

# Language and Statistics II

---

Lecture 13: Parsing algorithms

# Decoding Algorithms

---

- Suppose I have a PCFG and a sentence.
- What might I want to do?
  - Find the most likely tree (if it exists).
  - Find the  $k$  most likely trees.
  - Gather statistics on the **distribution** over trees.
- Should remind you of HMMs!

# Probabilistic CKY

---

Input: PCFG  $G = (\Sigma, \mathbf{N}, S, \mathbf{R})$  in CNF and sequence  $\mathbf{w} \in \Sigma^*$

Output: most likely tree for  $\mathbf{w}$ , if it exists, and its probability.

# Resist This Temptation!

---

- CKY is not “building a tree” bottom-up.
- It is scoring partial hypotheses bottom-up.
- You can assume nothing about the tree until you get to the end!

# Probabilistic CKY

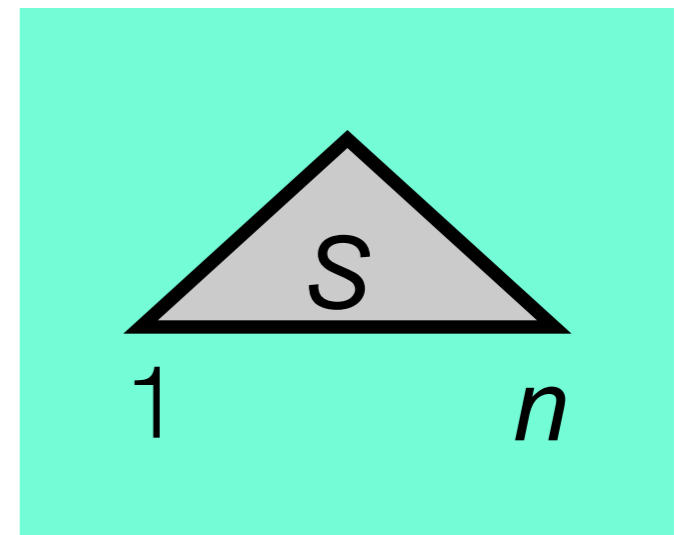
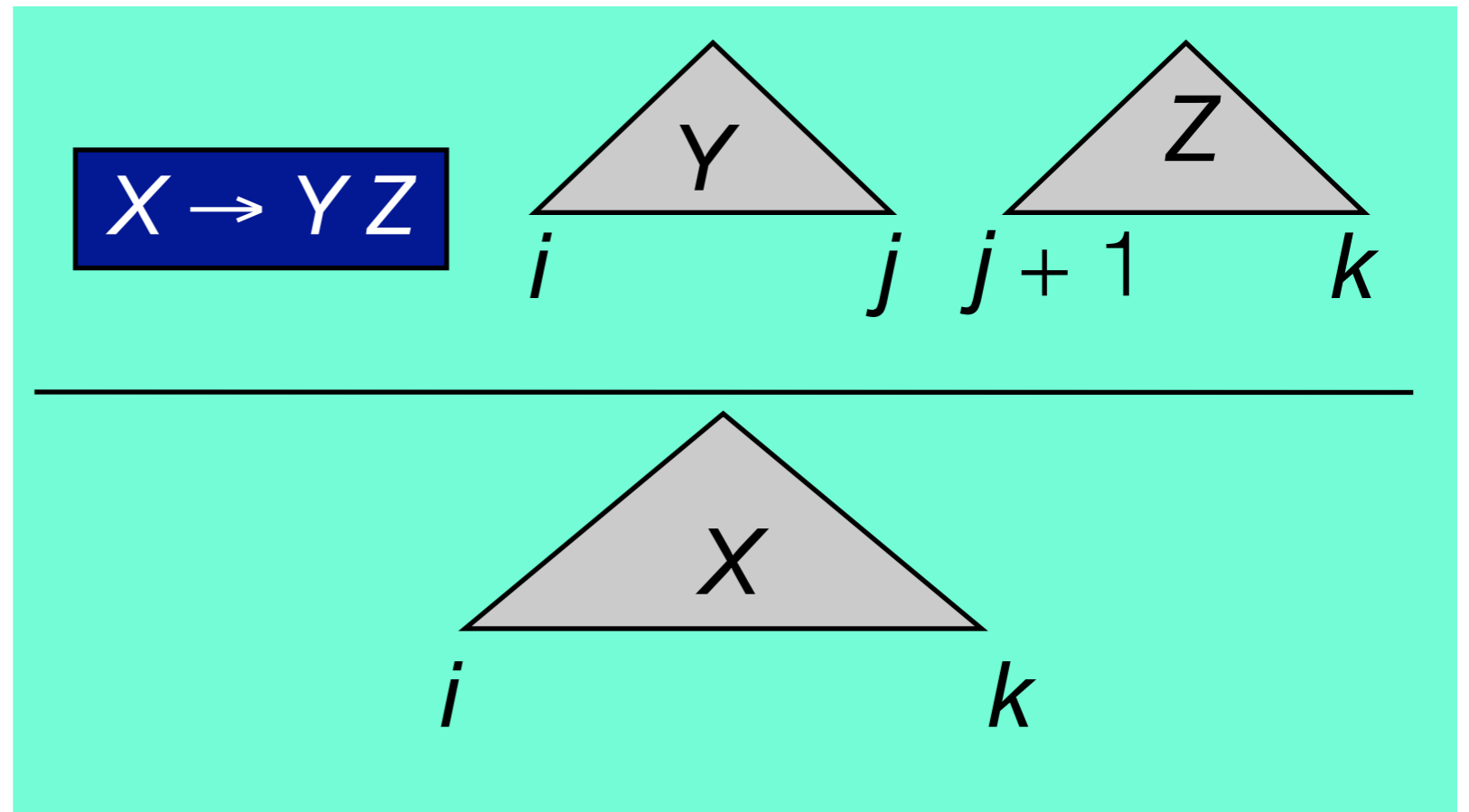
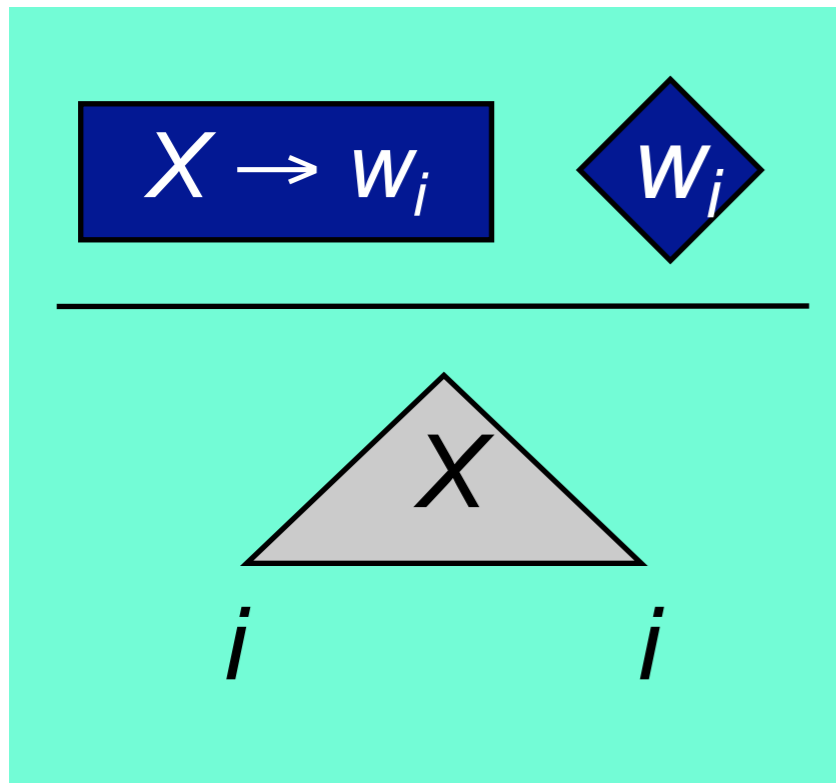
---

$\text{constit}(X, l, l) \text{ max} = \underline{\text{word}}(W, l) \times \underline{\text{unary}}(X, W).$

$\text{constit}(X, l, K) \text{ max} = \text{constit}(Y, l, J) \times \text{constit}(Z, J+1, K) \times$   
 $\underline{\text{binary}}(X, Y, Z).$

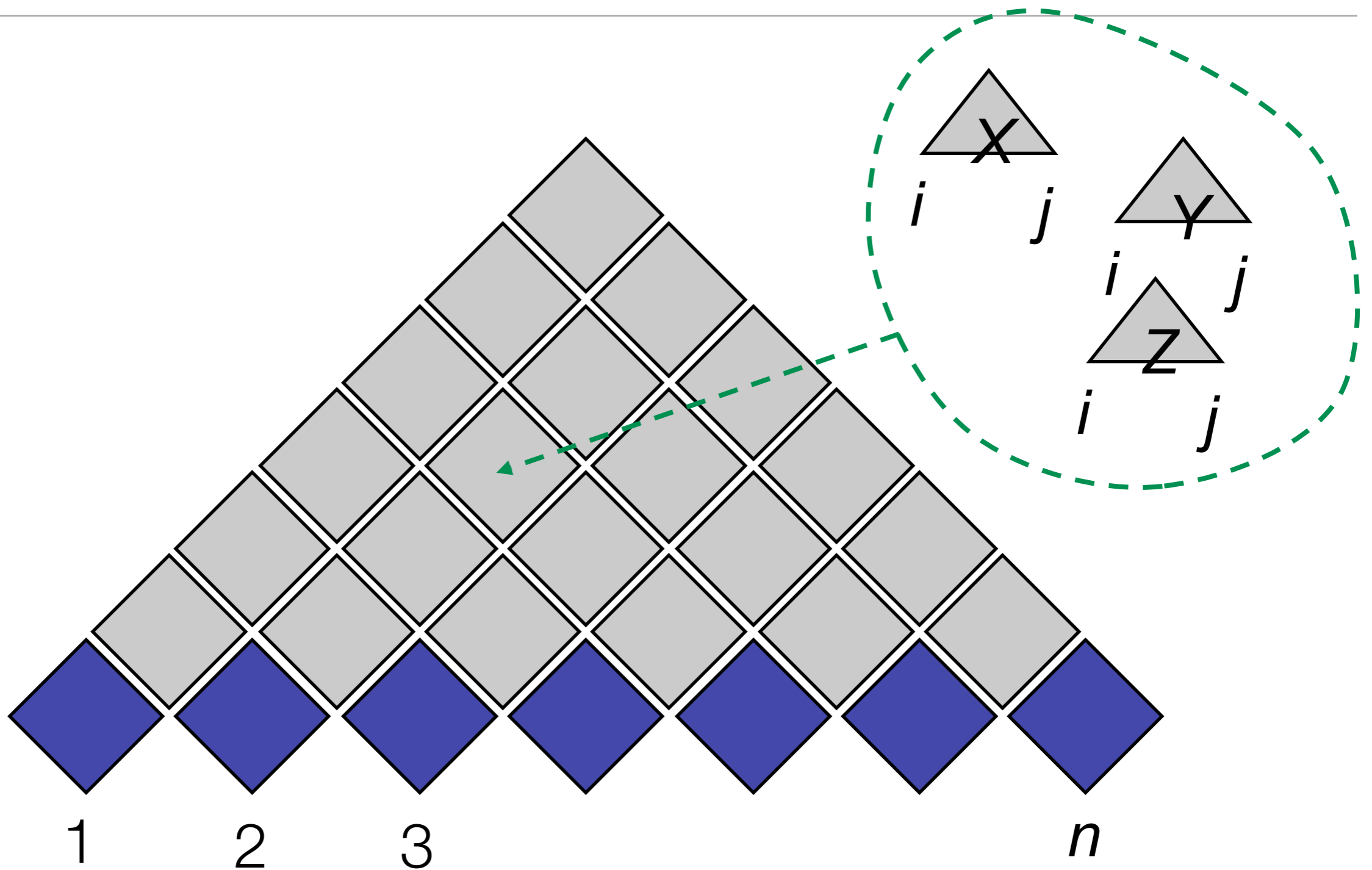
$\text{goal} \text{ max} = \text{constit}(S, 1, N) \times \underline{\text{length}}(N) \times \underline{\text{startsymbol}}(S).$

# Visualizing Probabilistic CKY



# Visualizing Probabilistic CKY

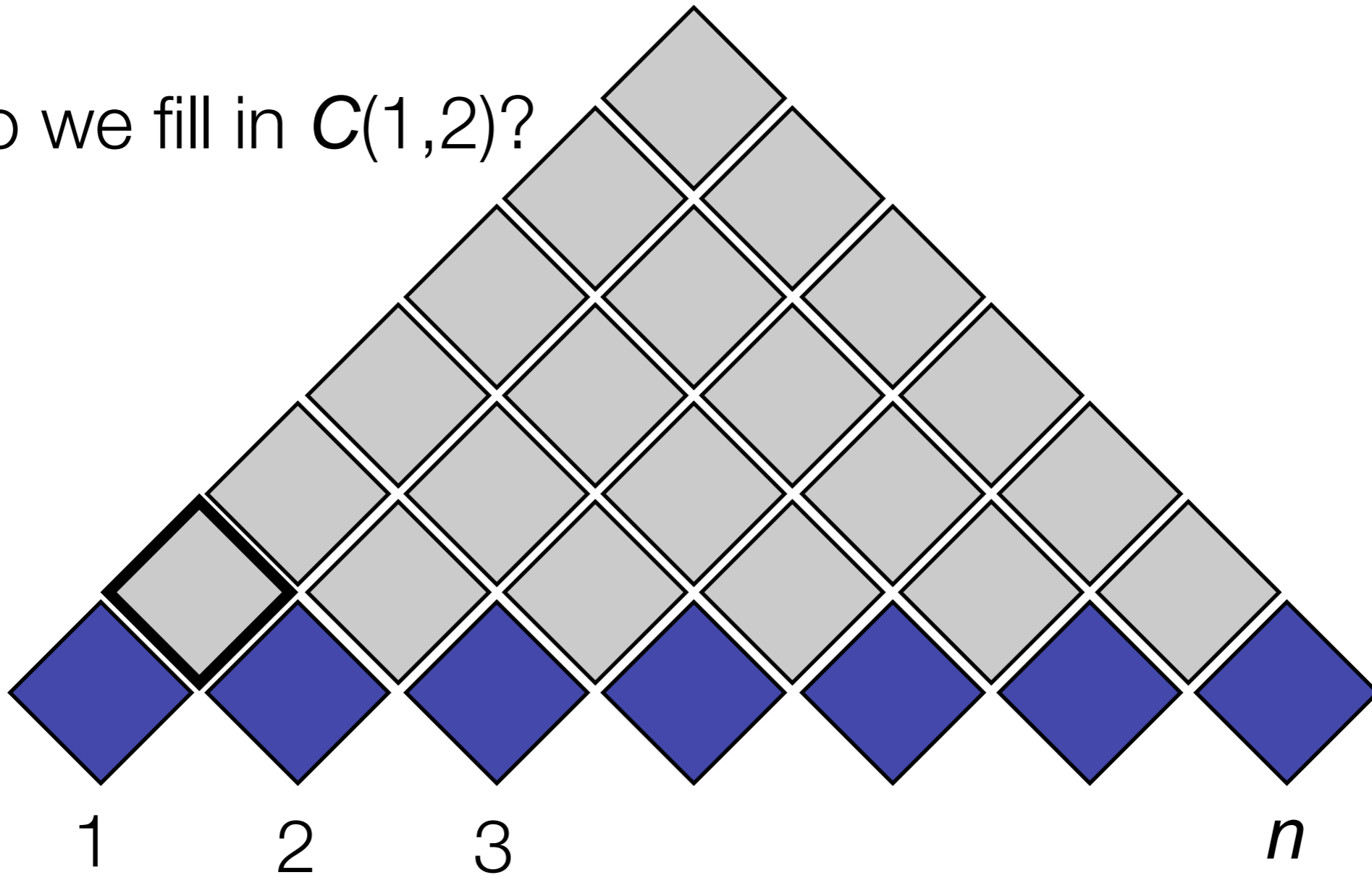
---



# Visualizing Probabilistic CKY

---

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

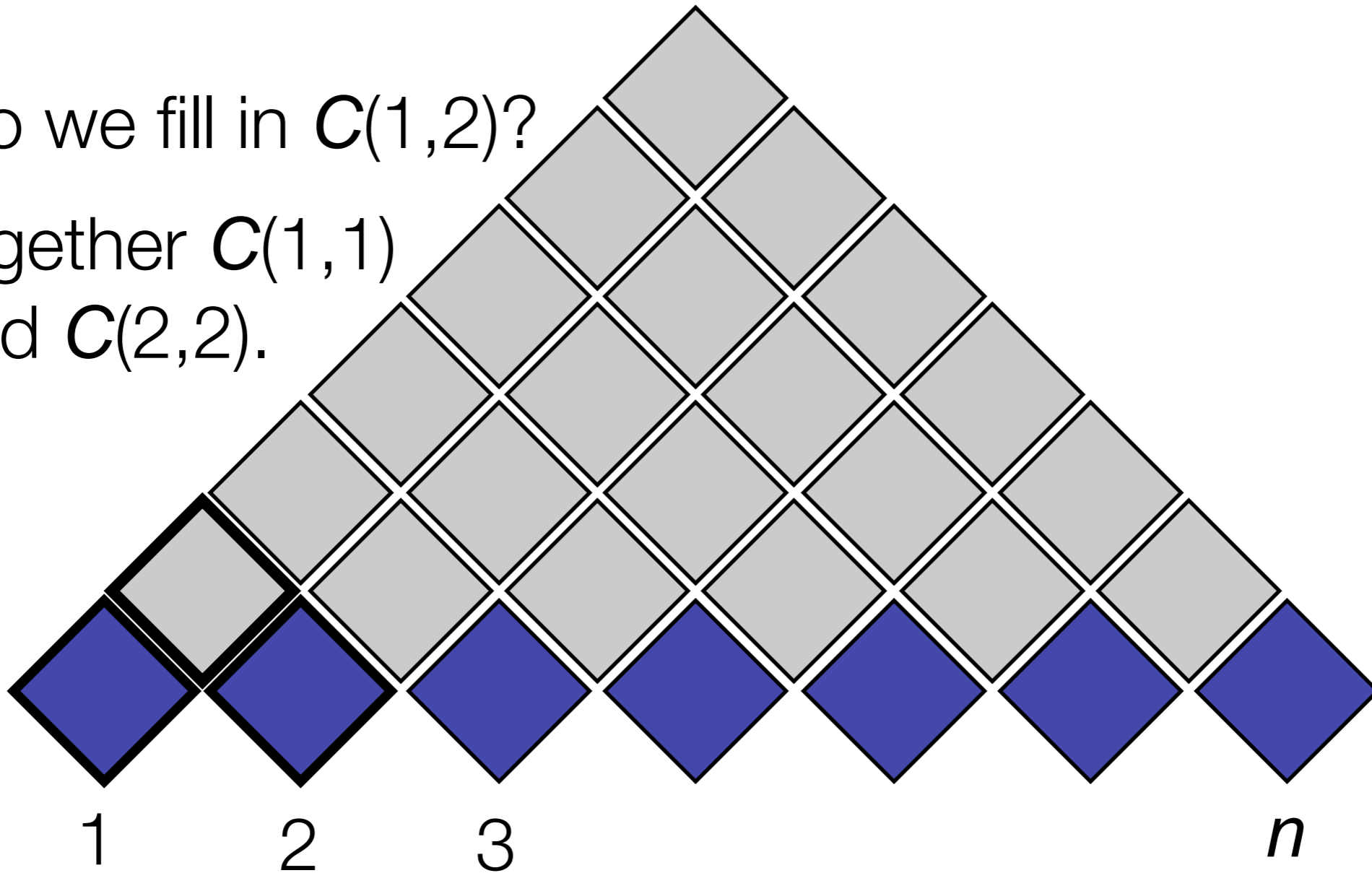


# Visualizing Probabilistic CKY

---

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

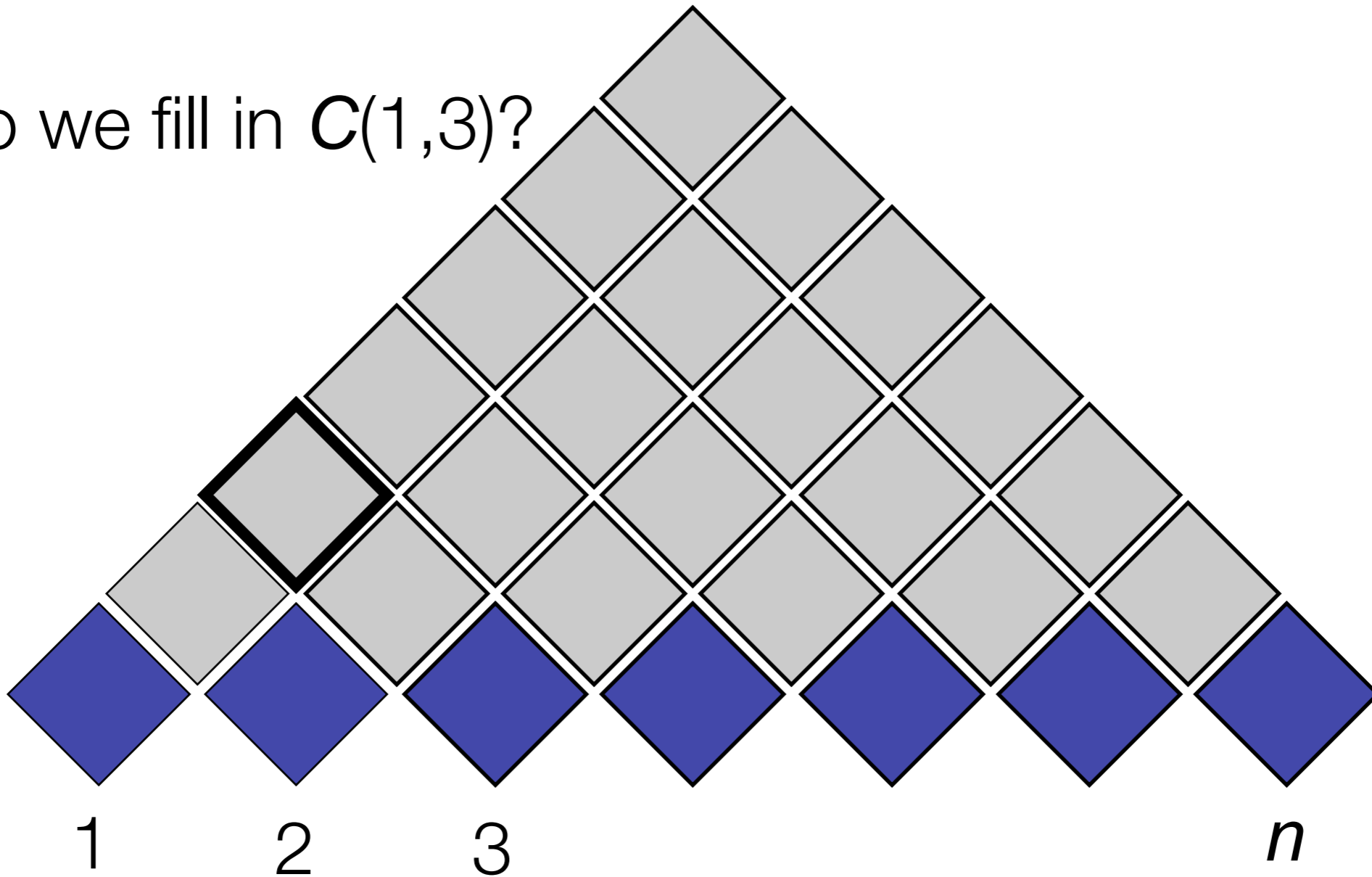
Put together  $C(1,1)$   
and  $C(2,2)$ .



# Visualizing Probabilistic CKY

---

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

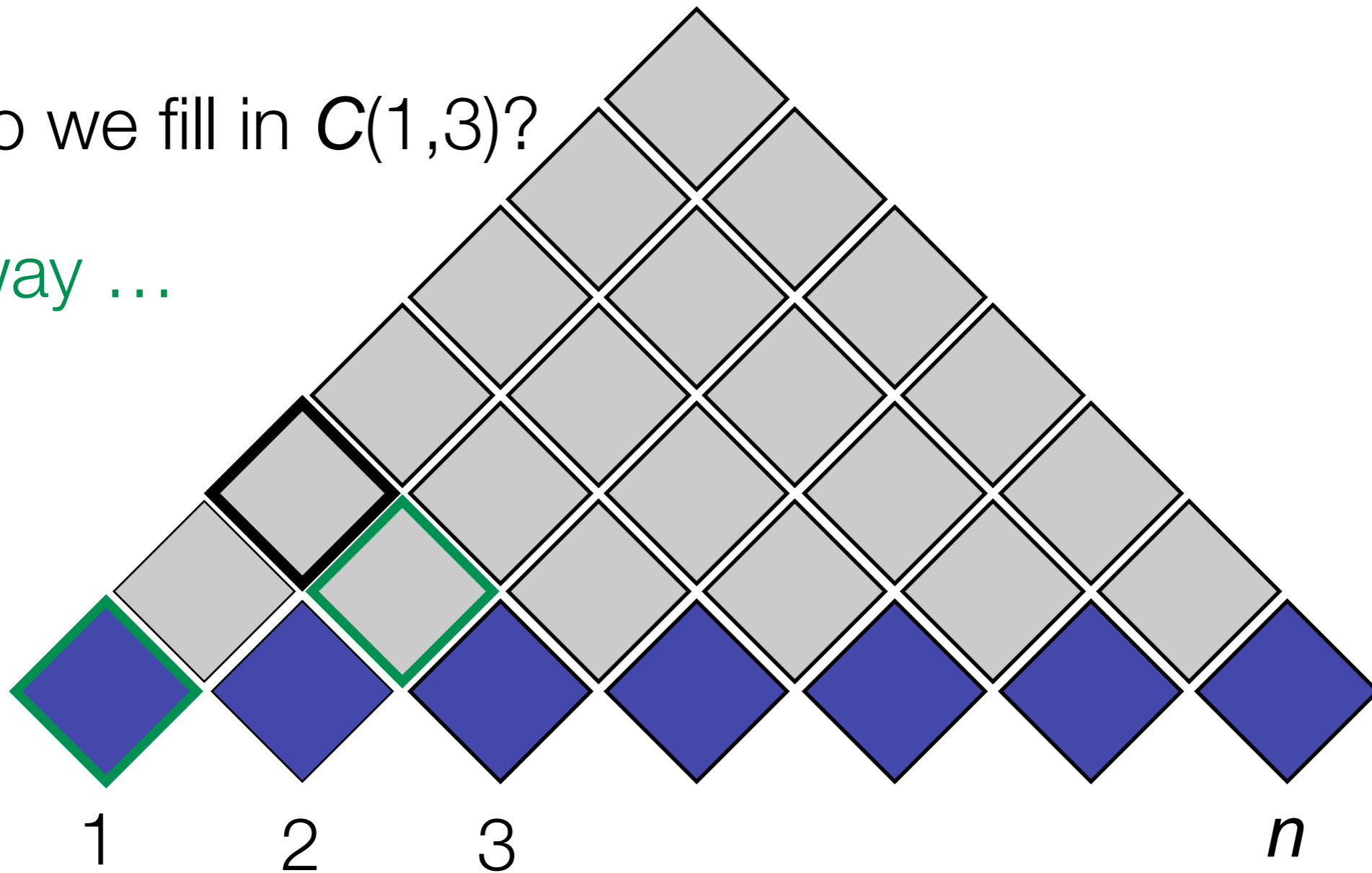


# Visualizing Probabilistic CKY

---

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

One way ...



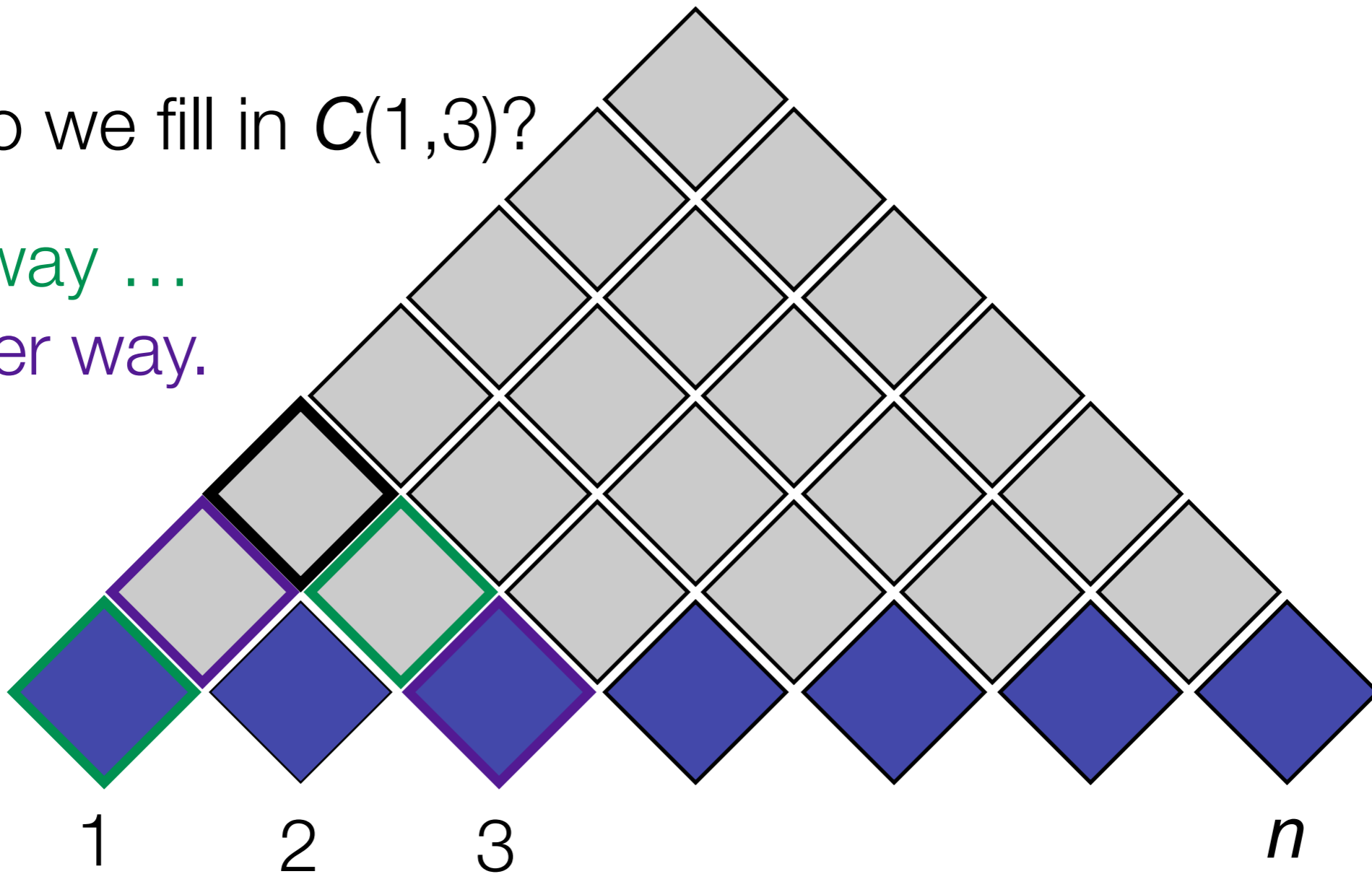
# Visualizing Probabilistic CKY

---

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

One way ...

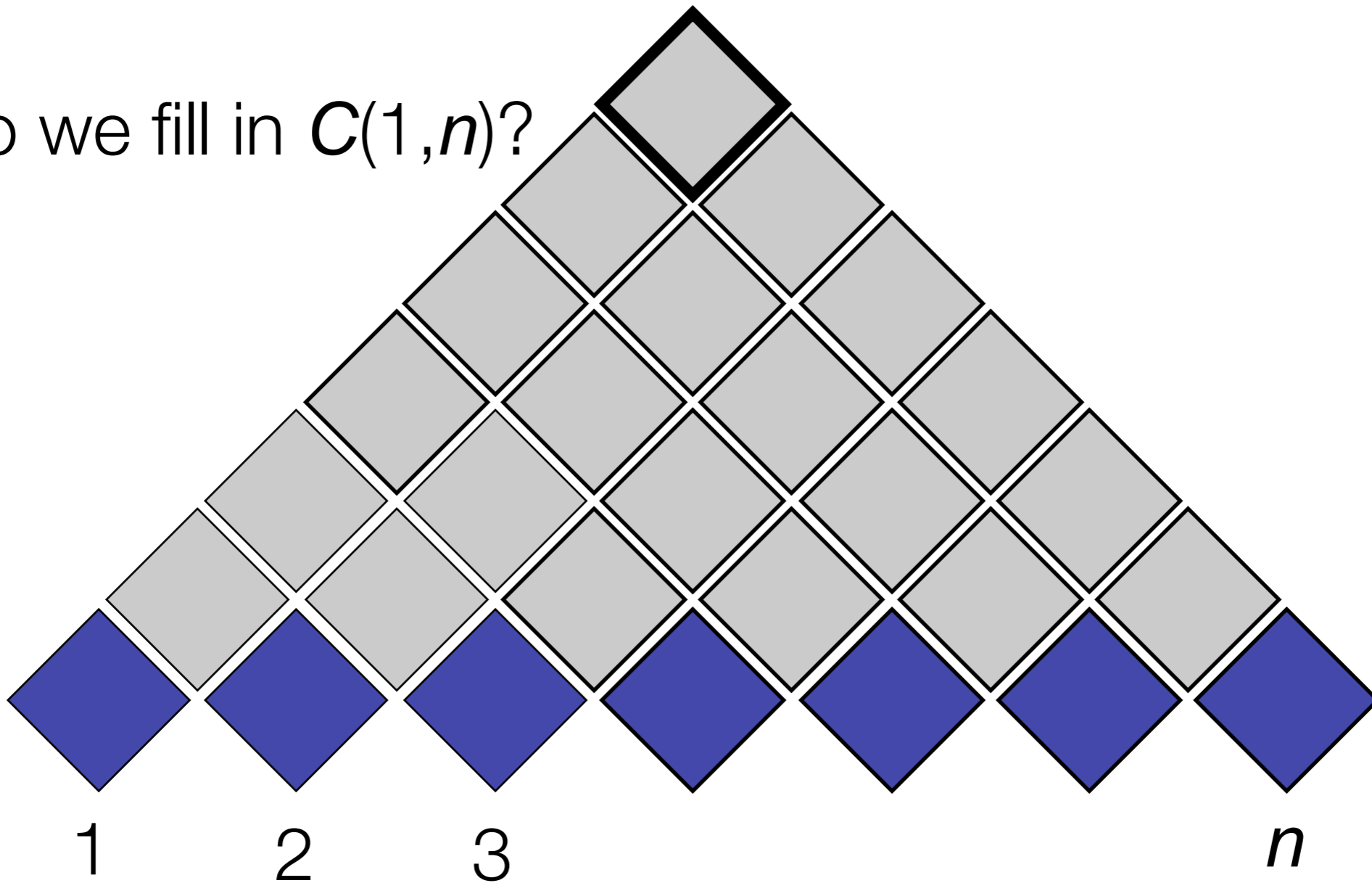
Another way.



# Visualizing Probabilistic CKY

---

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

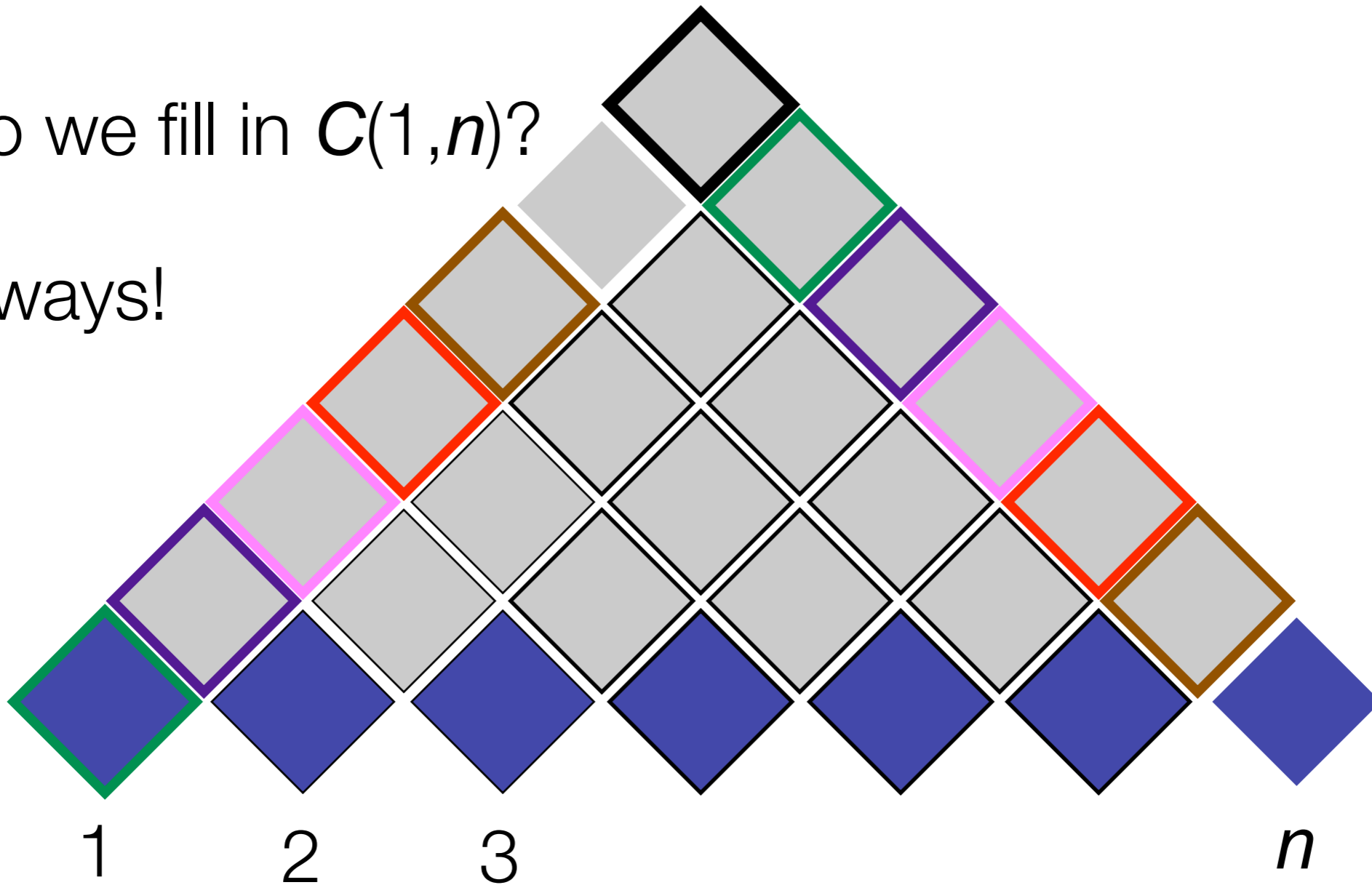


# Visualizing Probabilistic CKY

---

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

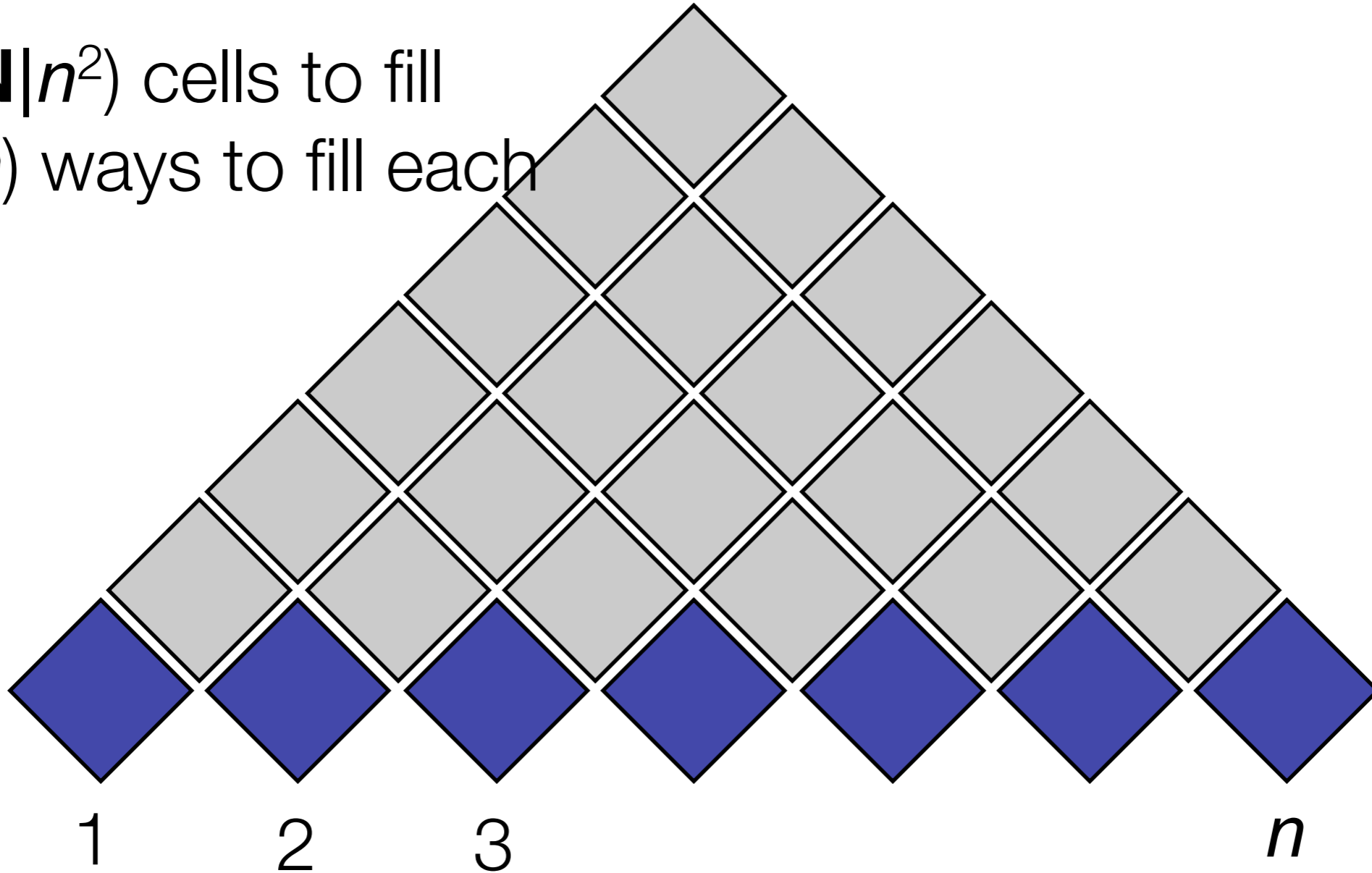
$n - 1$  ways!



# Visualizing Probabilistic CKY

---

$O(|\mathbf{N}|n^2)$  cells to fill  
 $O(|\mathbf{N}|^2n)$  ways to fill each



# Probabilistic Earley's

---

Input: PCFG  $G = (\Sigma, \mathbf{N}, S, \mathbf{R})$  and sequence  $\mathbf{w} \in \Sigma^*$

Output: most likely tree for  $\mathbf{w}$ , if it exists, and its probability.

# Probabilistic Earley's

---

$\text{need}(X, l) \max = \text{constit}(\_ / X \alpha, \_, l)$ .

$\text{need}(S, 0) \max = \text{startsymbol}(S)$ .

$\text{constit}(X/\alpha, l, l) \max = \text{rewrite}(X, \alpha)$  whenever  $\text{need}(X, l)$ .

$\text{constit}(X/\alpha, l, J+1) \max = \text{constit}(X/W \alpha, l, J) \times \text{word}(W, J + 1)$ .

$\text{constit}(X/\alpha, l, K) \max = \text{constit}(X/Y \alpha, l, J) \times \text{constit}(Y/\epsilon, J, K)$ .

$\text{goal} \max = \text{constit}(S/\epsilon, 0, N) \times \text{length}(N) \times \text{startsymbol}(S)$ .

# Probabilistic Earley's

---

$\text{need}(X, l) \max = \text{constit}(\_ / X \alpha, \_, l)$ .

$\text{need}(S, 0) \max = \text{startsymbol}(S)$ .

$\text{constit}(X/\alpha, l, l) \max = \text{rewrite}(X, \alpha)$  whenever  $\text{need}(X, l)$ . predict

$\text{constit}(X/\alpha, l, J+1) \max = \text{constit}(X/W \alpha, l, J) \times \text{word}(W, J + 1)$ .

$\text{constit}(X/\alpha, l, K) \max = \text{constit}(X/Y \alpha, l, J) \times \text{constit}(Y/\epsilon, J, K)$ .

$\text{goal} \max = \text{constit}(S/\epsilon, 0, N) \times \text{length}(N) \times \text{startsymbol}(S)$ .

# Probabilistic Earley's

---

$\text{need}(X, l) \max = \text{constit}(\_ / X \alpha, \_, l)$ .

$\text{need}(S, 0) \max = \text{startsymbol}(S)$ .

$\text{constit}(X/\alpha, l, l) \max = \text{rewrite}(X, \alpha)$  whenever  $\text{need}(X, l)$ .

$\text{constit}(X/\alpha, l, J+1) \max = \text{constit}(X/W \alpha, l, J) \times \text{word}(W, J + 1)$ . scan

$\text{constit}(X/\alpha, l, K) \max = \text{constit}(X/Y \alpha, l, J) \times \text{constit}(Y/\epsilon, J, K)$ .

$\text{goal} \max = \text{constit}(S/\epsilon, 0, N) \times \text{length}(N) \times \text{startsymbol}(S)$ .

# Probabilistic Earley's

---

$\text{need}(X, l) \max = \text{constit}(\_ / X \alpha, \_, l)$ .

$\text{need}(S, 0) \max = \text{startsymbol}(S)$ .

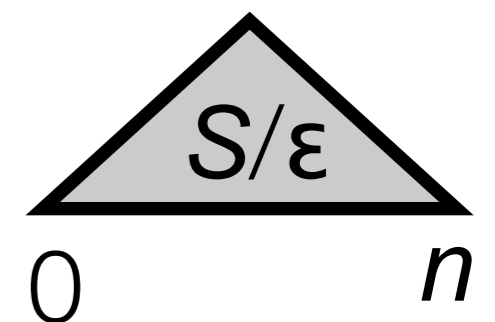
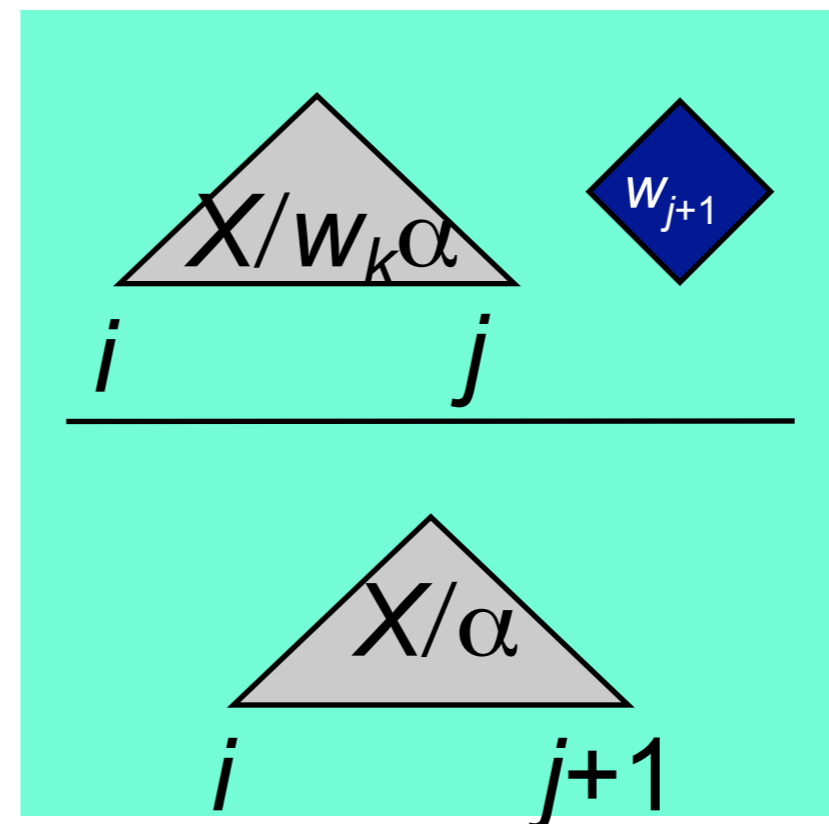
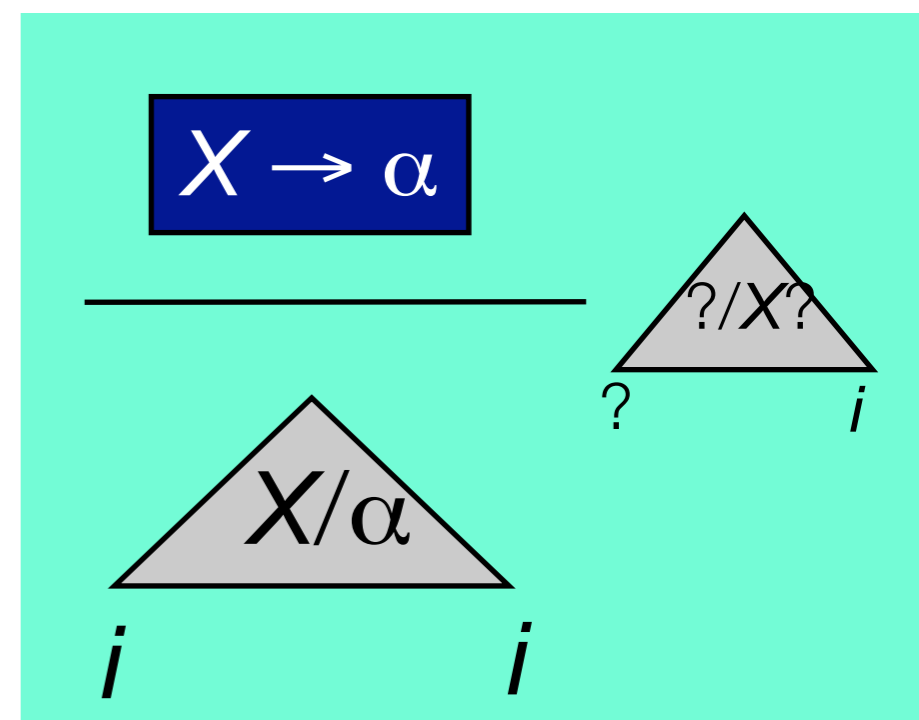
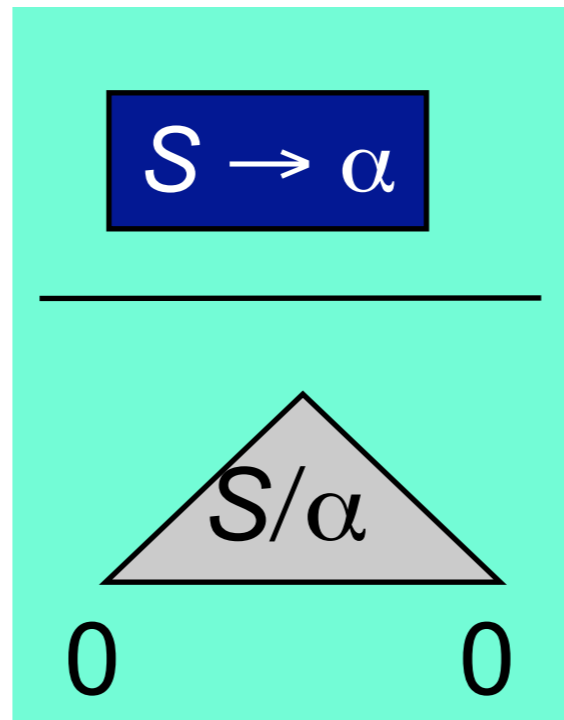
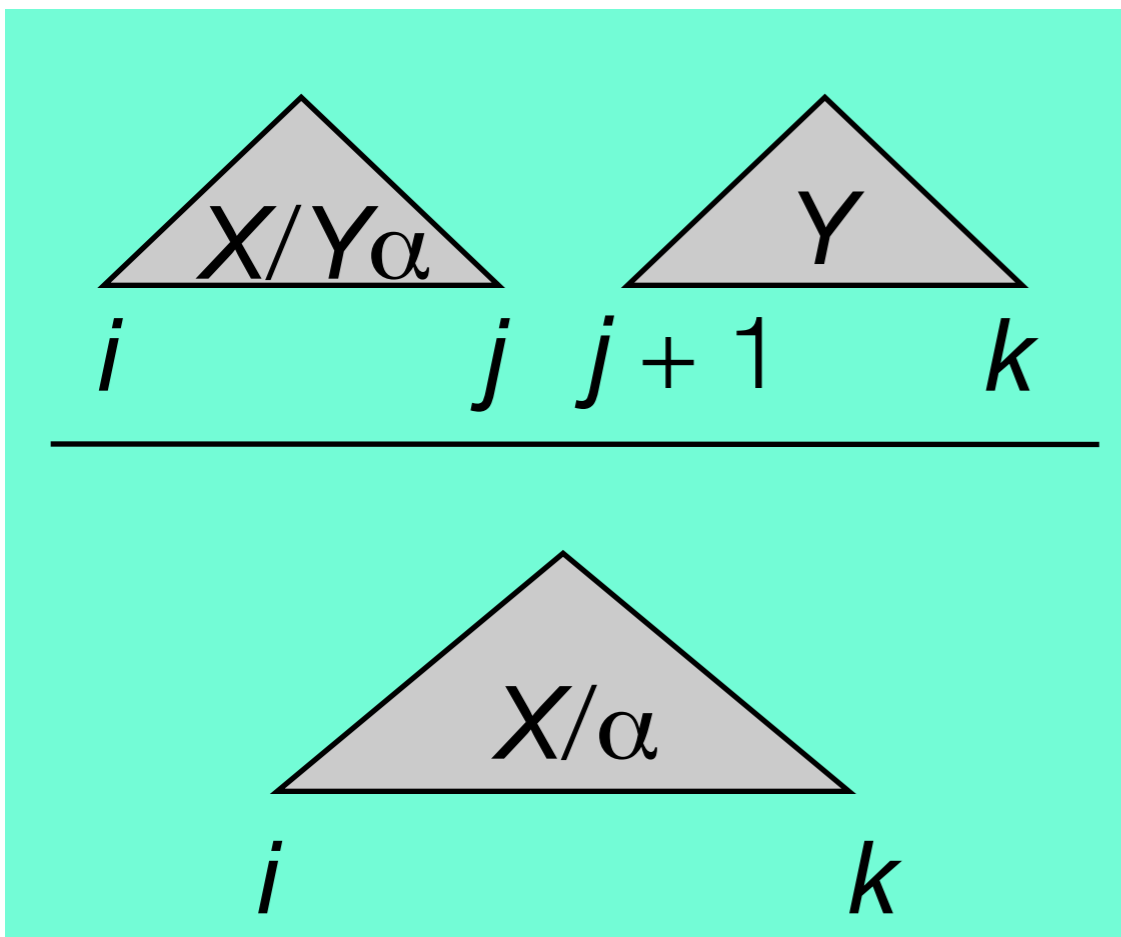
$\text{constit}(X/\alpha, l, l) \max = \text{rewrite}(X, \alpha)$  whenever  $\text{need}(X, l)$ .

$\text{constit}(X/\alpha, l, J+1) \max = \text{constit}(X/W \alpha, l, J) \times \text{word}(W, J + 1)$ .

$\text{constit}(X/\alpha, l, K) \max = \text{constit}(X/Y \alpha, l, J) \times \text{constit}(Y/\epsilon, J, K)$ . complete

$\text{goal} \max = \text{constit}(S/\epsilon, 0, N) \times \text{length}(N) \times \text{startsymbol}(S)$ .

# Visualizing Probabilistic Earley's



# CKY vs. Earley's

---

- Both  $O(n^3)$  runtime,  $O(n^2)$  space
- Earley's doesn't require the grammar to be in CNF
- Proof structures in Earley's "move" left-to-right; CKY "moves" bottom-to-top.
- Earley's  $\approx$  on-the-fly binarization + CKY
  
- Thought question: Does either remind you of Viterbi?

# CKY and Earley's vs. Other Methods

---

- Tomita parsing - shift and reduce operations, with a stack - inspired by **search** in AI.
  - Can make it probabilistic.
  - No polynomial guarantees (could be exponential if lots of stack splitting).
  - In practice usually fast.
- CKY and Earley's algorithms **can** be generalized to use an agenda, rather than filling in all cells.
  - “Best-first” tricks; sometimes optimality is not sacrificed!
- Remember the Forward algorithm?
  - We'll come back to “inside” algorithms in a couple of weeks.

# **Some Important Parsers**

# Collins Model 1 (1997)

---

- Trees are headed and lexicalized
  - What's the difference?

- Huge number of rules!

$VP_{\text{saw}} \rightarrow \underline{V}_{\text{saw}} NP_{\text{man}} PP_{\text{through}}$

$VP_{\text{saw}} \rightarrow \underline{V}_{\text{saw}} NP_{\text{man}} PP_{\text{with}}$

$VP_{\text{saw}} \rightarrow \underline{V}_{\text{saw}} NP_{\text{woman}} PP_{\text{through}}$

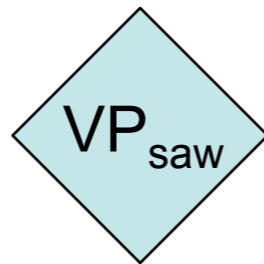
$VP_{\text{saw}} \rightarrow \underline{V}_{\text{saw}} NP_{\text{man}}$

- Key: factor probabilities within rule.

# Collins Model 1 (1997)

---

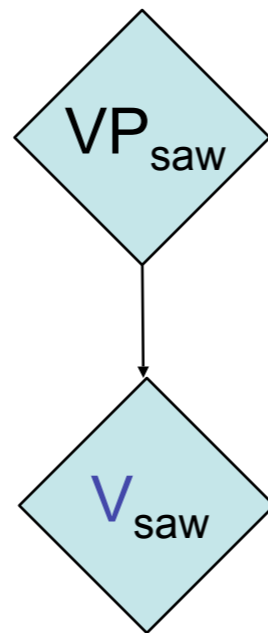
- Everything factors down to rules, then further. We're given the parent nonterminal and head word.



# Collins Model 1 (1997)

---

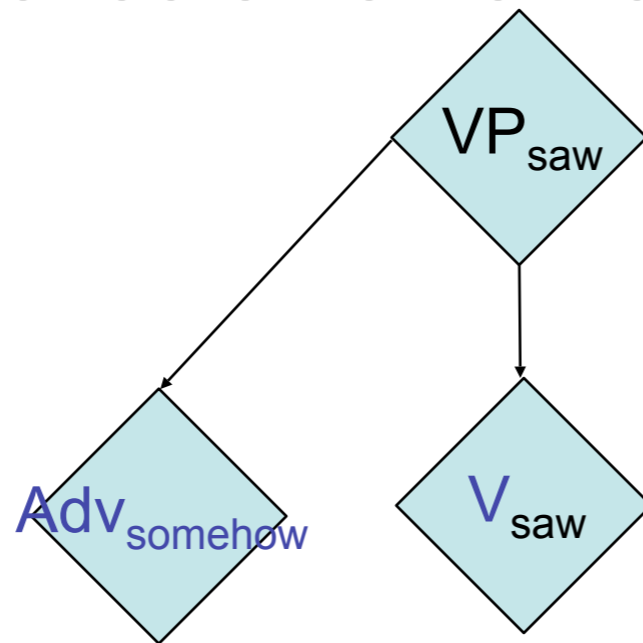
- Everything factors down to rules, then further. We're given the parent nonterminal and head word.
- Randomly generate the head child's nonterminal.



# Collins Model 1 (1997)

---

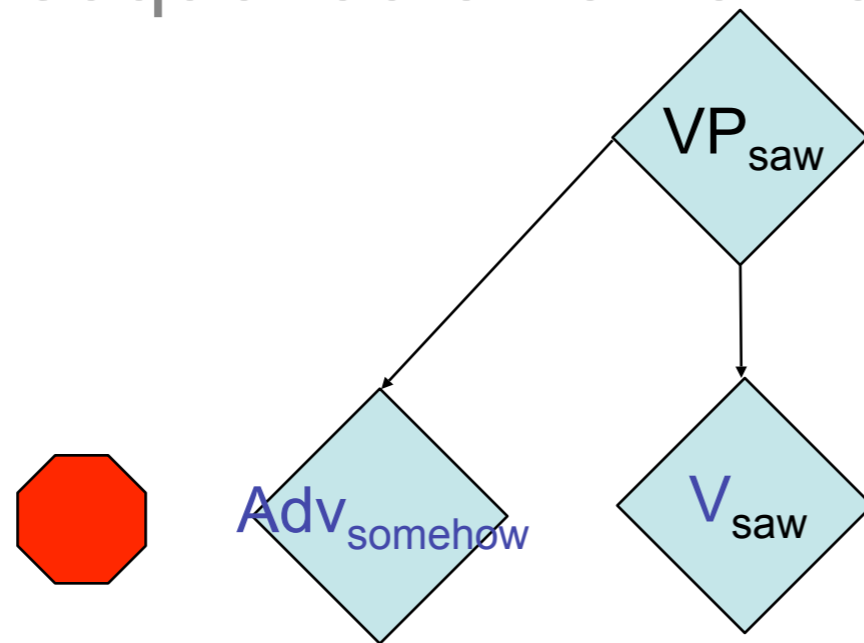
- Everything factors down to rules, then further. We're given the parent nonterminal and head word.
- Randomly generate the head child's nonterminal.
- Generate a sequence of left children.



# Collins Model 1 (1997)

---

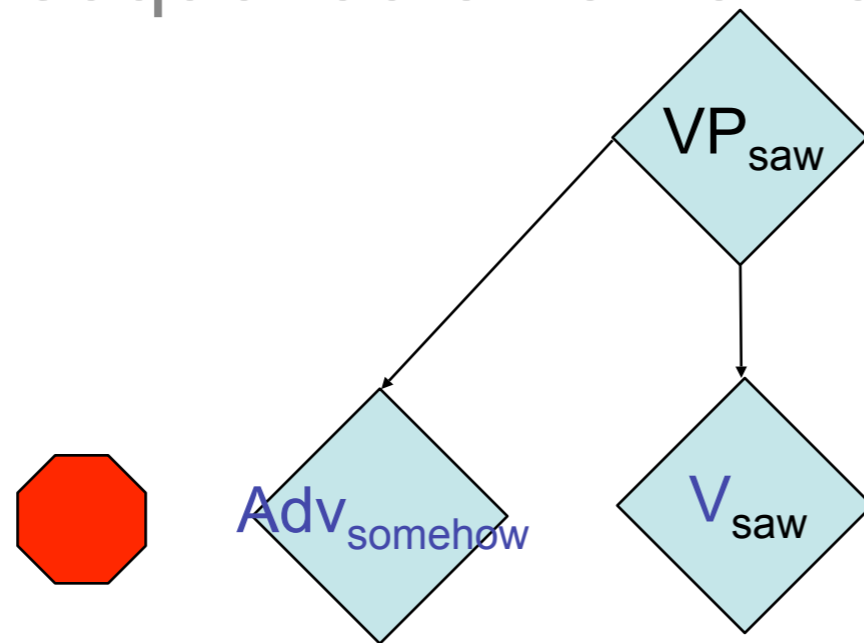
- Everything factors down to rules, then further. We're given the parent nonterminal and head word.
- Randomly generate the head child's nonterminal.
- Generate a sequence of left children.



# Collins Model 1 (1997)

---

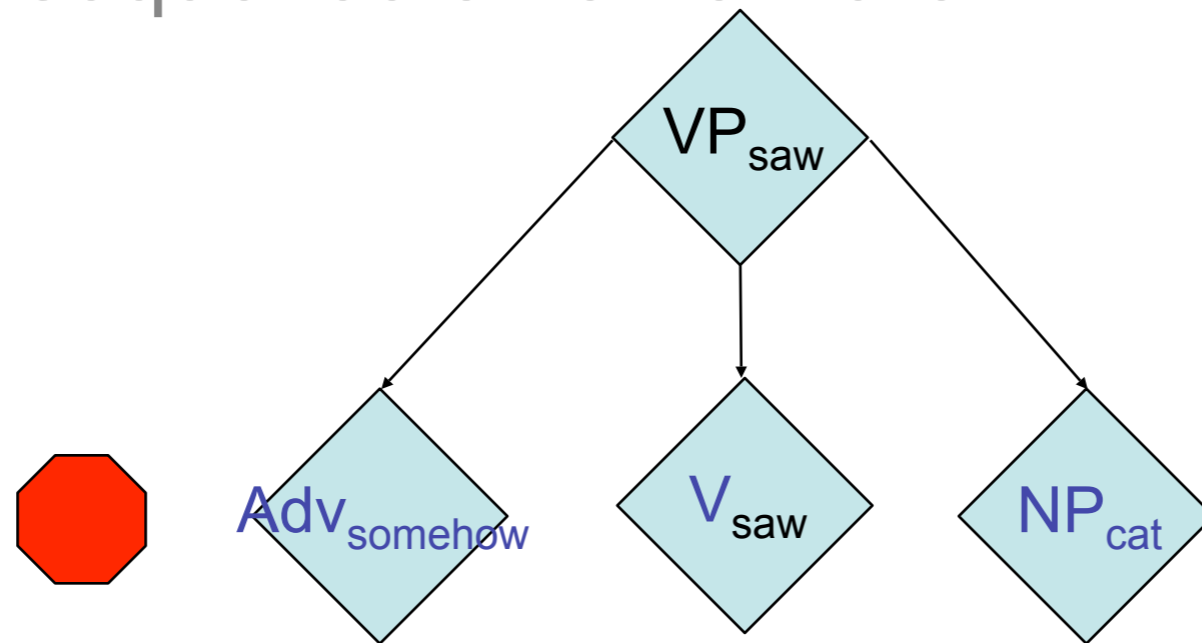
- Everything factors down to rules, then further. We're given the parent nonterminal and head word.
- Randomly generate the head child's nonterminal.
- Generate a sequence of left children.
- Then right.



# Collins Model 1 (1997)

---

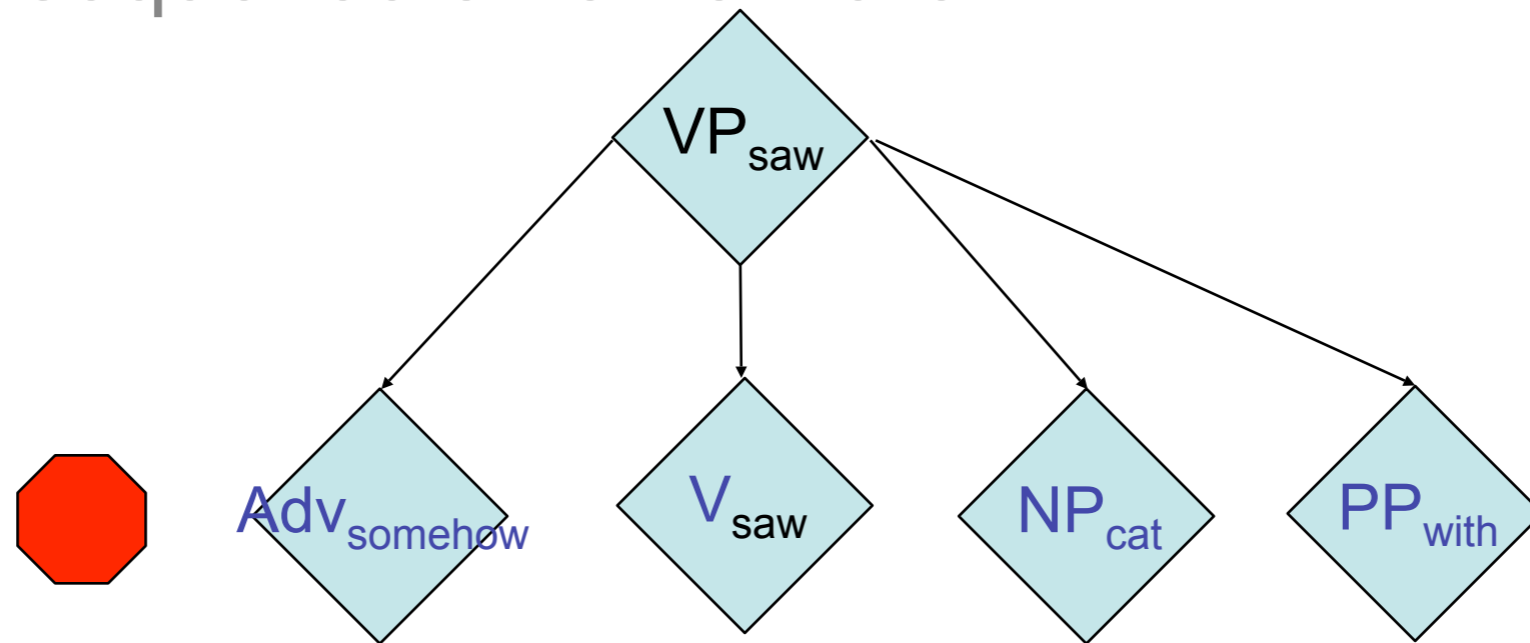
- Everything factors down to rules, then further. We're given the parent nonterminal and head word.
- Randomly generate the head child's nonterminal.
- Generate a sequence of left children.
- Then right.



# Collins Model 1 (1997)

---

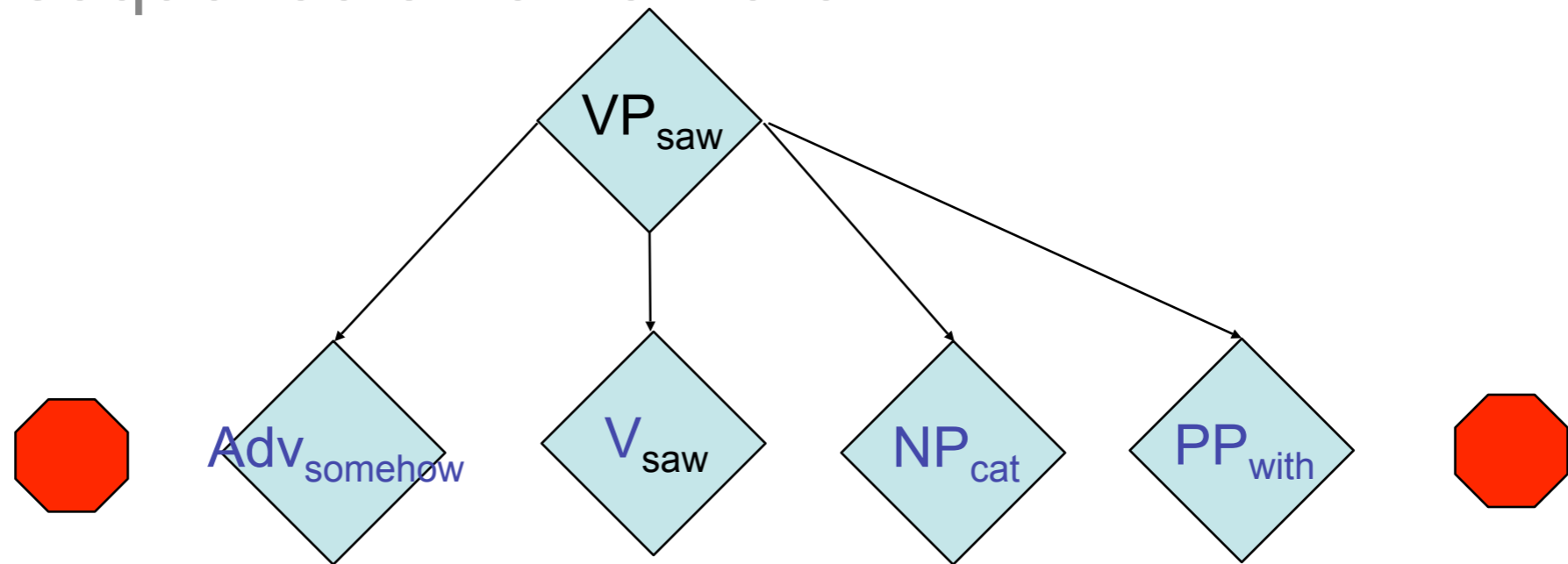
- Everything factors down to rules, then further. We're given the parent nonterminal and head word.
- Randomly generate the head child's nonterminal.
- Generate a sequence of left children.
- Then right.



# Collins Model 1 (1997)

---

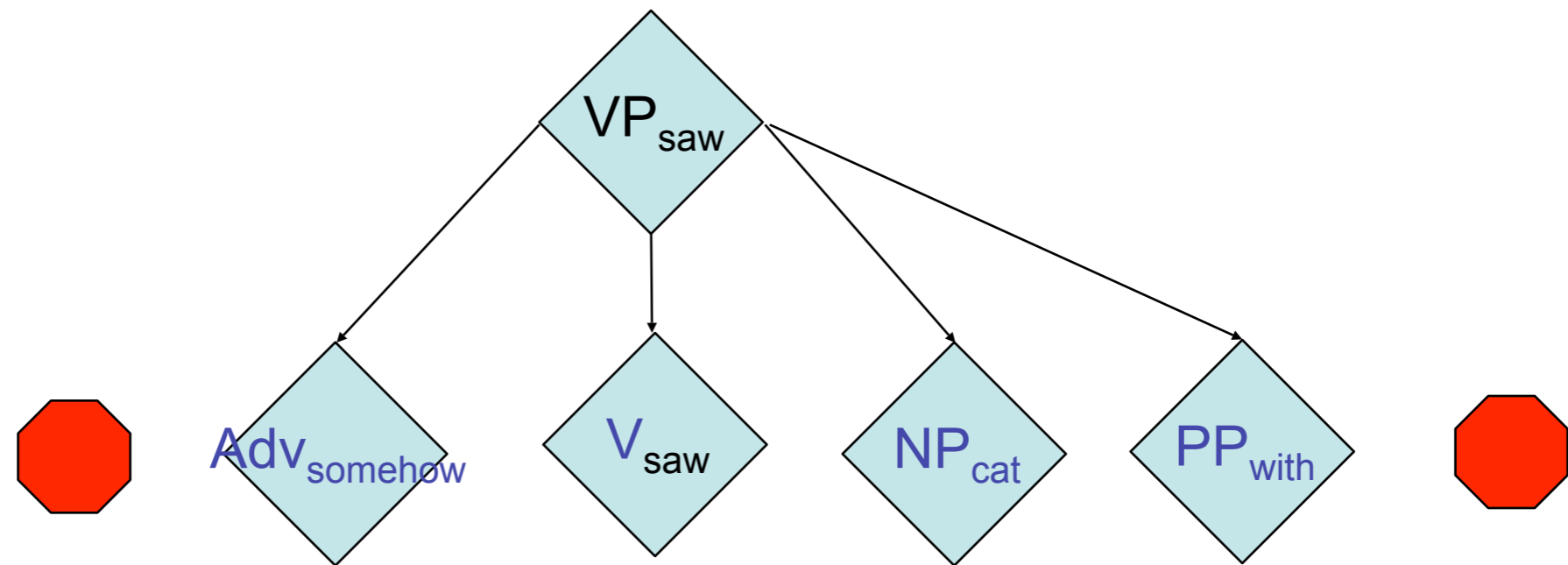
- Everything factors down to rules, then further. We're given the parent nonterminal and head word.
- Randomly generate the head child's nonterminal.
- Generate a sequence of left children.
- Then right.



# Collins Model 1 (1997)

---

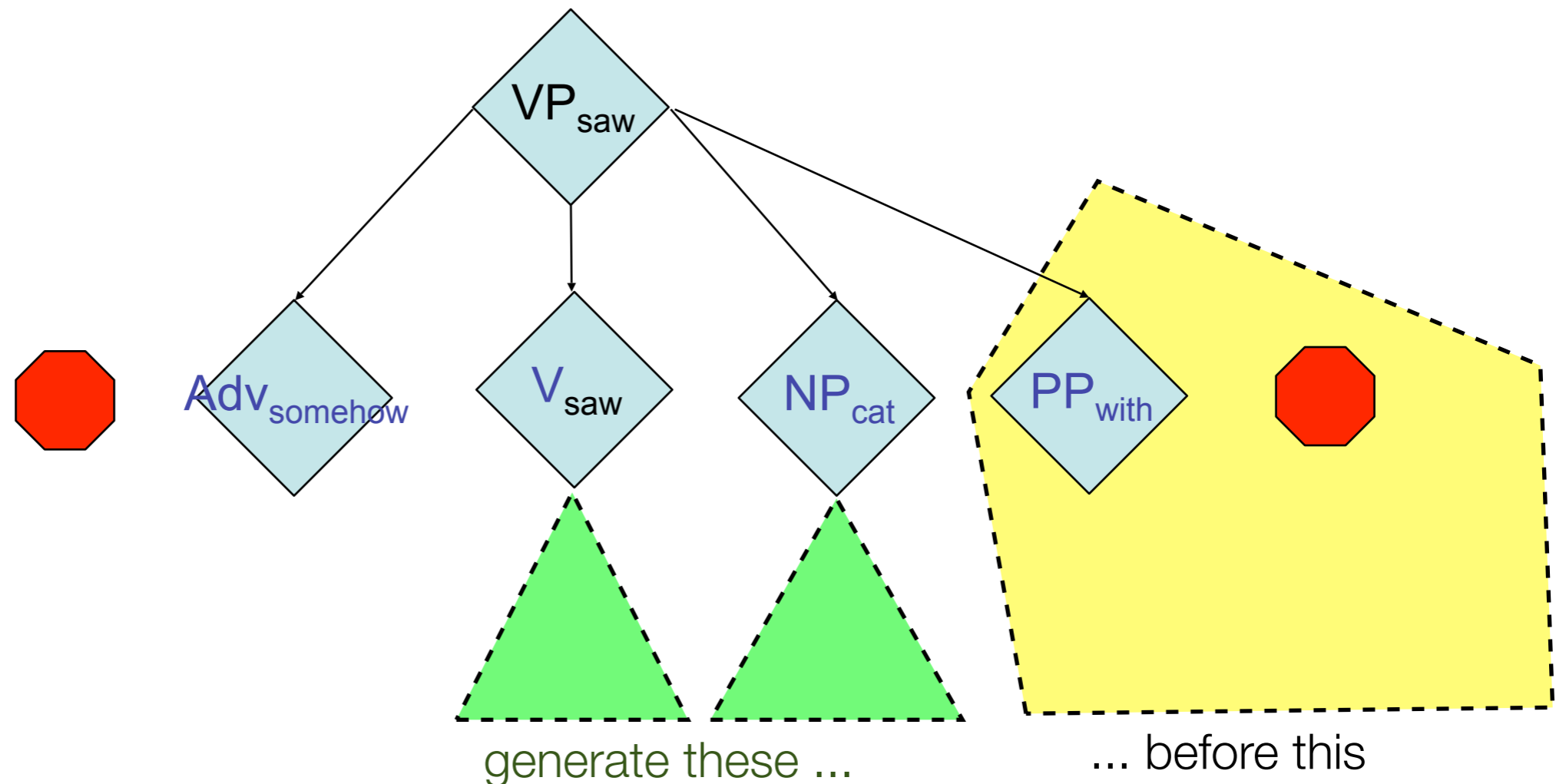
- Interesting twist: want to model the **distance** between head constituent and child constituent. How?



# Collins Model 1 (1997)

---

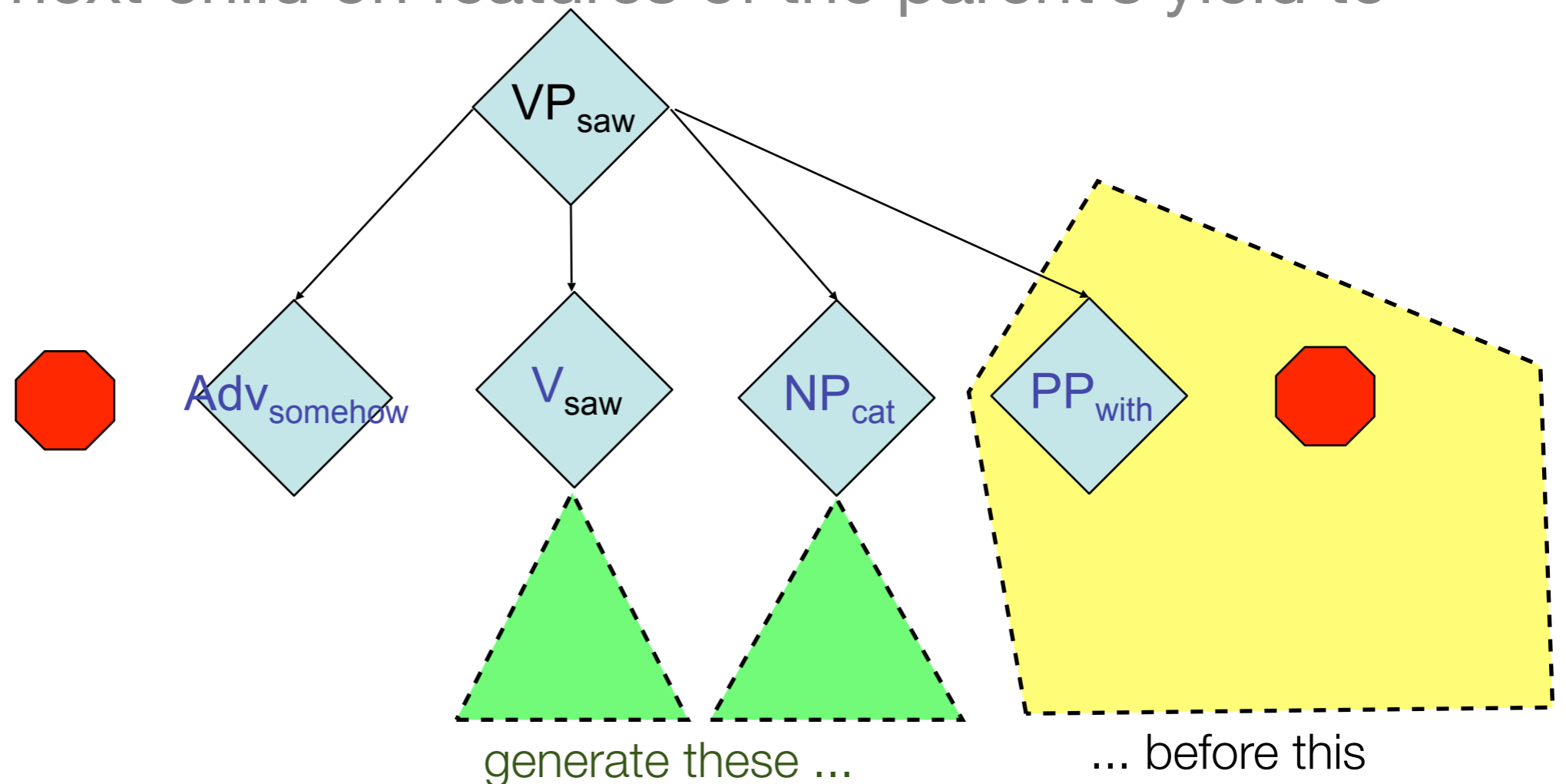
- Interesting twist: want to model the **distance** between head constituent and child constituent. How?
- Depth-first recursion.



# Collins Model 1 (1997)

---

- Interesting twist: want to model the **distance** between head constituent and child constituent. How?
- Depth-first recursion.
- Condition next child on features of the parent's yield to date.



# Collins Model 1 (1997)

---

- Interesting twist: want to model the **distance** between head constituent and child constituent. How?
- Depth-first recursion.
- Condition next child on features of the parent's yield to date.

$$p(\text{PP}_{\text{with}} \mid \text{VP}_{\text{saw, right}}, \text{“the cat who liked milk”}) \approx p(\text{PP}_{\text{with}} \mid \text{VP}_{\text{saw, right}}, \text{length} > 0, +\text{verb})$$

$$p(L_n, u_n, L_{n-1}, u_{n-1}, \dots, L_1, u_1, H, w, R_1, v_1, R_2, v_2, \dots, R_m, v_m \mid P, w)$$

$$= p(H \mid P, w)$$

$$\cdot \prod_{i=1}^n p(L_i, u_i \mid P, w, H, \text{left}, \Delta_i)$$

$$\cdot p(\text{stop} \mid P, w, H, \text{left}, \Delta_{n+1})$$

$$\cdot \prod_{i=1}^m p(R_i, v_i \mid P, w, H, \text{right}, \Delta'_i)$$

$$\cdot p(\text{stop} \mid P, w, H, \text{right}, \Delta'_{n+1})$$

# Collins Models 2 and 3 (1997)

---

- Model 2: Complements, adjuncts and subcategorization frames
  - Treebank decoration: -C on specifiers and arguments
  - Probability model: first pick set of complements (side-wise), must ensure they are all generated
  - *the issue was a bill funding Congress*
- Model 3: Wh-movement and extraction
  - Treebank decoration: “gap feature”
  - Probability model: gap feature “passed around the tree,” must be “discharged” as a trace element.
  - *the store that IBM bought last week*

# Other Points

---

- Unknown words at test time: any training word with count  $< 6$  becomes UNK
- Smoothing: deleted interpolation
- Tagging is just part of parsing (not a separate stage)
- Markov order increased in special cases:
  - within base noun phrases (NPBs) - first order
  - conjunctions (“and”) predicted together with second conjunct
  - punctuation (details in 2003 paper)

# Practical Notes

---

- Collins parser is freely available
- Bikel replicated the Collins parser cleanly in Java
  - Easier to re-train
  - Easier to plug-and-play with different options
  - Multilingual support
  - May be faster (with current Java) - I'm not sure