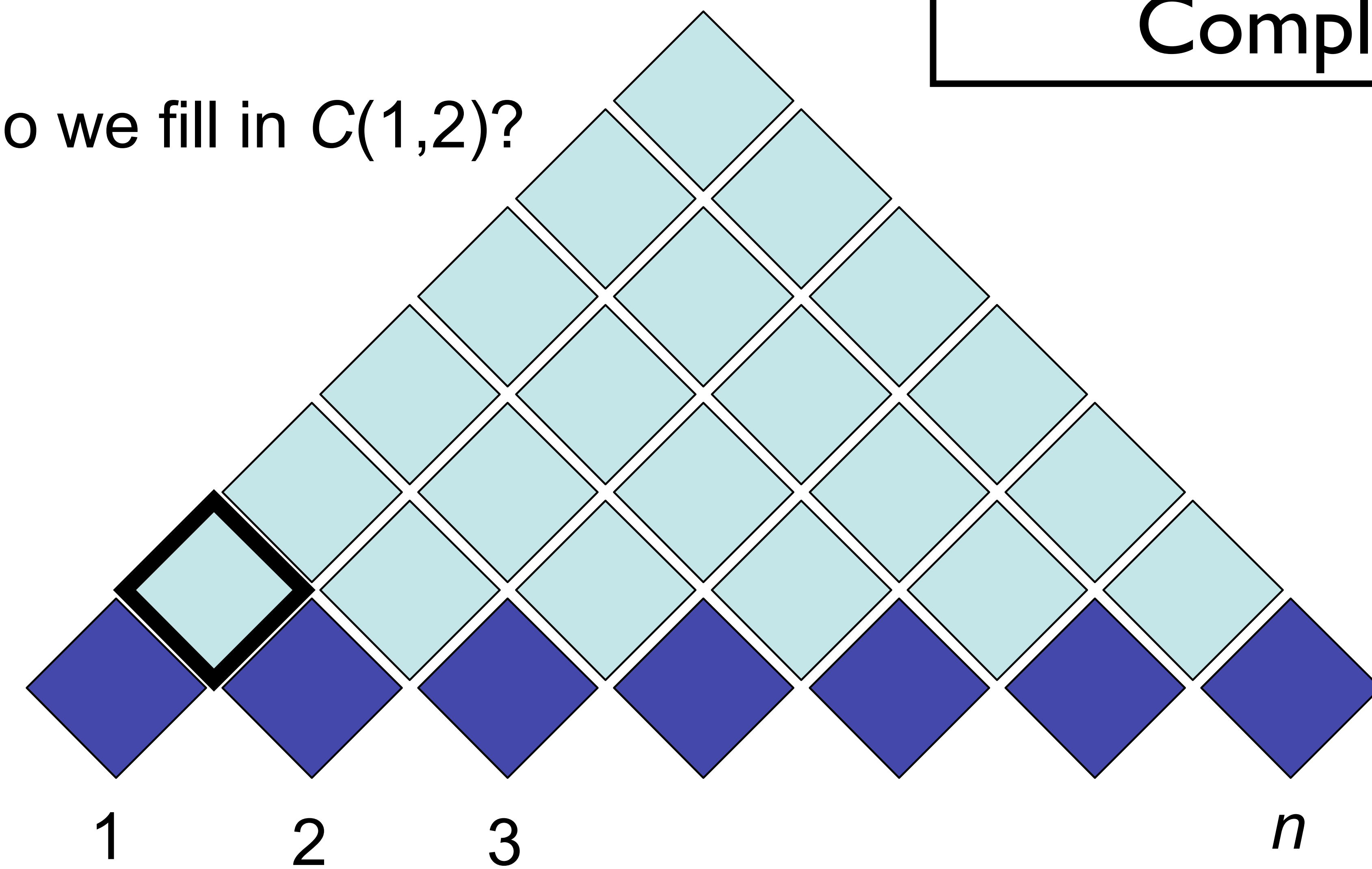
Recognizer: per span, record list of possible nonterminals

Parser: per span, record possible ways the nonterminal was constructed.

For cell [i,j]
    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]

Computational Complexity ?

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



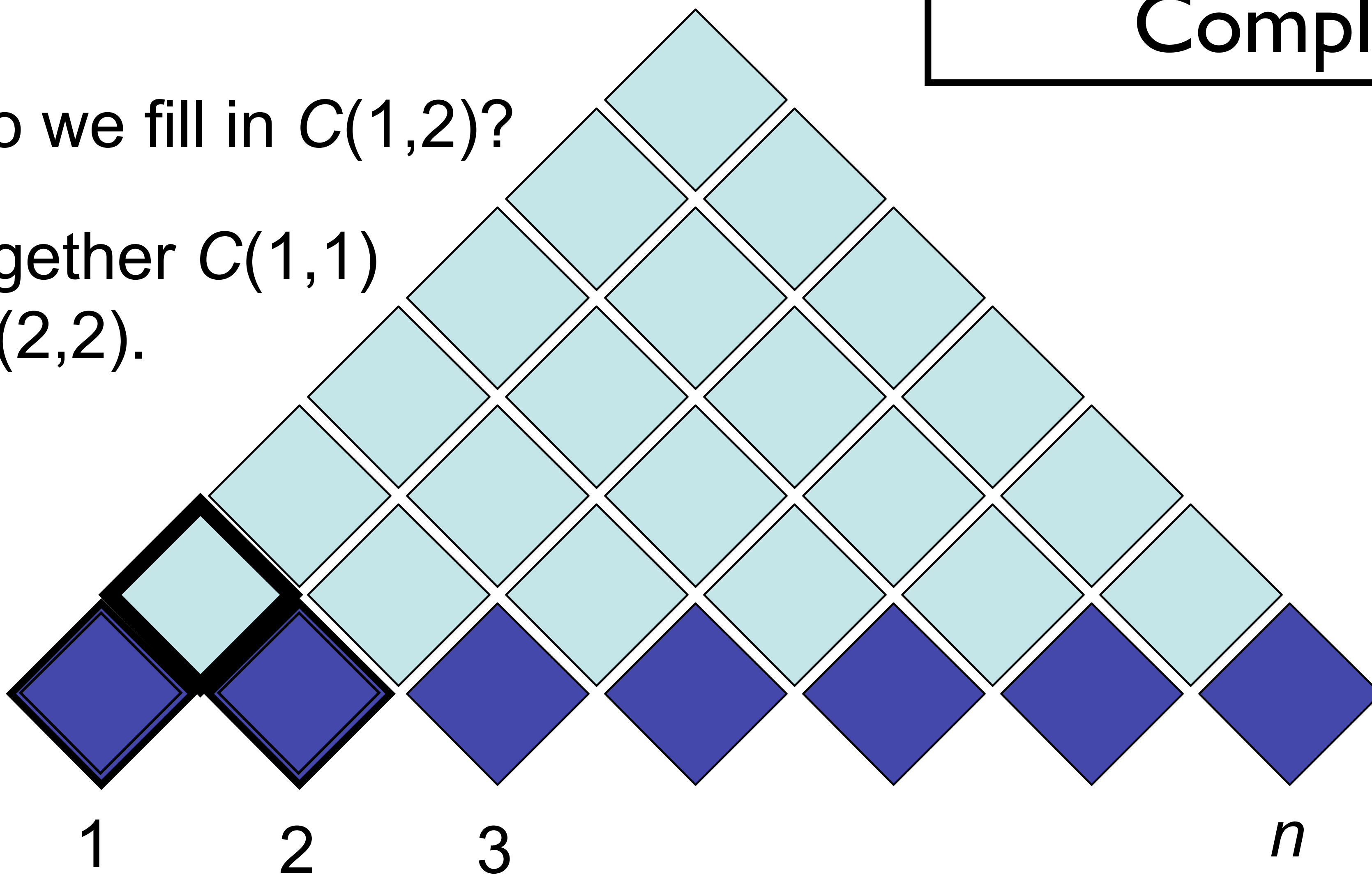
[Example from Noah Smith]

For cell [i,j]
    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]

Computational Complexity ?

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

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

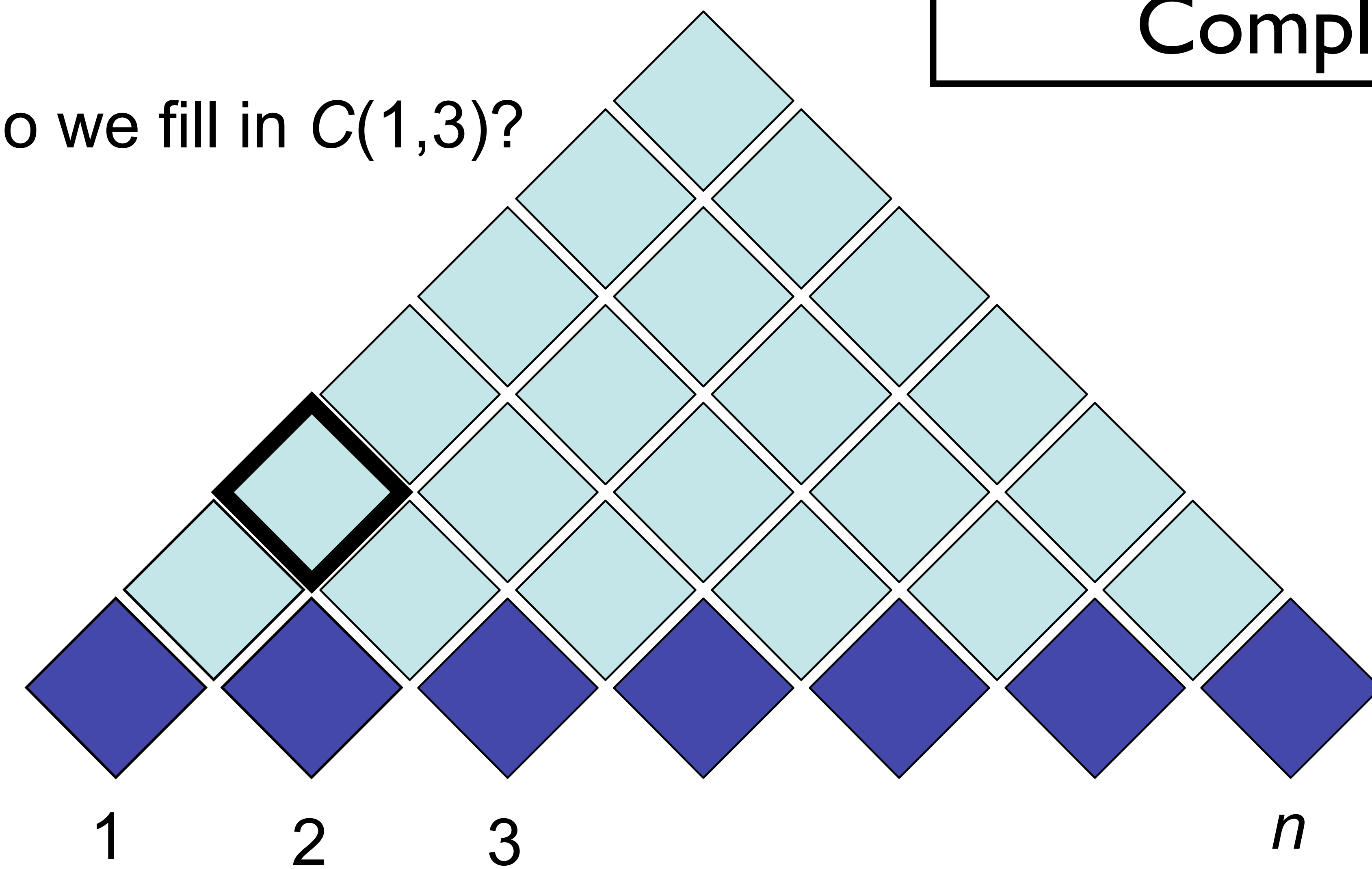


[Example from Noah Smith]

For cell [i,j]
    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]

Computational Complexity ?

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



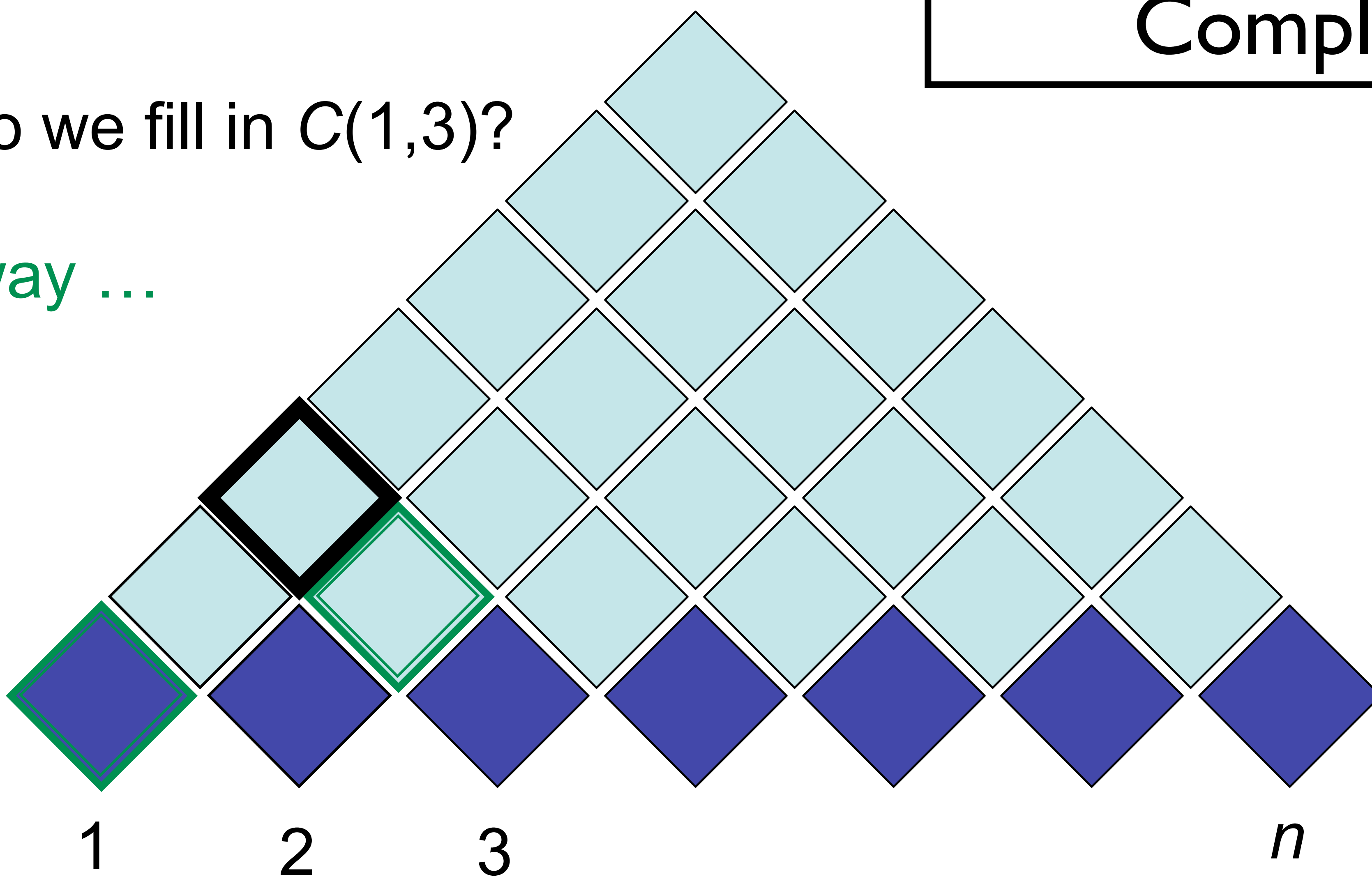
[Example from Noah Smith]

```
For cell [i,j]
    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]
```

Computational Complexity ?

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

One way …



[Example from Noah Smith]

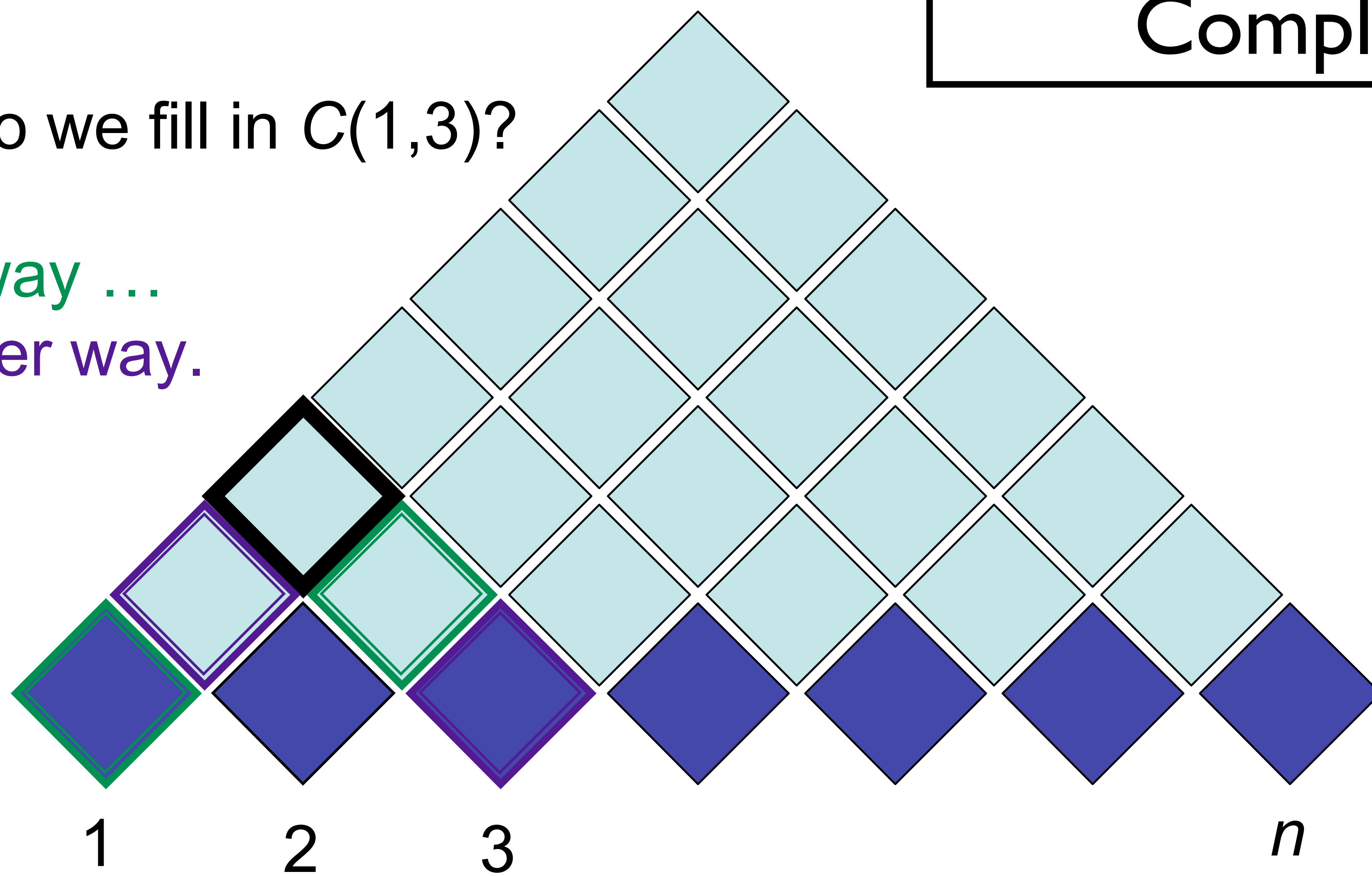
For cell [i,j]
    For possible splitpoint k=(i+1)..(j-1):
        For every B in [i,k] and C in [k,j],
            If exists rule A -> B C,
                add A to cell [i,j]

Computational Complexity ?

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

One way …

Another way.

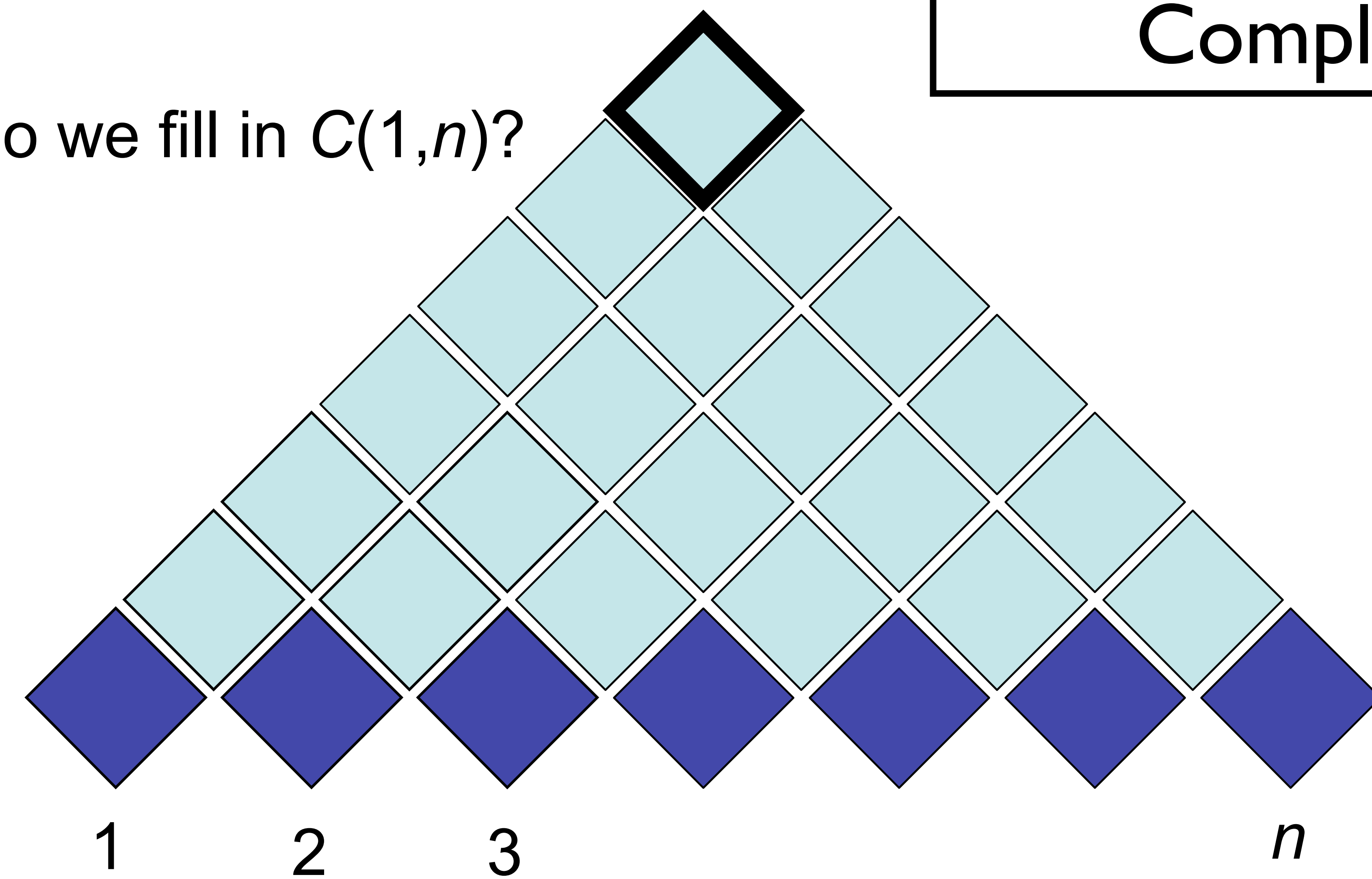


[Example from Noah Smith]

For cell [i,j]
    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]

Computational Complexity ?

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



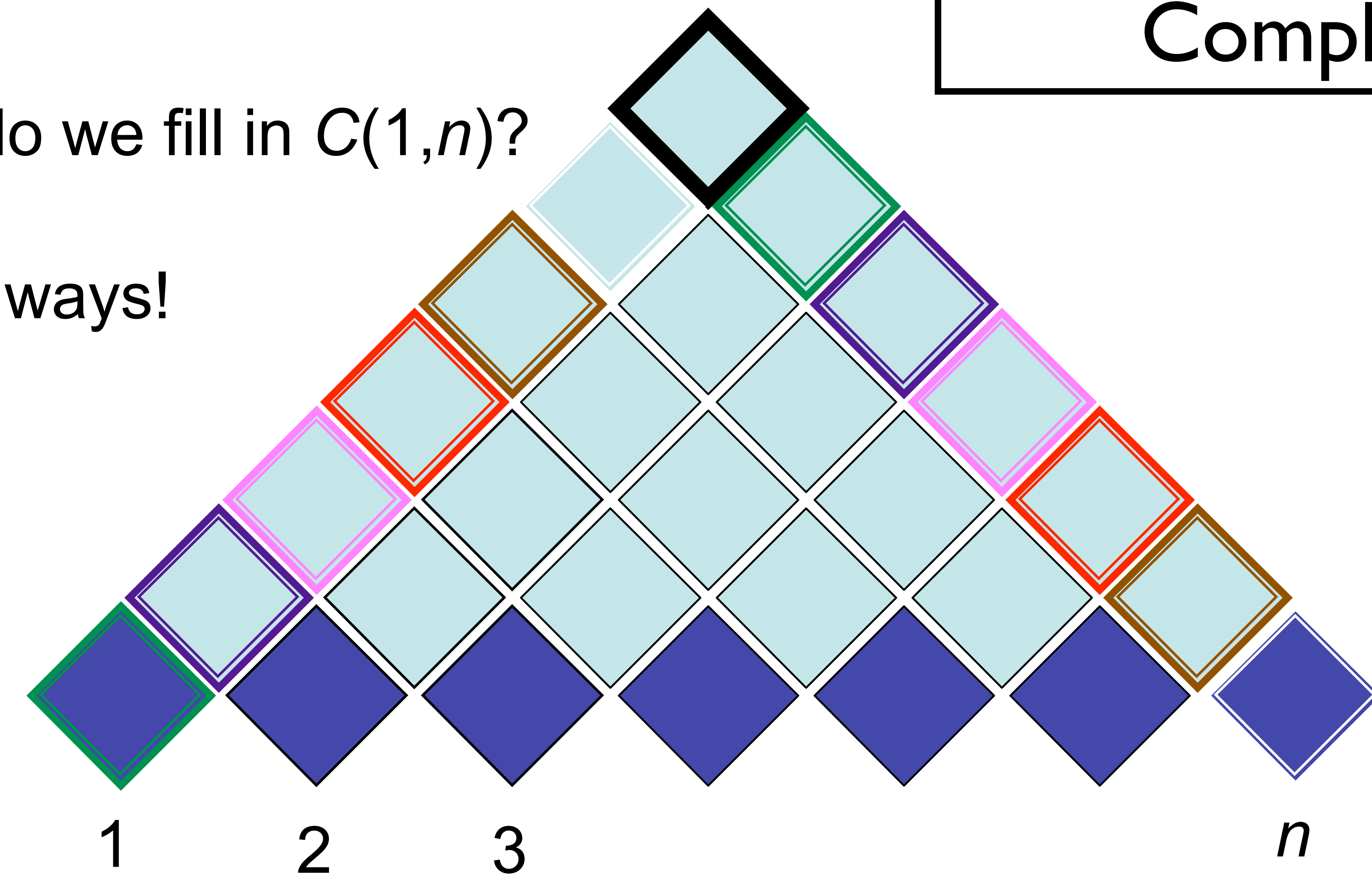
[Example from Noah Smith]

For cell [i,j]
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]

Computational Complexity ?

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

n - 1 ways!



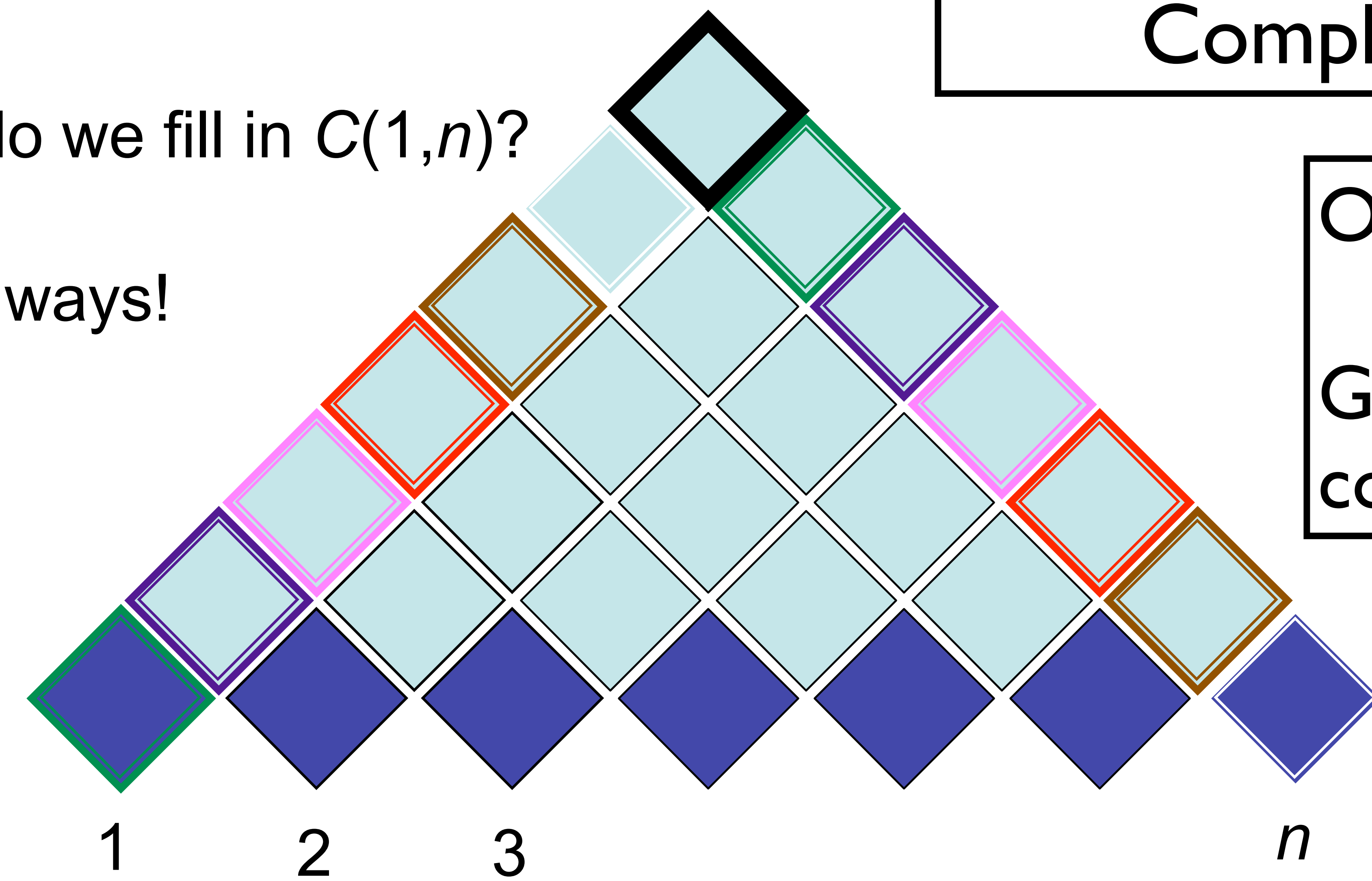
[Example from Noah Smith]

For cell [i,j]
    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]

Computational Complexity ?

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

n - 1 ways!



O(G n^3)

G = grammar constant

[Example from Noah Smith]

- Problem with a boolean grammar: Ambiguities!

Attachment ambiguity *we eat sushi with chopsticks, I shot an elephant in my pajamas.*

Modifier scope *southern food store*

Particle versus preposition *The puppy tore up the staircase.*

Complement structure *The tourists objected to the guide that they couldn't hear.*

Coordination scope *"I see," said the blind man, as he picked up the hammer and saw.*

Multiple gap constructions *The chicken is ready to eat*

[Examples from Eisenstein (2017)]

Probabilistic CFGs

$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.10] | a [.30] | the [.60]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$ [.10] | $flight$ [.30]
$S \rightarrow VP$	[.05]	| $meal$ [.15] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flights$ [.40] | $dinner$ [.10]
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow book$ [.30] | $include$ [.30]
$NP \rightarrow Det\ Nominal$	[.20]	| $prefer$; [.40]
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I$ [.40] | she [.05]
$Nominal \rightarrow Noun$	[.75]	| me [.15] | you [.40]
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow Houston$ [.60]
$Nominal \rightarrow Nominal\ PP$	[.05]	| TWA [.40]
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does$ [.60] | can [40]
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow from$ [.30] | to [.30]
$VP \rightarrow Verb\ NP\ PP$	[.10]	| on [.20] | $near$ [.15]
$VP \rightarrow Verb\ PP$	[.15]	| $through$ [.05]
$VP \rightarrow Verb\ NP\ NP$	[.05]	
$VP \rightarrow VP\ PP$	[.15]	
$PP \rightarrow Preposition\ NP$	[1.0]	

- Defines a probabilistic generative process for words in a sentence
- Extension of HMMs, strictly speaking
- (How to learn? Fully supervised with a treebank... EM for unsup...)

[J&M textbook]

Penn Treebank

```
( (S
    (NP-SBJ (NNP General) (NNP Electric) (NNP Co.) )
    (VP (VBD said)
      (SBAR (-NONE- 0)
        (S
          (NP-SBJ (PRP it) )
          (VP (VBD signed)
            (NP
              (NP (DT a) (NN contract) )
              (PP (-NONE- *ICH*-3) ))
            (PP (IN with)
              (NP
                (NP (DT the) (NNS developers) )
                (PP (IN of)
                  (NP (DT the) (NNP Ocean) (NNP State) (NNP Power) (NN project) ))))
            (PP-3 (IN for)
              (NP
                (NP (DT the) (JJ second) (NN phase) )
                (PP (IN of)
                  (NP
                    (NP (DT an) (JJ independent)
                      (ADJP
                        (QP ($ $) (CD 400) (CD million) )
                        (-NONE- *U*) )
                      (NN power) (NN plant) )
                    (, ,)
                    (SBAR
                      (WHNP-2 (WDT which) )
                      (S
                        (NP-SBJ-1 (-NONE- *T*-2) )
                        (VP (VBZ is)
                          (VP (VBG being)
                            (VP (VBN built)
                              (NP (-NONE- *-1) )
                              (PP-LOC (IN in)
                                (NP
                                  (NP (NNP Burrillville) )
                                  (, ,)
                                  (NP (NNP R.I) )))))))))))))))))
```

(P)CFG model, (P)CKY algorithm

- CKY: given CFG and sentence w
  - Does there exist at least one parse?
  - Enumerate parses (backpointers)

- Probabilistic/Weighted CKY: given PCFG and sentence w
  - Likelihood of sentence P(w)
  - Most probable parse ("Viterbi parse")
    $\text{argmax}_y\ P(y \mid w) = \text{argmax}_y\ P(y, w)$
  - Non-terminal span marginals

- Discriminative Tree-CRF parsing:
  $\text{argmax}_y\ P(y \mid w)$

- Parsing model accuracy: lots of ambiguity!!
  - PCFGs lack lexical information to resolve ambiguities (sneak in world knowledge?)
  - Modern constituent parsers: enrich PCFG with lexical information and fine-grained nonterminals
  - Modern dependency parsers: effectively the same trick

- Parsers' computational efficiency
  - Grammar constant; pruning & heuristic search
  - $O(N^3)$ for CKY (ok? sometimes...)
  - O(N) left-to-right incremental algorithms

- Evaluate: precision and recall of labeled spans
- Treebank data

Better PCFG grammars

- Nonterminal splitting: because substitutability is too strong (e.g. “she” as subject vs object)

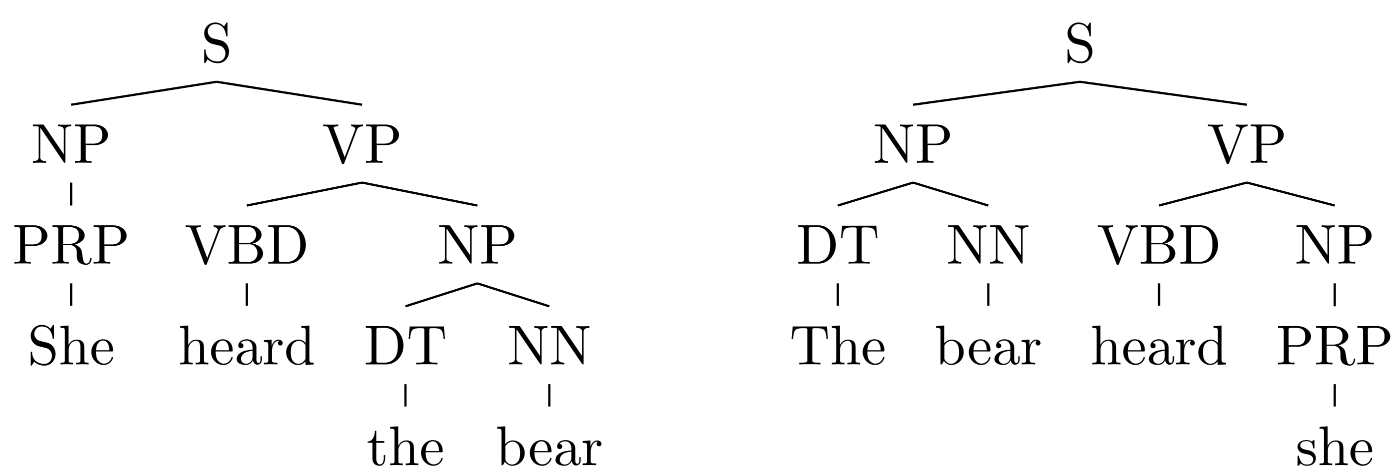


Figure 11.5: A grammar that allows *she* to take the object position wastes probability mass on ungrammatical sentences.

[From Eisenstein (2017)]

Better PCFG grammars

- Parent annotation

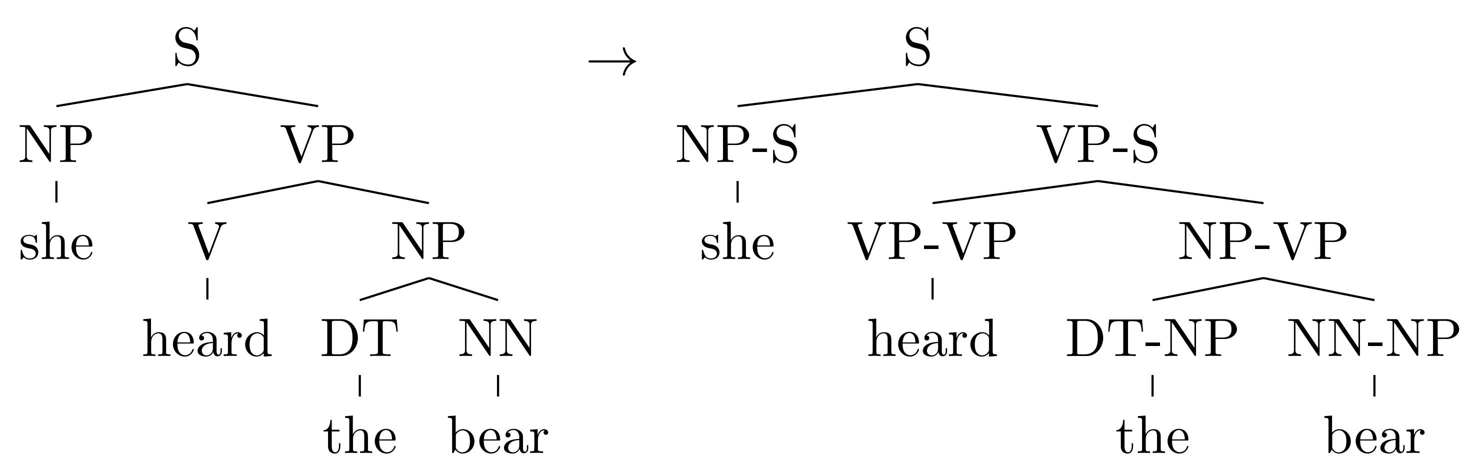


Figure 11.8: Parent annotation in a CFG derivation

[From Eisenstein (2017)]

Better PCFG grammars

- Linguistically designed state splits
  - (Or: automatically learned ones with split-merge EM)

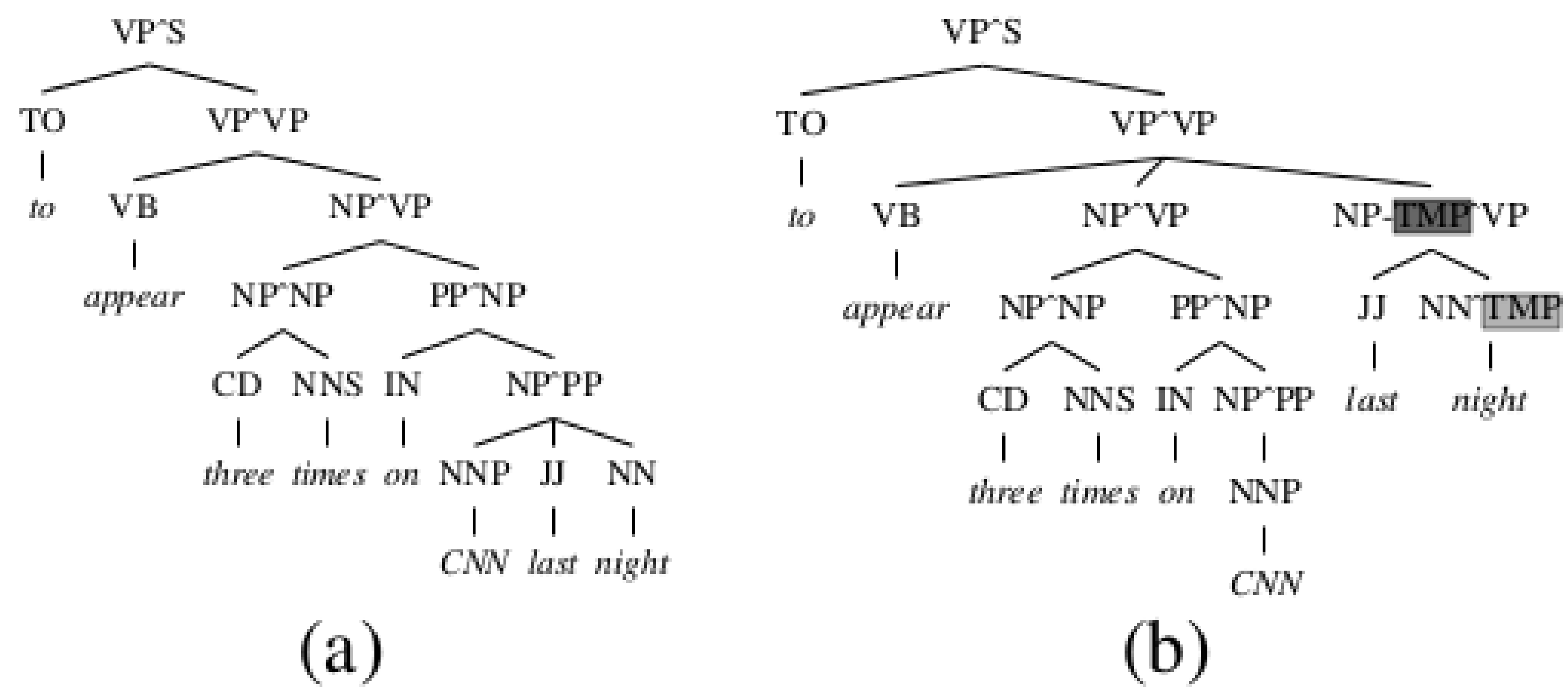


Figure 11.13: State-splitting creates a new non-terminal called NP-TMP, for temporal noun phrases. This corrects the PCFG parsing error in (a), resulting in the correct parse in (b).

[From Eisenstein (2017)]

Better PCFG grammars

- Lexicalization: encode semantic preferences

Non-terminal	Direction	Priority
S	right	VP SBAR ADJP UCP NP
VP	left	VBD VBN MD VBZ TO VB VP VBG VBP ADJP NP
NP	right	N* EX $ CD QP PRP ...
PP	left	IN TO FW

Table 11.3: A fragment of head percolation rules

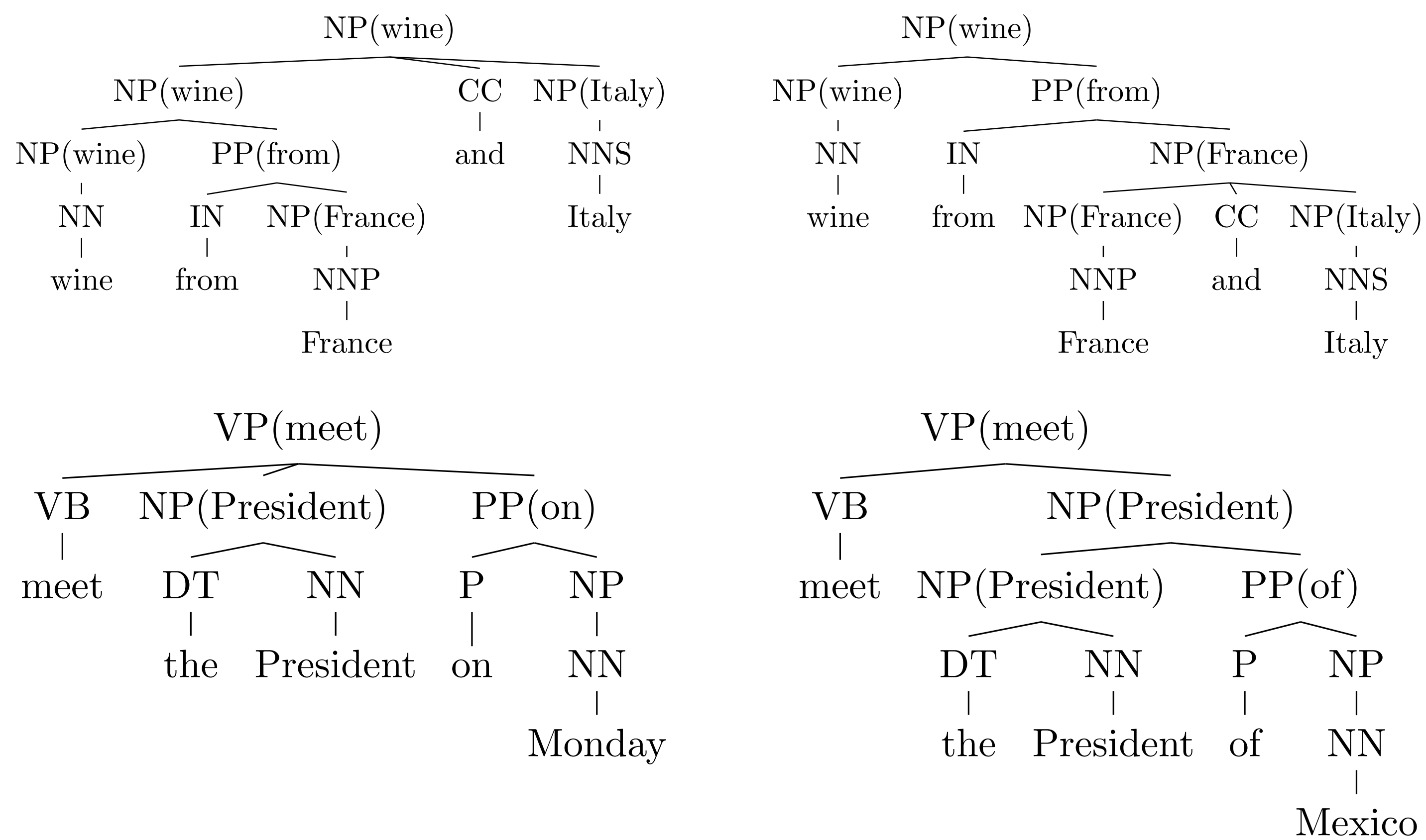


Figure 11.9: Lexicalization can address ambiguity on coordination scope (upper) and PP attachment (lower)

[From Eisenstein (2017)]

Better PCFG grammars/more

Vanilla PCFG	72%
Parent-annotations (Johnson, 1998)	80%
Lexicalized (Charniak, 1997)	86%
Lexicalized (Collins, 2003)	87%
Lexicalized, reranking, self-training (McClosky et al., 2006)	92.1%
State splitting (Petrov and Klein, 2007)	90.1%
CRF Parsing (Finkel et al., 2008)	89%
TAG Perceptron Parsing (Carreras et al., 2008)	91.1%
Compositional Vector Grammars (Socher et al., 2013a)	90.4%
Neural CRF (Durrett and Klein, 2015)	91.1%

Table 11.7: Penn Treebank parsing scoreboard, circa 2015 (Durrett and Klein, 2015)

[From Eisenstein (2017)]

- stopped here 3/7

Treebanks

- Penn Treebank (constituents, English)
  - http://www.cis.upenn.edu/~treebank/home.html
  - Recent revisions in Ononotes
- Universal Dependencies
  - http://universaldependencies.org/
- Prague Treebank (syn+sem)
- many others...
- Know what you're getting!

Left-to-right parsing

- Shift-reduce parsing -- linear time (in sentence length)!
- Most practically efficient for constituent parsing -- e.g. zpar and corenlp implementations

Stack$_t$	**Buffer**$_t$	**Open NTs**$_t$	**Action**	**Stack**$_{t+1}$	**Buffer**$_{t+1}$	**Open NTs**$_{t+1}$
S	B	n	NT(X)	$S \mid (\mathrm{X}$	B	$n+1$
S	$x \mid B$	n	SHIFT	$S \mid x$	B	n
$S \mid (\mathrm{X} \mid \tau_1 \mid \ldots \mid \tau_\ell$	B	n	REDUCE	$S \mid (\mathrm{X}\ \tau_1\ \ldots\ \tau_\ell)$	B	$n-1$

Input: *The hungry cat meows .*

	Stack	**Buffer**	**Action**
0		*The* | *hungry* | *cat* | *meows* | .	NT(S)
1	(S	*The* | *hungry* | *cat* | *meows* | .	NT(NP)
2	(S | (NP	*The* | *hungry* | *cat* | *meows* | .	SHIFT
3	(S | (NP | *The*	*hungry* | *cat* | *meows* | .	SHIFT
4	(S | (NP | *The* | *hungry*	*cat* | *meows* | .	SHIFT
5	(S | (NP | *The* | *hungry* | *cat*	*meows* | .	REDUCE
6	(S | (NP *The hungry cat*)	*meows* | .	NT(VP)
7	(S | (NP *The hungry cat*) | (VP	*meows* | .	SHIFT
8	(S | (NP *The hungry cat*) | (VP *meows*	.	REDUCE
9	(S | (NP *The hungry cat*) | (VP *meows*)	.	SHIFT
10	(S | (NP *The hungry cat*) | (VP *meows*) | .		REDUCE
11	(S (NP *The hungry cat*) (VP *meows*) .)		

https://arxiv.org/pdf/1602.07776.pdf

Question answering in the news

https://theoutline.com/post/1192/google-s-featured-snippets-are-worse-than-fake-news

https://twitter.com/ruskin147/status/838445095410106368/video/1