Natural Language Parsing with Context-Free Grammars

SPFLODD

September 10, 2013

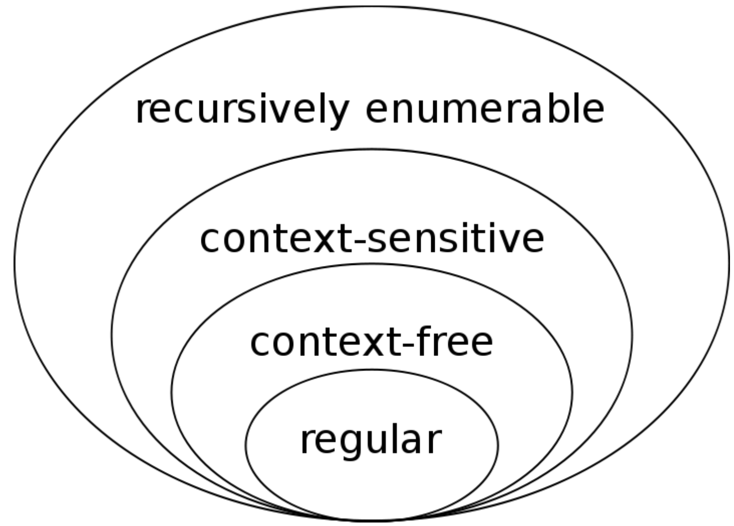
What is "Parsing"?

- General answer: analyze text with respect to some theory.
- Usually it means **syntactic analysis**.
- **Syntax**: branch of linguistics dealing with how words and phrases are *ordered* to create well-formed sentences.
  - As in programming languages, syntax is understood as relevant to the mapping from strings to their meanings.
- Different theories of syntax → different kinds of parsing.
  - Today we'll talk about context-free syntax
  - Thursday we'll talk about dependency syntax

Formal Stuff First

Context-Free Grammars

- Chomsky hierarchy:
- Informally, CFGs can represent center-embedding, which regular grammars can't.



- Classic argument from Chomsky (1956): NL is not regular.
  - Pumping lemma-type argument on (the Noun)^n (Verb-past)$^{n-1}$ VP

Context-Free Grammars

- Alphabet Σ
- Set of variables N
- Start symbol $S \in N$
- Rewrite rules: $X \rightarrow \alpha$, where $X \in N$ and $\alpha \in (N \cup \Sigma)^*$

- CNF: Assume $\alpha \in N^2 \cup \Sigma$. Can always convert to CNF.

- Grammars for NL usually have nonterminals like S, NP, VP, PP, and preterminals like N, V, Adj, Adv, …
  - Tokens of labeled spans are called **constituents**.

Probabilistic Context-Free Grammar

- Associate a multinomial distribution over *right-hand sides* to the set of rules sharing a *left-hand side*.
  - Conditional probability of "children" given "parent."
- Generative story:
  1. Instantiate the start symbol S as a single red node.
  2. While there are any red symbols:
     1. Choose a red node X and color it white.
     2. Draw $\alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_k \rangle$ according to $p(* \mid X)$.
     3. Add $\langle \alpha_1, \alpha_2, \ldots, \alpha_k \rangle$ to the tree as the sequence of children of the node X you selected.
     4. For any α_i that are nonterminals, color them red; color the terminals white.

Like “Branching” Bayesian Networks

- Everything in a subtree is conditionally independent of everything else given its parent.
- A node’s label is conditionally independent of its descendents given its children.

- But not easy to capture in a Bayesian network:
  - variable length derivations of the grammar
  - joint model of tree structure *and* labels
  - direct dependency between any span’s label (or lack of label) and any potential parent, child, or sibling

HMMs are Special PCFGs

- Alphabet Σ
- N = HMM states Q
- Start state q_0
- Rules

  $q \rightarrow x\ q'$ with probability $p_{emit}(x \mid q)\ p_{trans}(q' \mid q)$

  $q \rightarrow \varepsilon$ with probability $p_{trans}(stop \mid q)$

Weighted Context-Free Grammar

- Don't need a generative story; just assign weights to rules.
  - Can featurize
- Like a Markov network, but representing a WCFG as a MN is not elegant.

Parsing Natural Language

Penn Treebank (Marcus et al., 1993)

- A million words (40K sentences) of *Wall Street Journal* text (late 1980s).
  - This is important to remember!
- Parsed by experts; consensus parse for each sentence was published.
- The structure is basically what you'd expect from a PCFG.
  - Tends to be "flat" where there's controversy.
  - Some "traces" for extraposed elements.
- Attempts to be theory-neutral, probably more accurate to say that it represents its own syntactic theory.
- Many other treebanks now available in other languages.

Example

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
     (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
       (JJ old) )
     (, ,) )
   (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
       (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
       (NP-TMP (NNP Nov.) (CD 29) )))
   (. .) ))
```

Example

```
( (S
    (NP-SBJ-1
      (NP (NNP Rudolph) (NNP Agnew) )
     (, ,)
      (UCP
        (ADJP
          (NP (CD 55) (NNS years) )
         (JJ old) )
       (CC and)
        (NP
         (NP (JJ former) (NN chairman) )
         (PP (IN of)
            (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC) ))))
     (, ,) )
    (VP (VBD was)
      (VP (VBN named)
        (S
          (NP-SBJ (-NONE- *-1) )
         (NP-PRD
            (NP (DT a) (JJ nonexecutive) (NN director) )
           (PP (IN of)
              (NP (DT this) (JJ British) (JJ industrial) (NN conglomerate) ))))))
    (. .) ))
```

Evaluation

- Take a sentence from the test set.
- Use your parser to propose a hypothesis parse.
- Treebank gives you the correct parse.
- Precision and recall on labeled (or unlabeled) constituents.
  - Also, average number of crossing brackets (compared to correct parses) in your hypotheses.
- The training/development/test split has been held constant for a long time; possibly a cause for concern.

Basic Algorithms

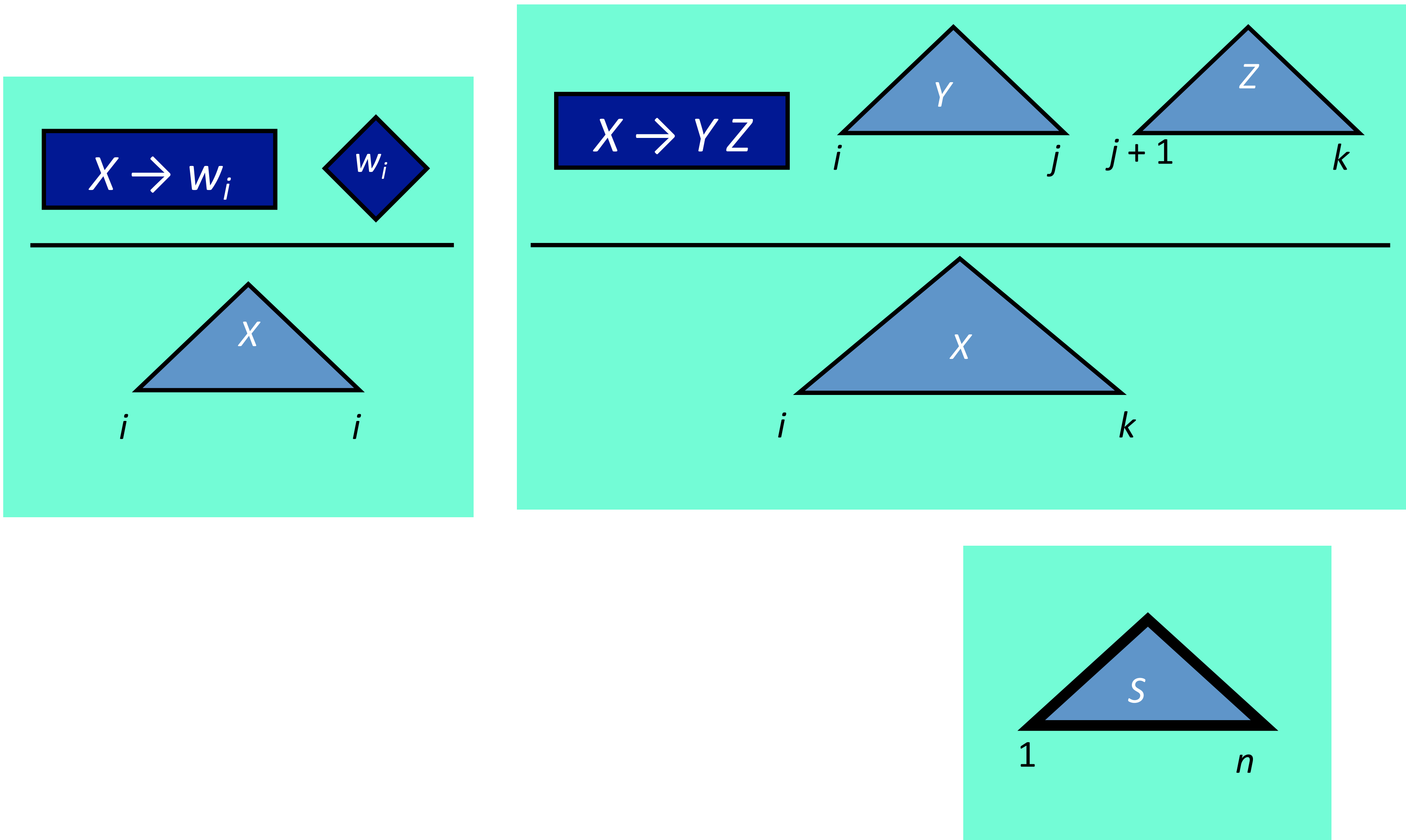
CFG Parsing

- Given a treebank of reasonable size, the grammar we extract will be ambiguous.
  - Algorithms used for programming languages will not work.
- The most common approaches are based on two dynamic programming algorithms:
  - Cocke-Kasami-Younger (CKY) algorithm
  - Earley's algorithm
- Originally these were not weighted, but today we assume rules have weights.

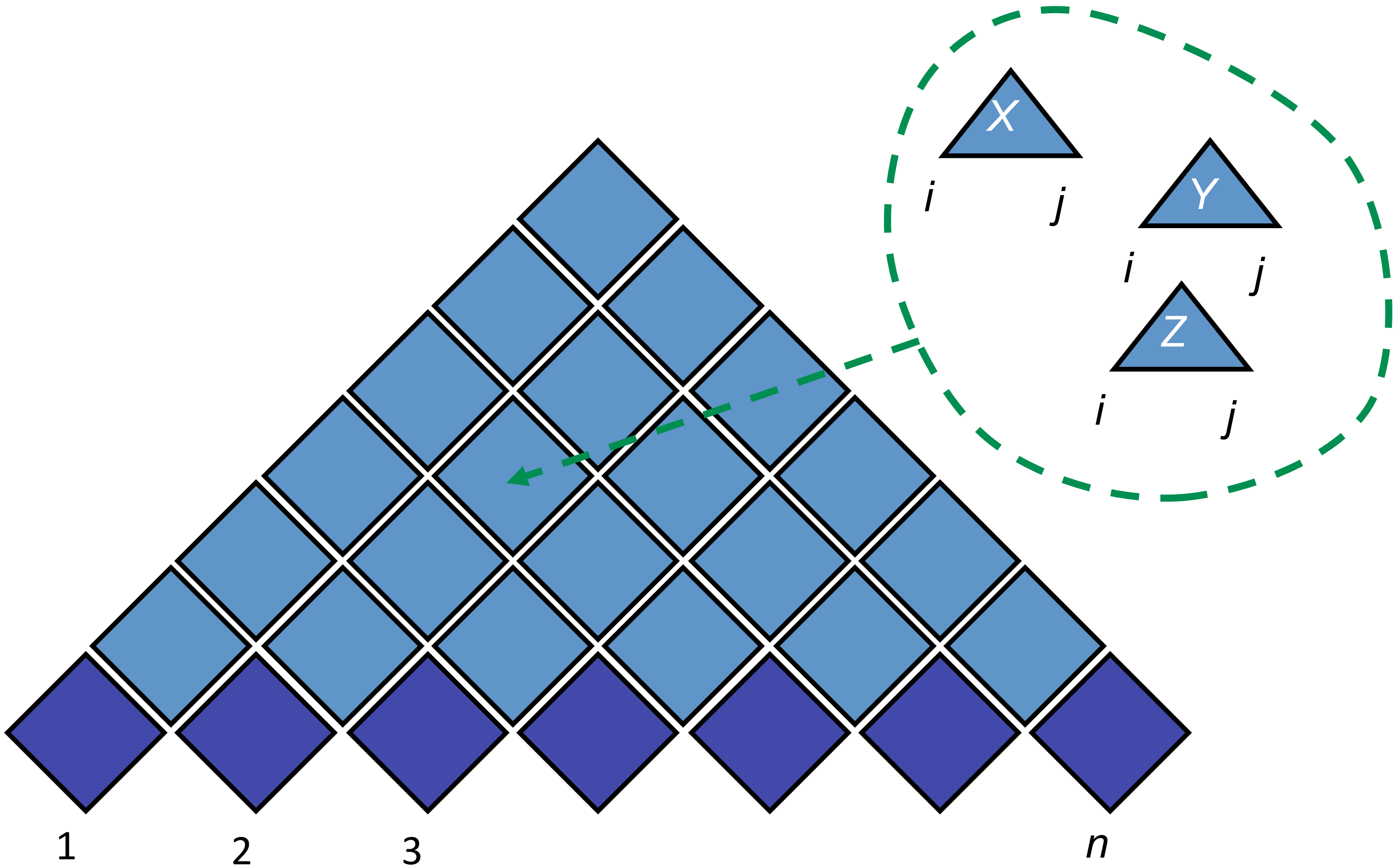
CKY: Weighted Logic Program

- constit(X, I, I) max= word(W, I) × unary(X, W).
- constit(X, I, K) max= constit(Y, I, J)
  × constit(Z, J+1, K)
  × binary(X, Y, Z).
- goal max= constit(S, 1, N)
  × length(N) × startsymbol(S).

Visualizing Probabilistic CKY

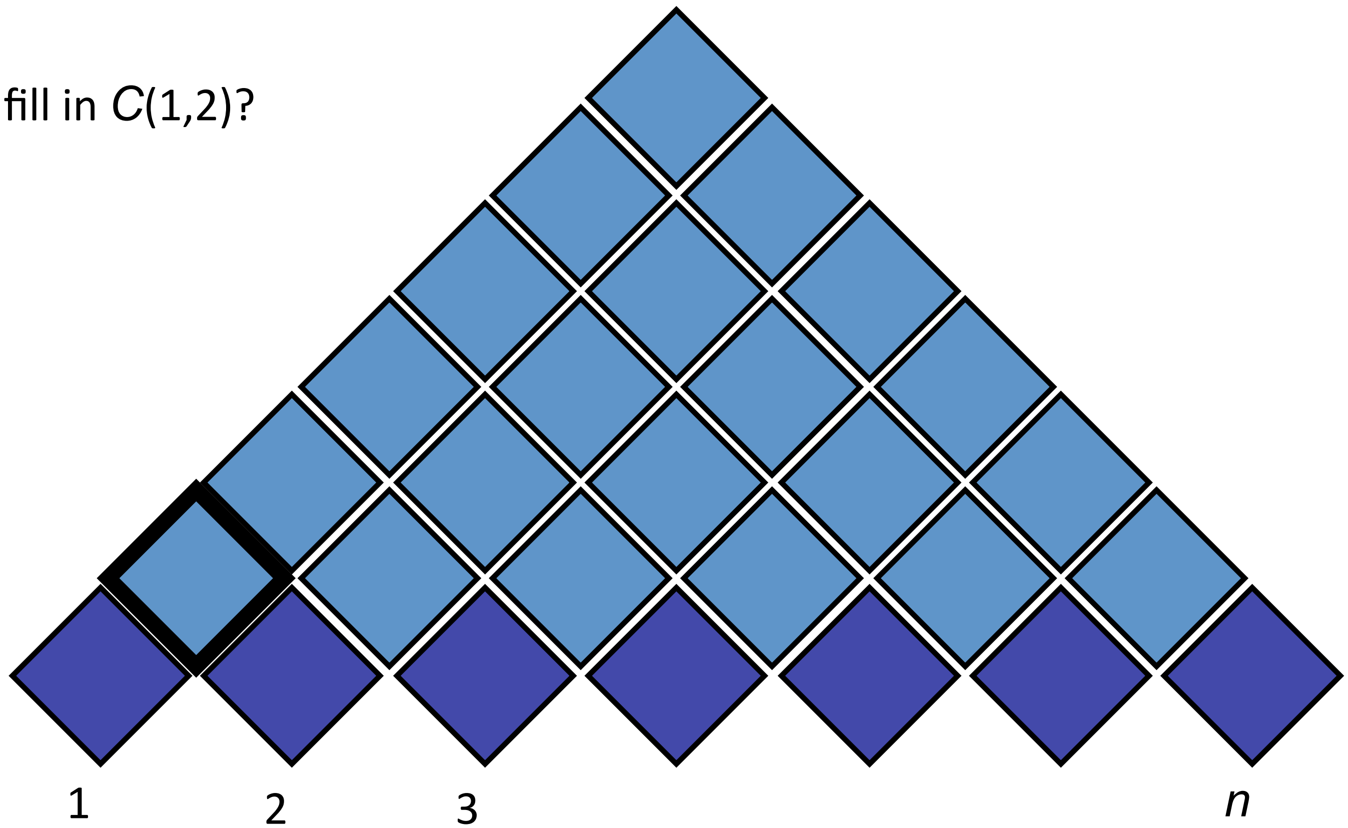


Visualizing Probabilistic CKY



Visualizing Probabilistic CKY

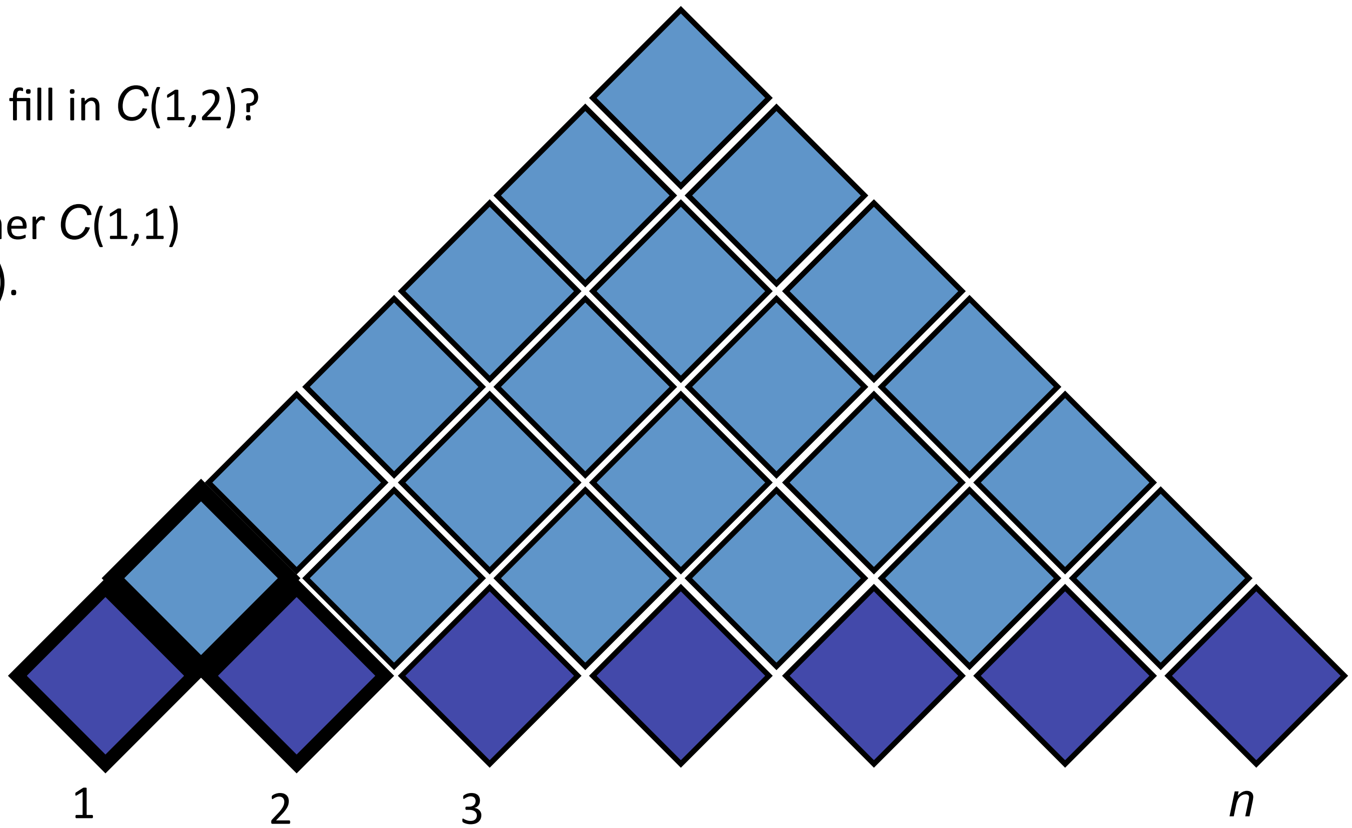
How do we fill in $C(1,2)$?



Visualizing Probabilistic CKY

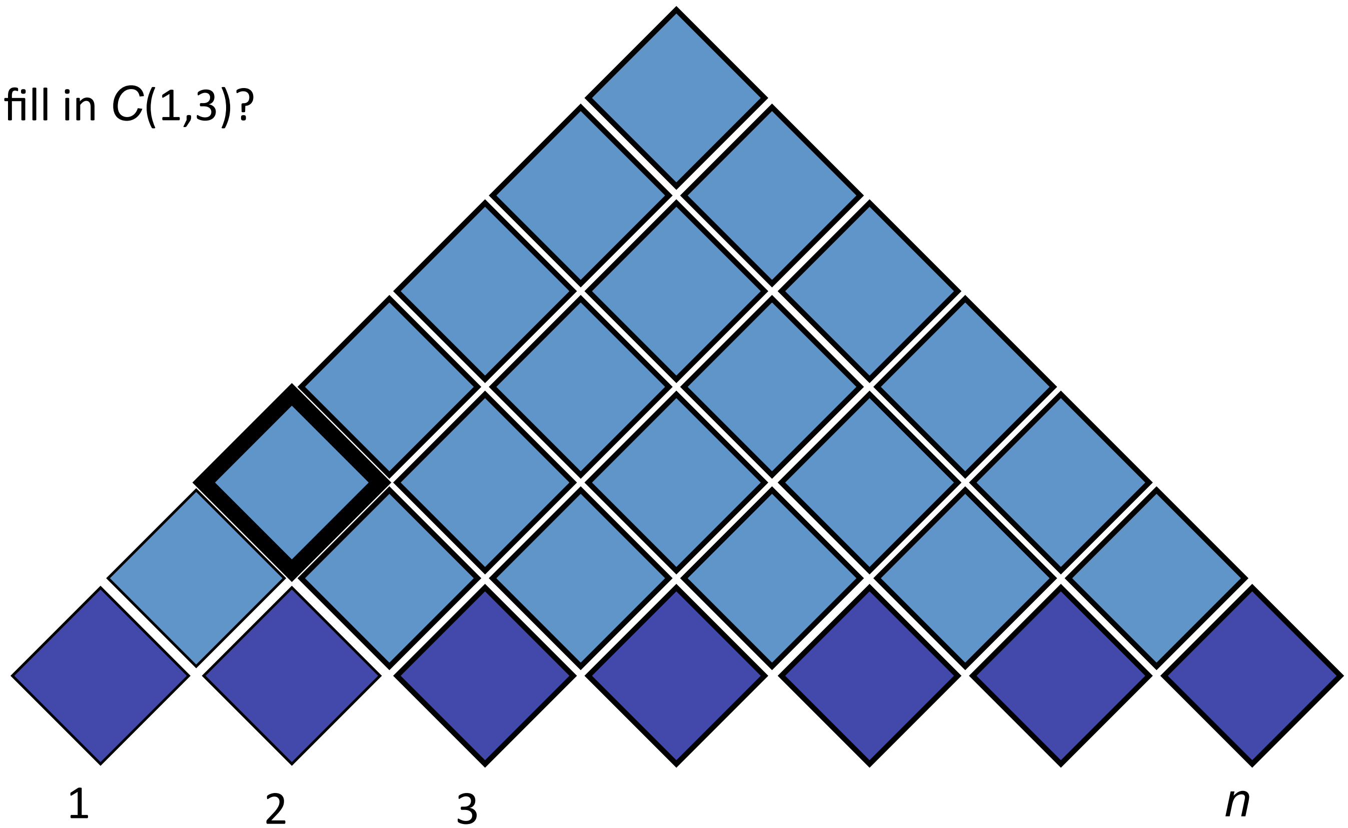
How do we fill in $C(1,2)$?

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



Visualizing Probabilistic CKY

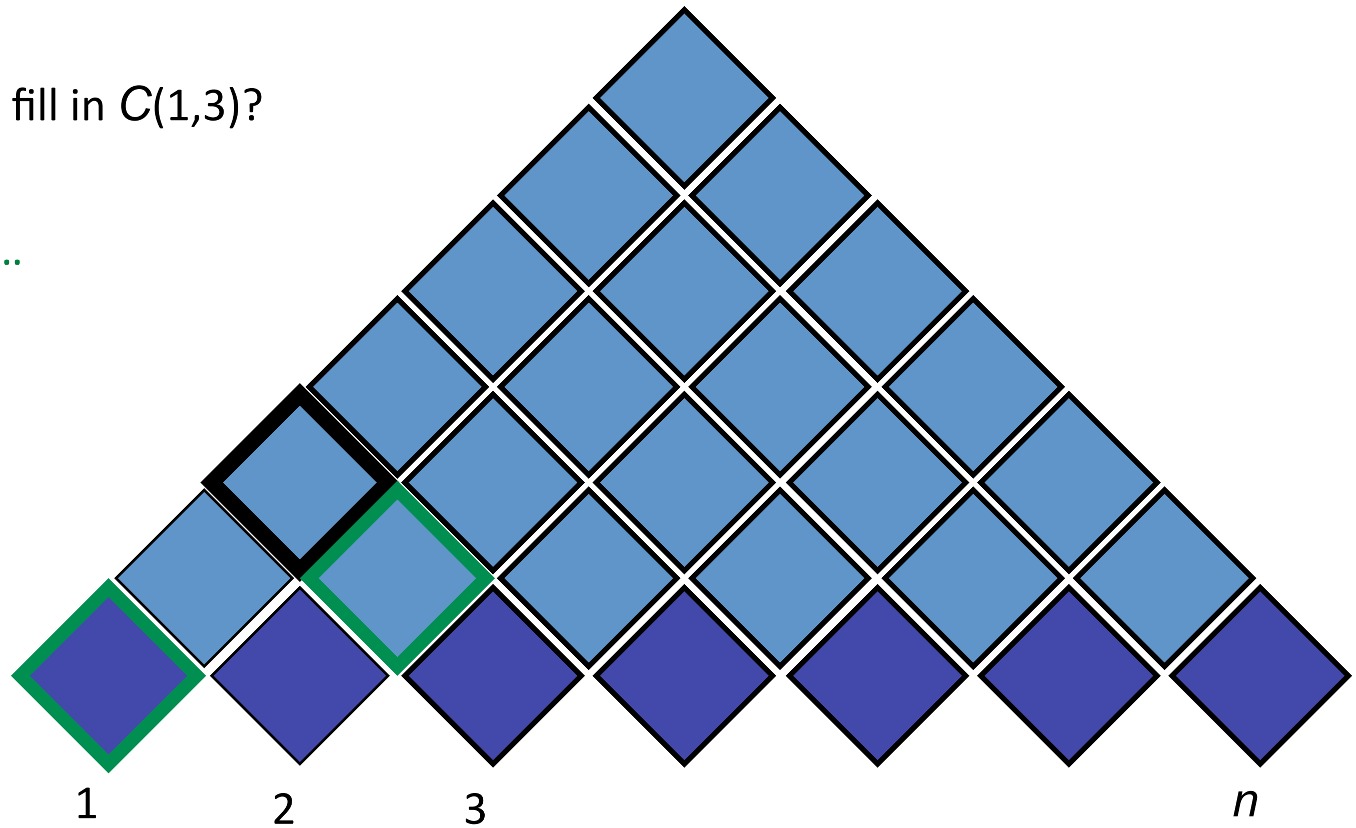
How do we fill in $C(1,3)$?



Visualizing Probabilistic CKY

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

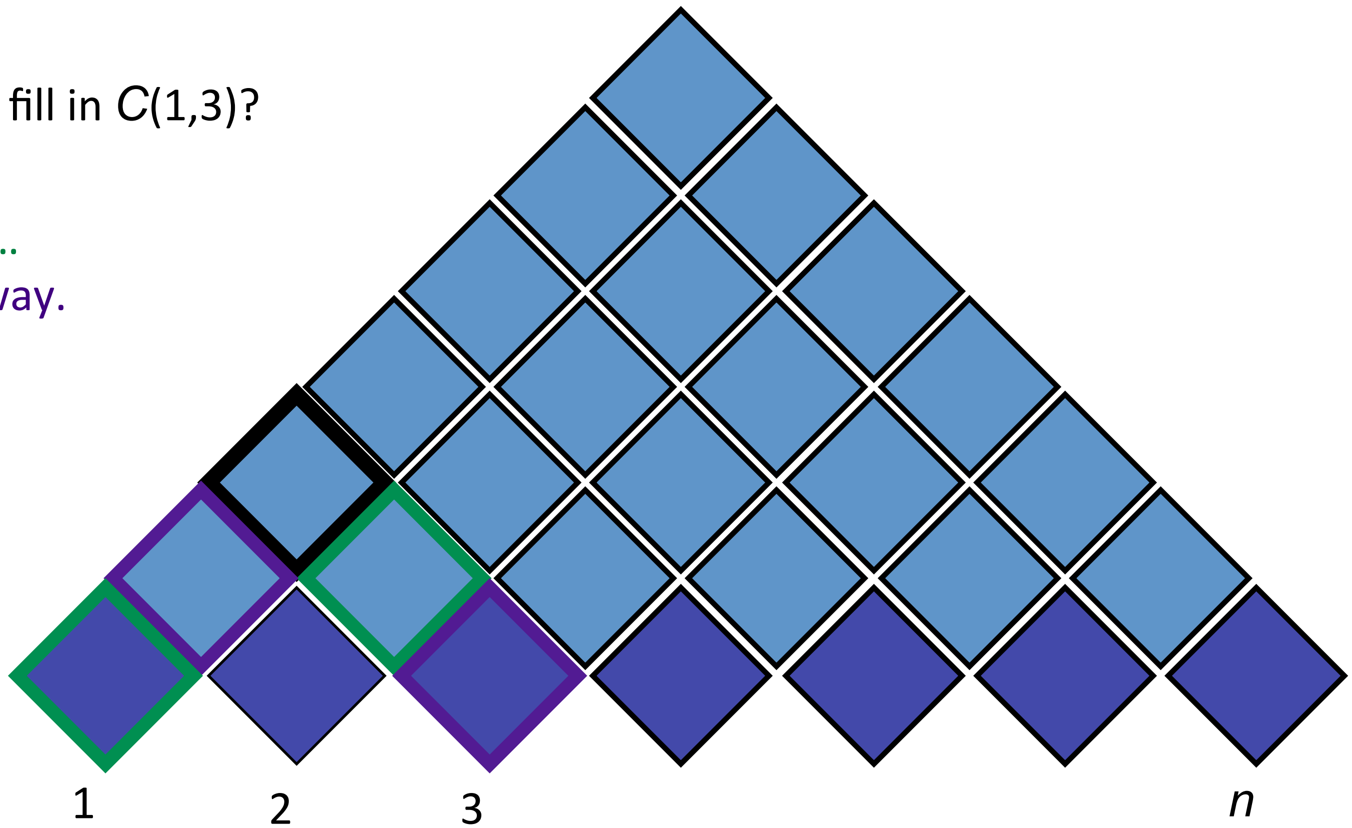
One way …



Visualizing Probabilistic CKY

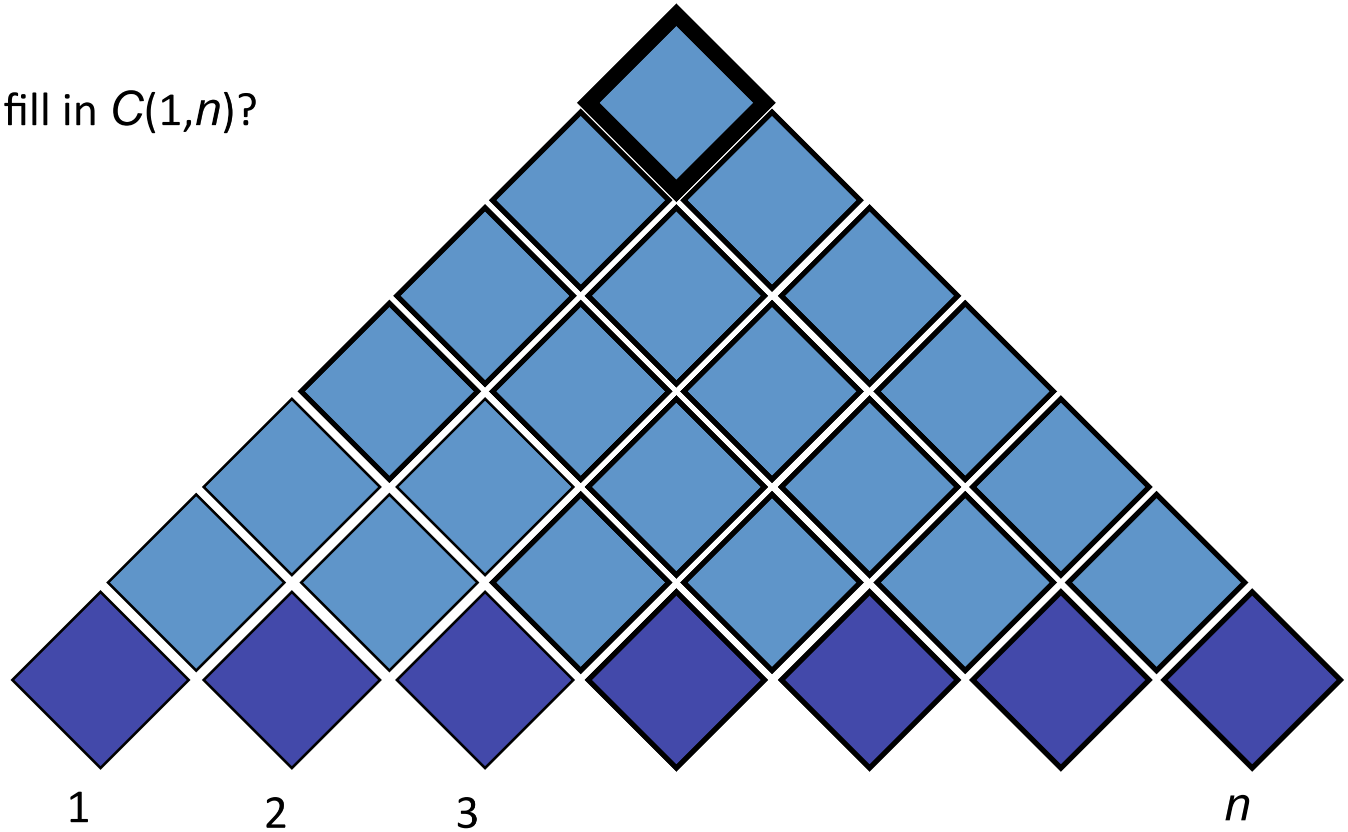
How do we fill in $C(1,3)$?

One way …
Another way.



Visualizing Probabilistic CKY

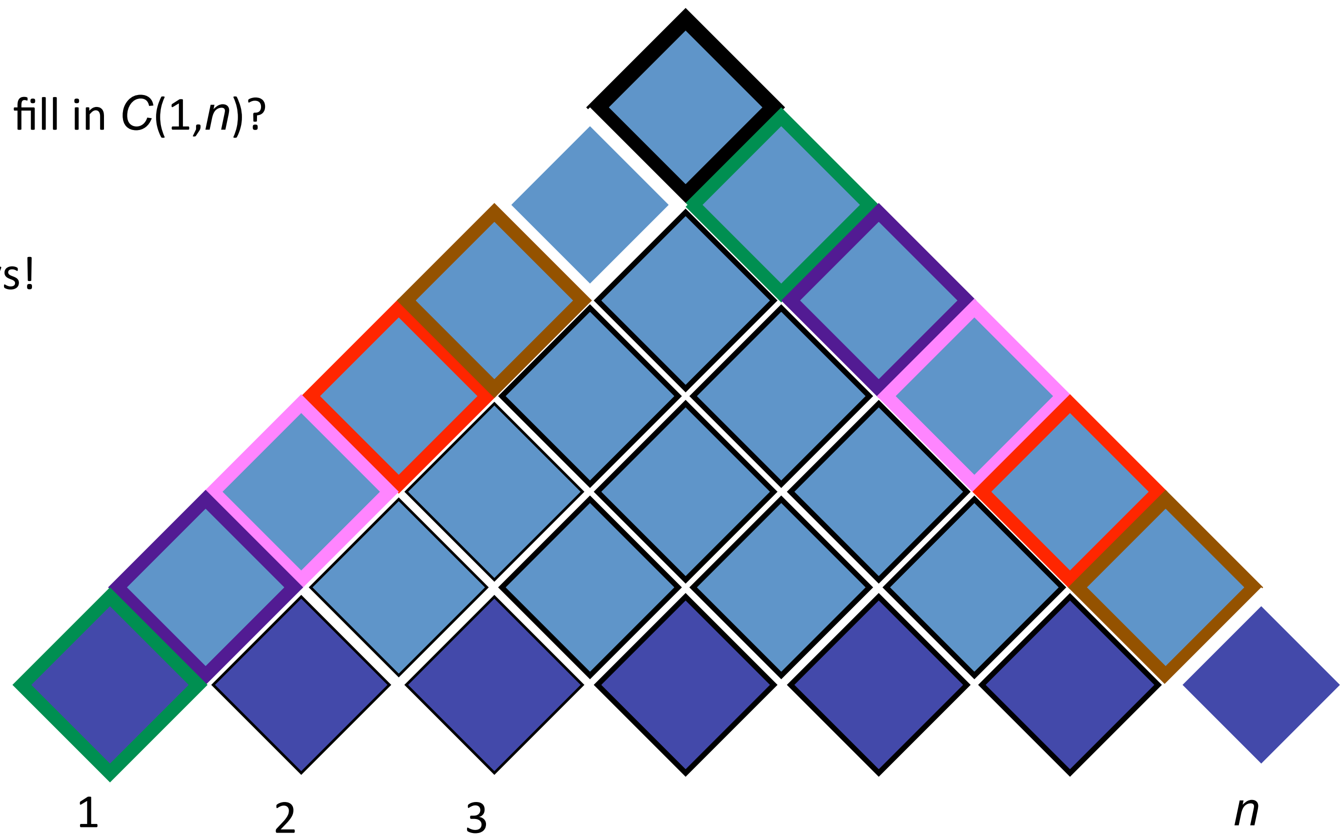
How do we fill in $C(1,n)$?



Visualizing Probabilistic CKY

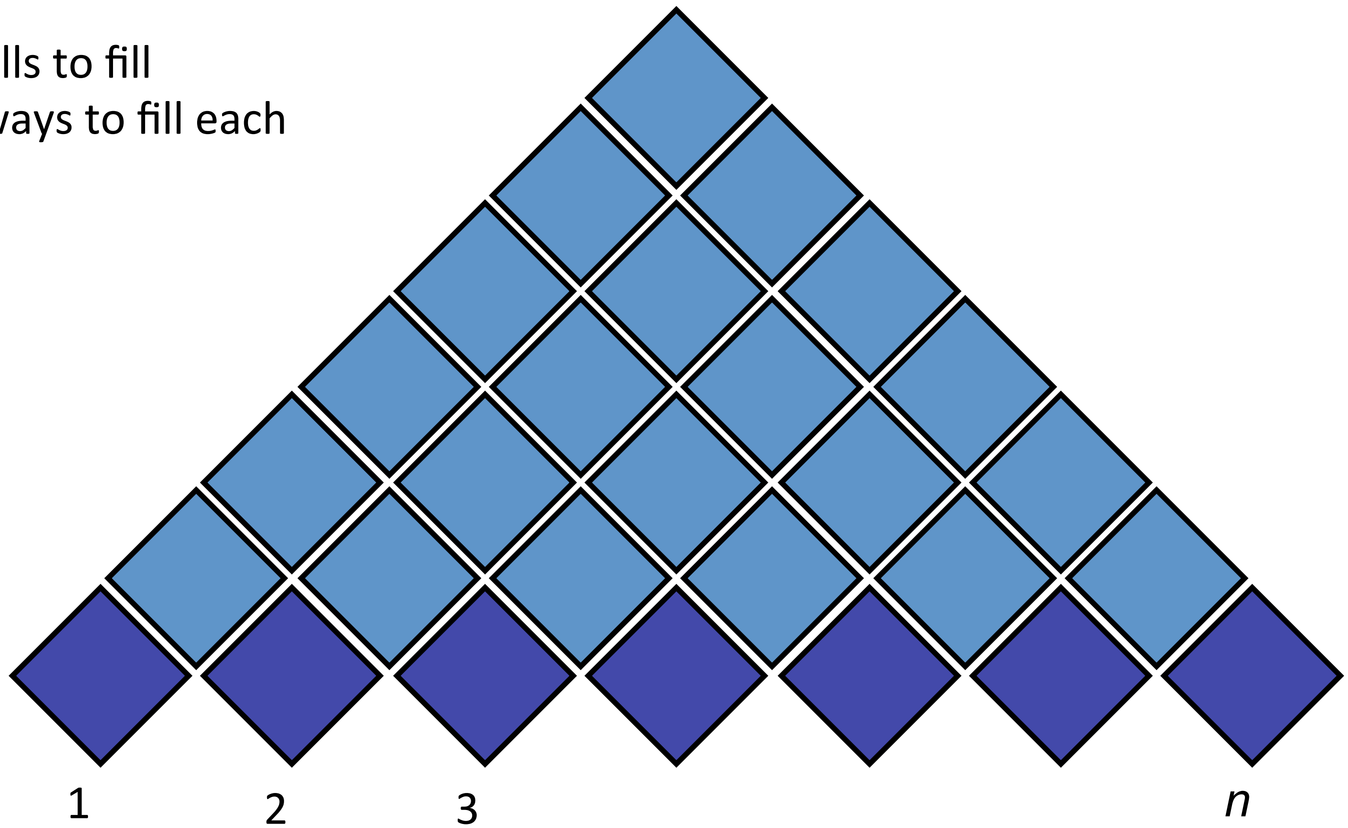
How do we fill in $C(1,n)$?

n - 1 ways!



Visualizing Probabilistic CKY

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



Earley's Algorithm

need(X, I) max= constit(_/Xα, _, I).

need(S, 0) max= startsymbol(S).

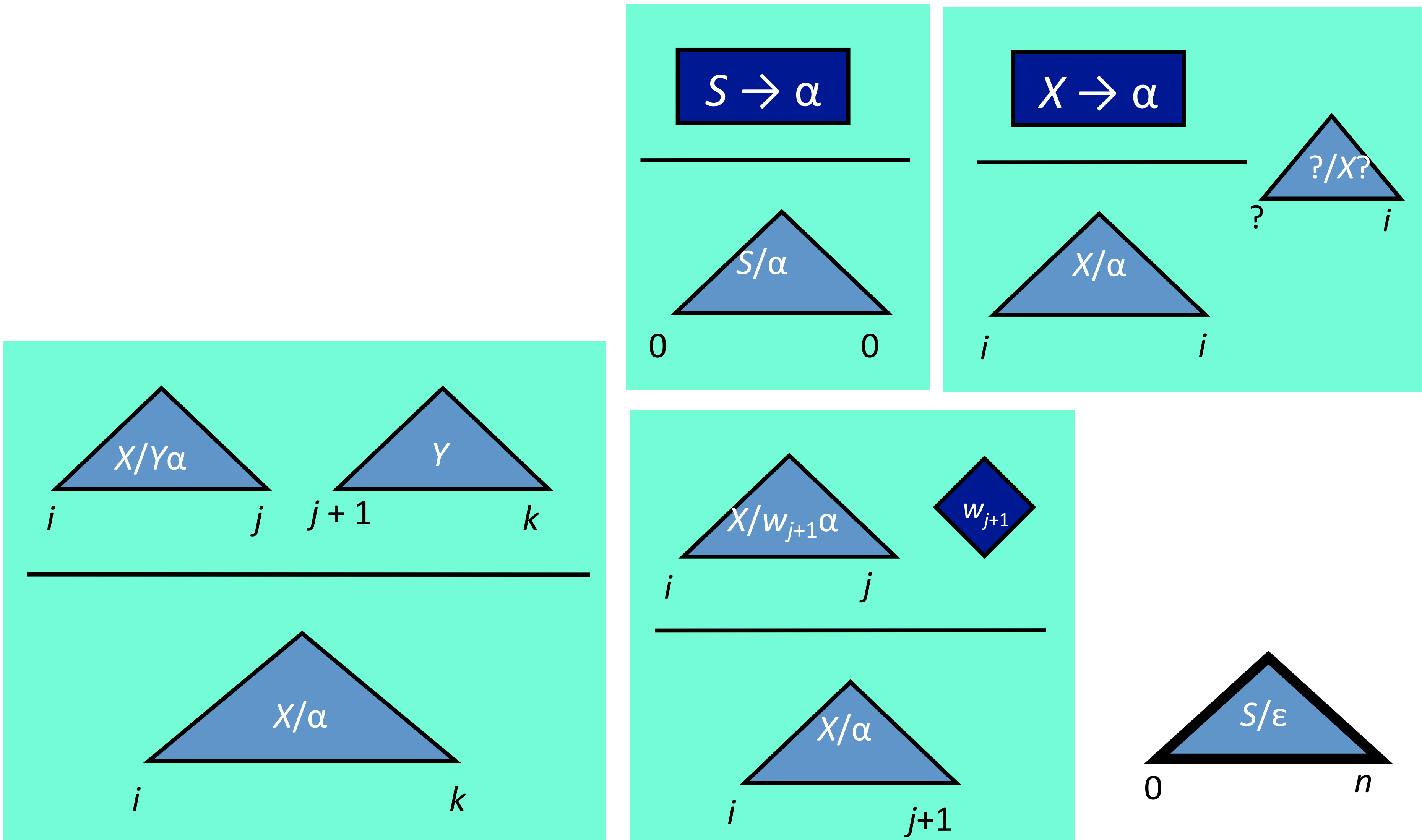
constit(X/α, I, I) max= rewrite(X, α) **whenever** need(X, I). *predict*

constit(X/α, I, J+1) max= constit(X/W α, I, J) × word(W, J + 1). *scan*

constit(X/α, I, K) max= constit(X/Yα, I, J) × constit(Y/ε, J, K). *complete*

goal max= constit(S/ε, 0, N) × length(N) × startsymbol(S).

Visualizing Probabilistic Earley's



CKY vs. Earley's

- Both $O(n^3)$ runtime, $O(n^2)$ space
- Neither requires weights to be probabilities, just like Viterbi.
- 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
- If you're into logic programming, there are interesting ways to derive each of these from the other.

Parsing in Reality

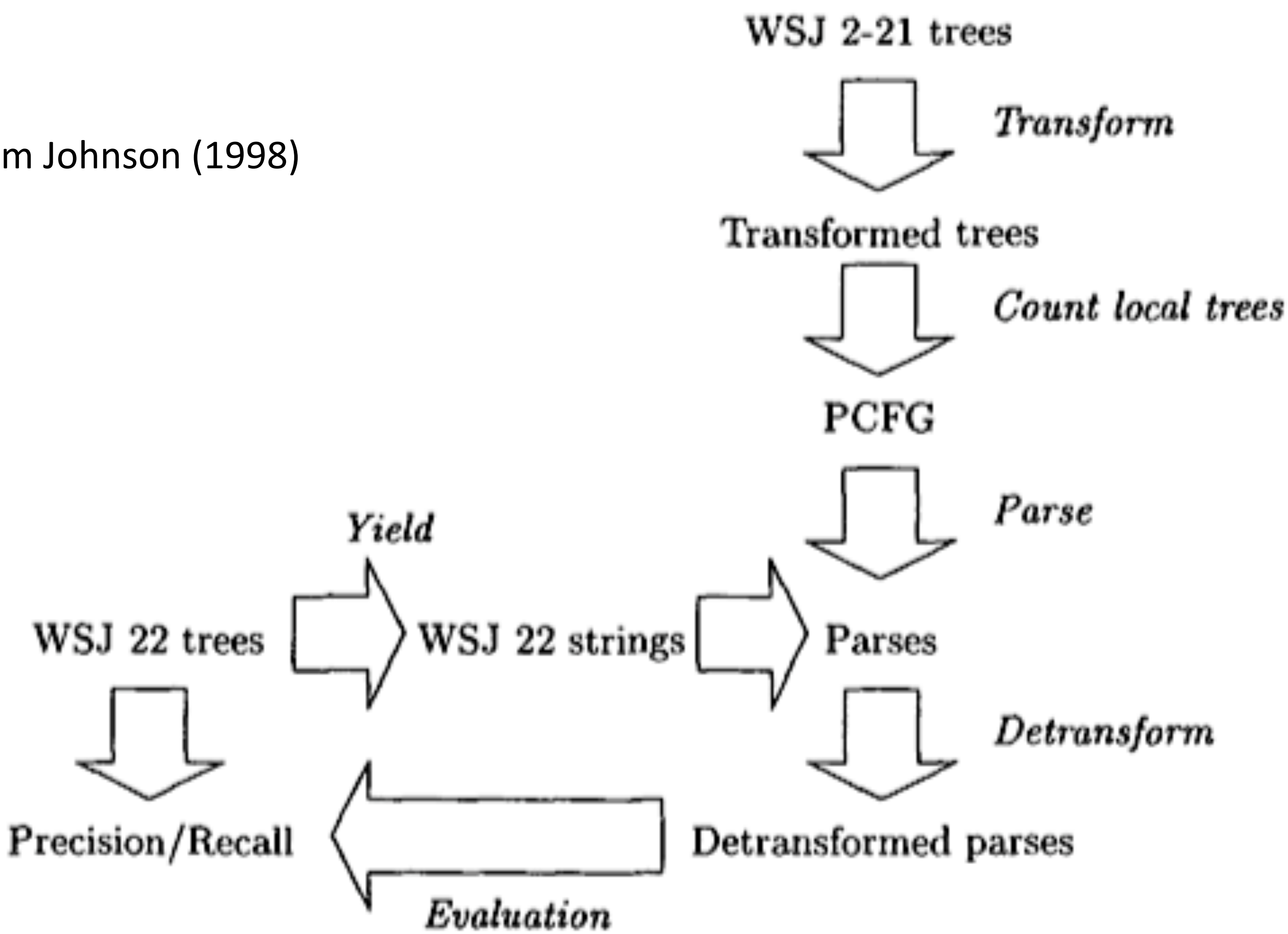
- Generally speaking, few industrial-strength parsers actually call CKY or Earley's.
- Extensions to the basic CFG model (next topic) make reduction to CFG expensive.
- Standard techniques:
  - Beam search
  - Agenda-based approximations with pruning and/or A*
  - "Coarse-to-fine"
  - "Cube pruning" that makes use of local k-best lists (Huang and Chiang, 2005)
  - Shift-reduce-style algorithms with search

Better CFGs

Training Parsers In Practice

- Transformations on trees
  - Some of these are generally taken to be crucial
  - Some are widely debated
  - Lately, people have started **learning** these transformations
- Smoothing is crucial; the grammars that result from transformed trees have lots more rules and therefore more parameters.

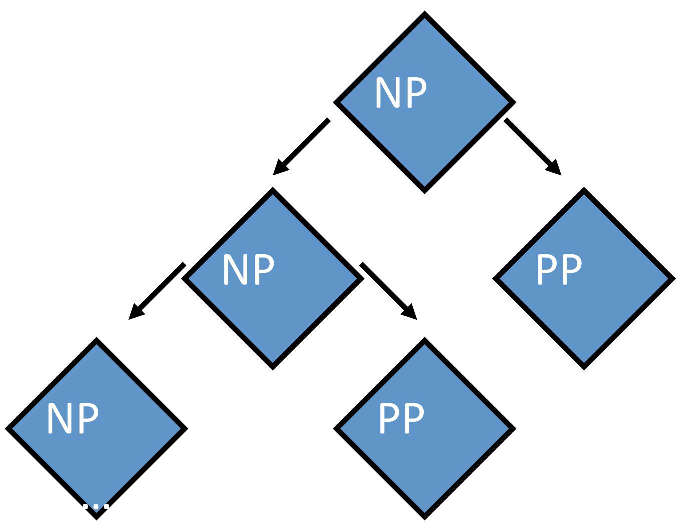
from Johnson (1998)

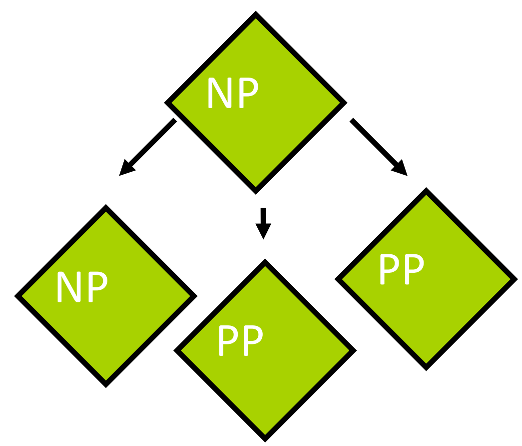


Parent Annotation

$NP \rightarrow^{p} NP\ PP$

$NP \rightarrow^{q} NP\ PP\ PP$



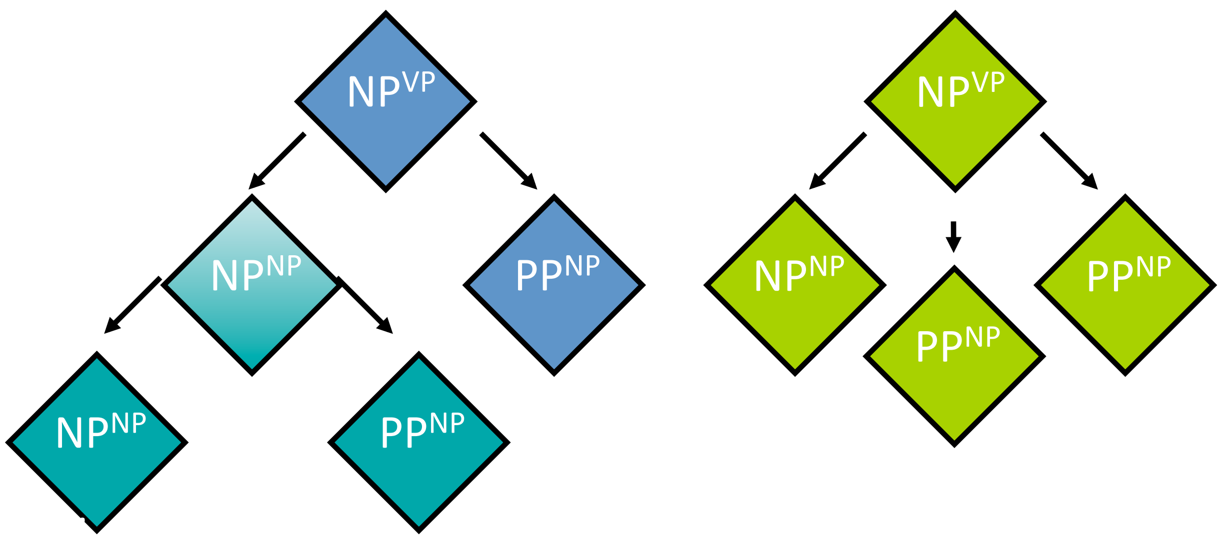


Parent Annotation

$NP^{VP} \rightarrow^{p} NP^{NP}\ PP^{NP}$

$NP^{NP} \rightarrow^{r} NP^{NP}\ PP^{NP}$

$NP^{VP} \rightarrow^{q} NP^{NP}\ PP^{NP}\ PP^{NP}$



Parent Annotation

- Another way to think about it …
- Before: $$p(\text{tree}) = \prod_{n \in nodes(\text{tree})} \rho(childsequence(n) \mid n)$$
- Now: $$p(\text{tree}) = \prod_{n \in nodes(\text{tree})} \rho(childsequence(n) \mid n, parent(n))$$
- This could conceivably **help** performance (weaker independence assumptions)
- This could conceivably **hurt** performance (data sparseness)

Parent Annotation

- From Johnson (1998):
- PCFG from WSJ Treebank: 14,962 rules
  - Of those, 1,327 would **always** be subsumed!
- After parent annotation: 22,773 rules
  - Only 965 would always be subsumed!
- Recall 69.7% $\rightarrow$ 79.2%; precision 73.5% $\rightarrow$ 80.0%
- Trick: check for subsumed rules, remove them from the grammar $\rightarrow$ faster parsing.

Head Annotation

- "I love all my children, but one of them is **special**."

$\boxed{\mathrm{S} \rightarrow \mathrm{NP}\ \underline{\mathrm{VP}}}$

$\boxed{\mathrm{VP} \rightarrow \underline{\mathrm{VBD}}\ \mathrm{NP}}$

$\boxed{\mathrm{NP} \rightarrow \mathrm{DT}\ \underline{\mathrm{NNS}}\ \mathrm{PP}}$

- Heads not in the Treebank.
- Usually people use **deterministic head rules** (Magerman, 1995).

Lexicalization

- Every nonterminal node is annotated with a word from its yield; such that
    - lex(n) = lex(head(n))

Lexicalization

- Every nonterminal node is annotated with a word from its yield; such that
    - lex(n) = lex(head(n))

- What might this allow?
- What might we worry about?

Algorithms

- These "decorations" affect our parser's runtime.
  - Why?
  - Any ideas about how to get around this?

Some Famous Parsers

Collins Model 1 (1997)

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

- Huge number of rules!

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

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

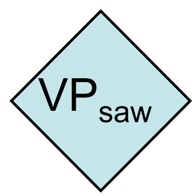
$VP_{saw} \rightarrow \underline{V}_{saw}\ NP_{woman}\ PP_{through}$

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

- Key: factor probabilities within rule.

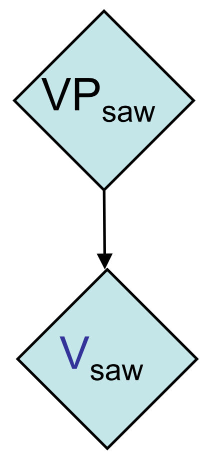
Collins Model 1 (1997)

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



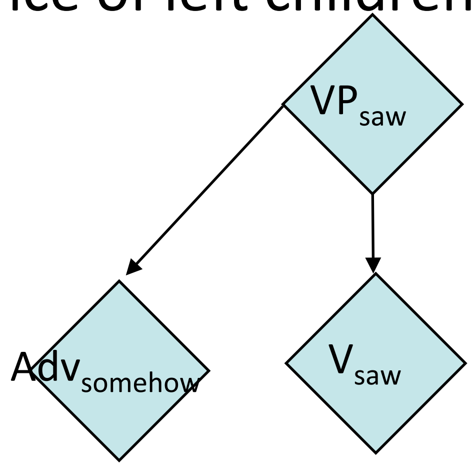
Collins Model 1 (1997)

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



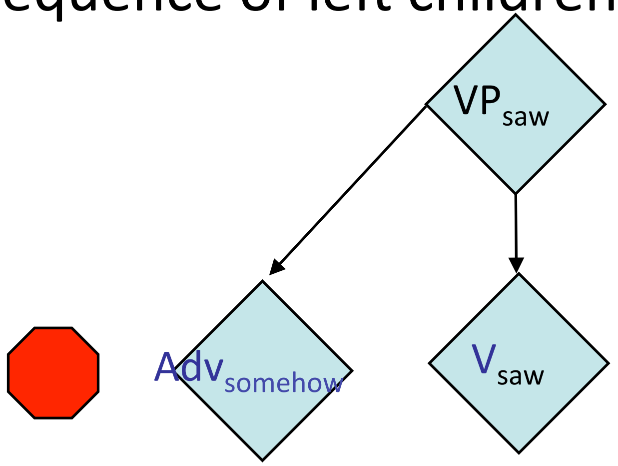
Collins Model 1 (1997)

- 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.



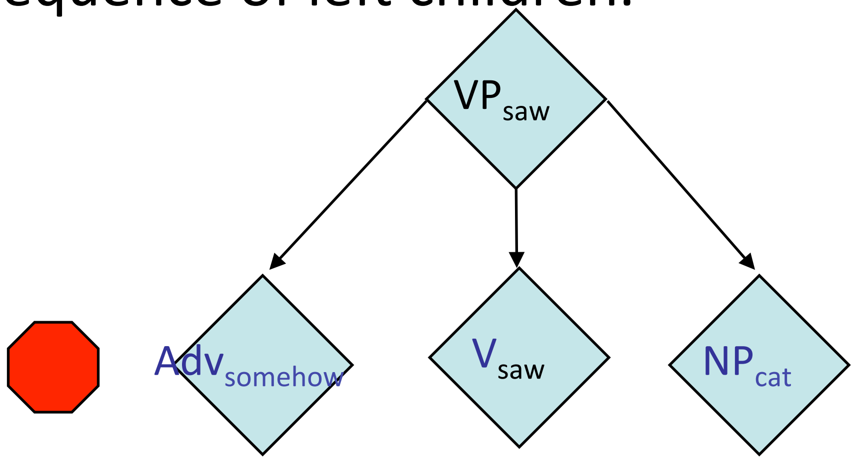
Collins Model 1 (1997)

- 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.



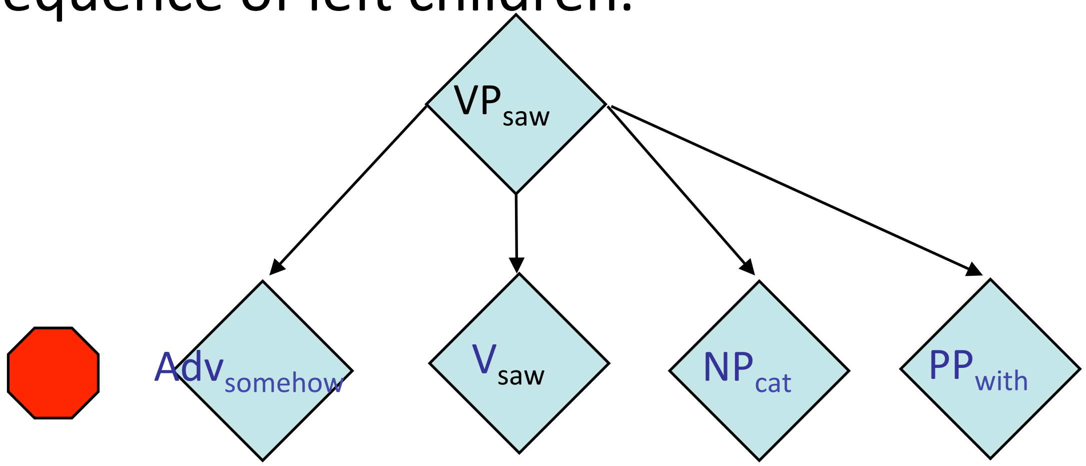
Collins Model 1 (1997)

- 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.



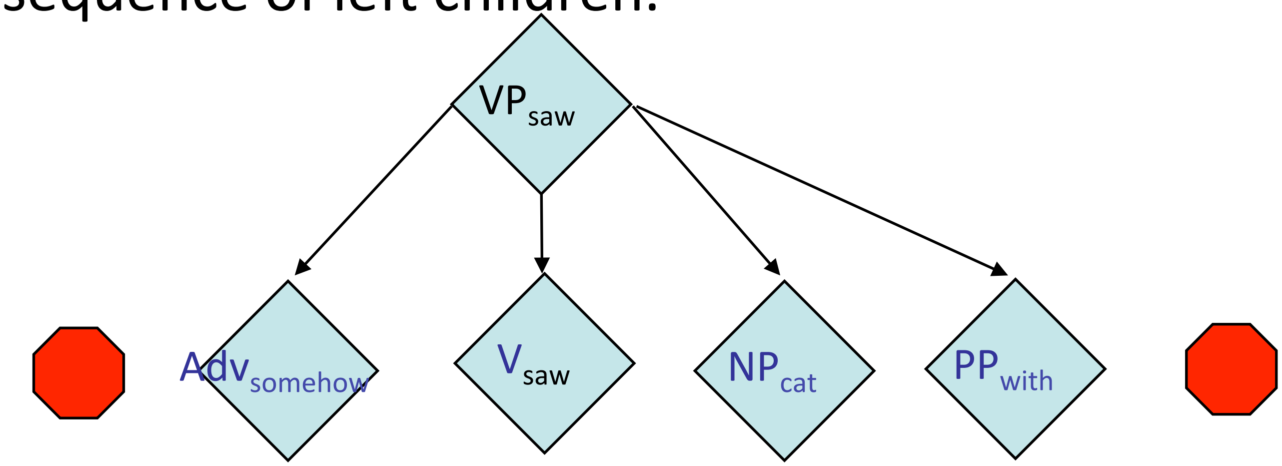
Collins Model 1 (1997)

- 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.



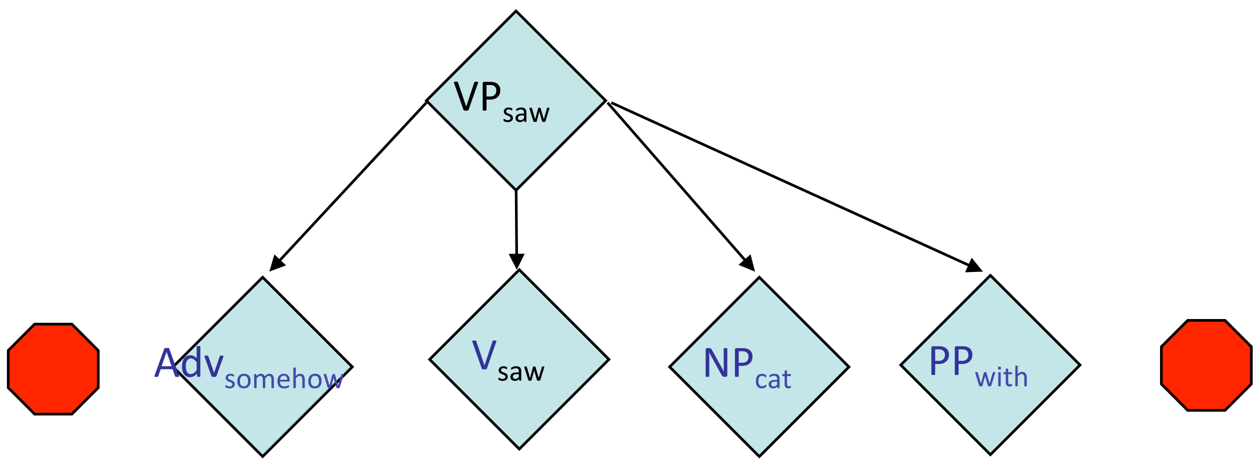
Collins Model 1 (1997)

- 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.



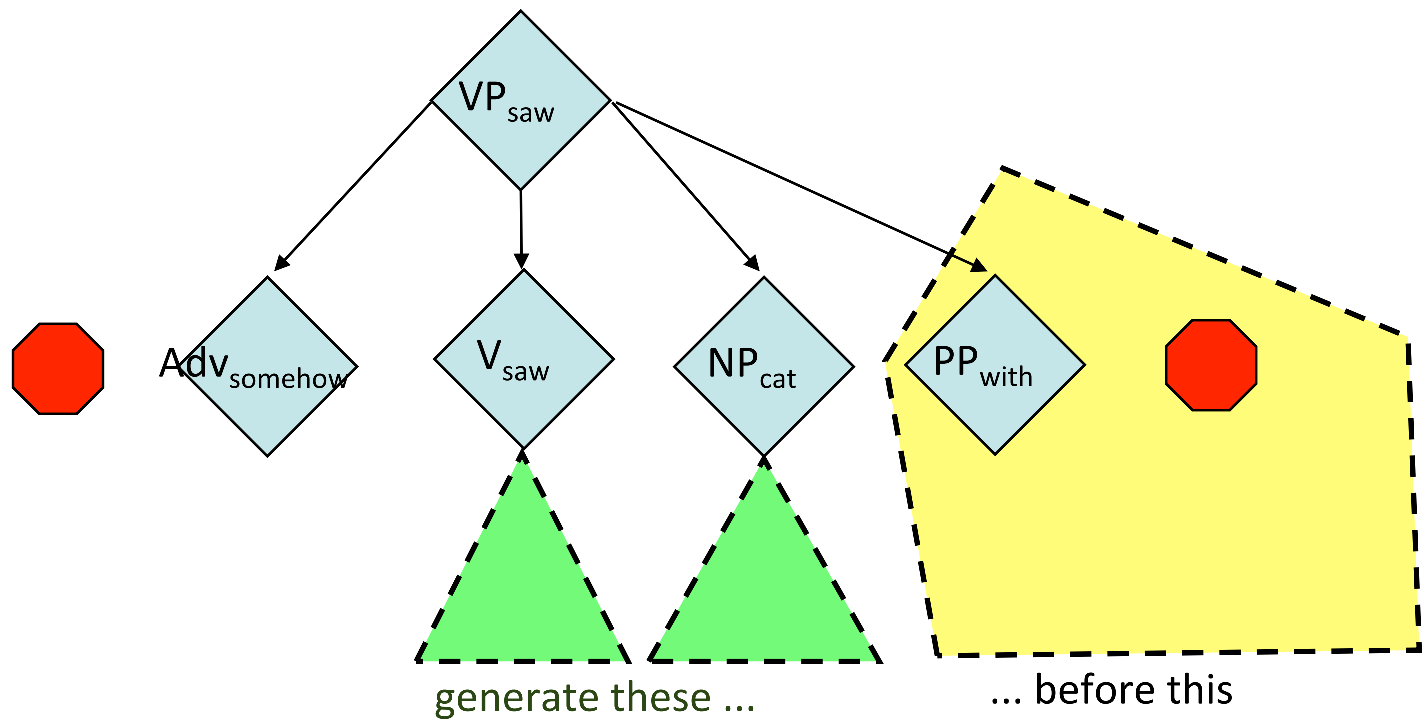
Collins Model 1 (1997)

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



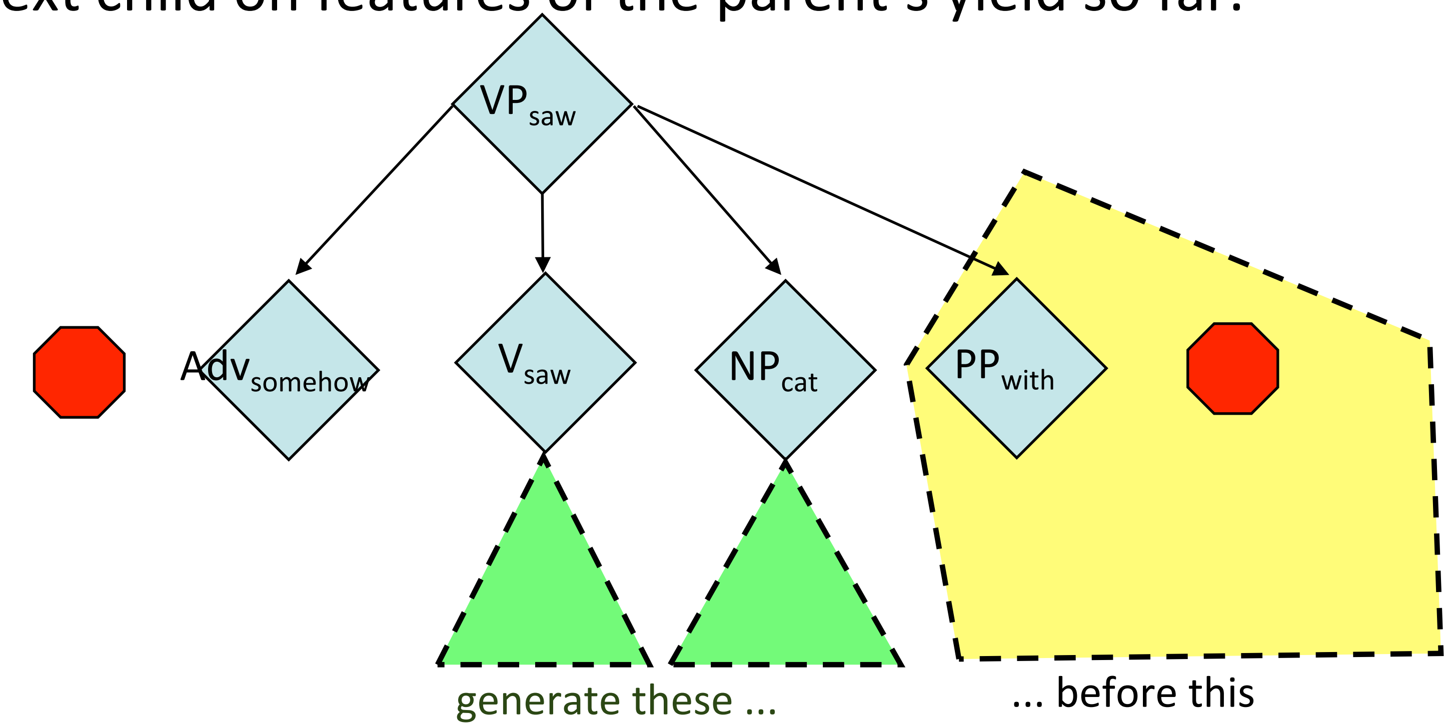
Collins Model 1 (1997)

- 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 so far.



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 so far.

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

$$\begin{aligned}
& p(L_n, u_n, L_{n-1}, u_{n-1}, ..., L_1, u_1, H, w, R_1, v_1, R_2, v_2, ..., R_m, v_m \mid P, w) \\
= \;& p(H \mid P, w) \\
& \cdot \prod_{i=1}^{n} p(L_i, u_i \mid P, w, H, \text{left}, \Delta_i) \\
& \cdot p(\text{stop} \mid P, w, H, \text{left}, \Delta_{n+1}) \\
& \cdot \prod_{i=1}^{m} p(R_i, v_i \mid P, w, H, \text{right}, \Delta'_i) \\
& \cdot p(\text{stop} \mid P, w, H, \text{right}, \Delta'_{n+1})
\end{aligned}$$

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)

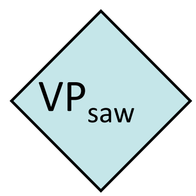
Practical Notes

- Collins parser is freely available
- Dan 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

Charniak (1997) - in brief

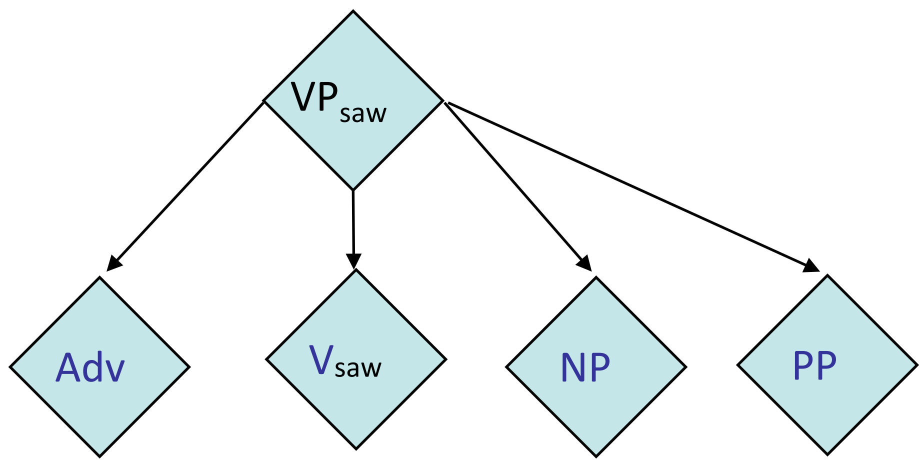
- Generally similar to Collins
- Key differences:
  - Used an additional 30 million words of unparsed text in training
  - Rules not fully markovized: pick full nonterminal sequence, then lexicalize each child independently

Charniak (1997) - in brief



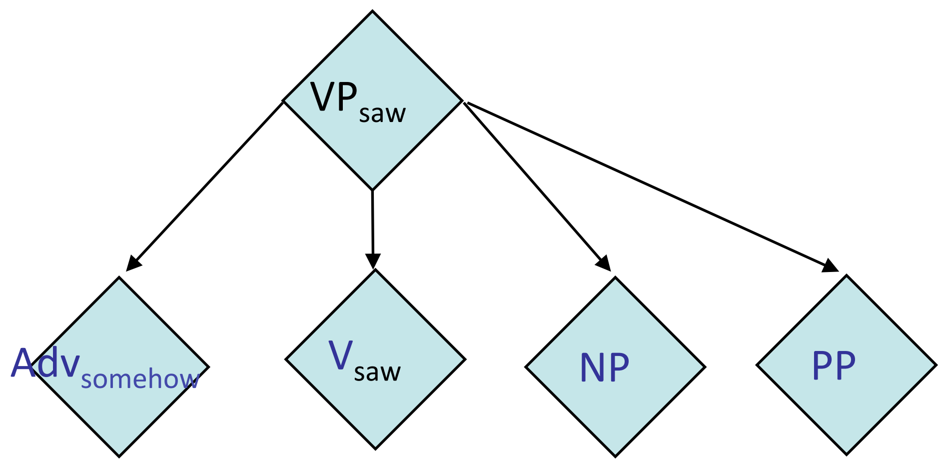
Charniak (1997) - in brief

$VP_{saw} \rightarrow$ Adv $\underline{V}$ NP PP



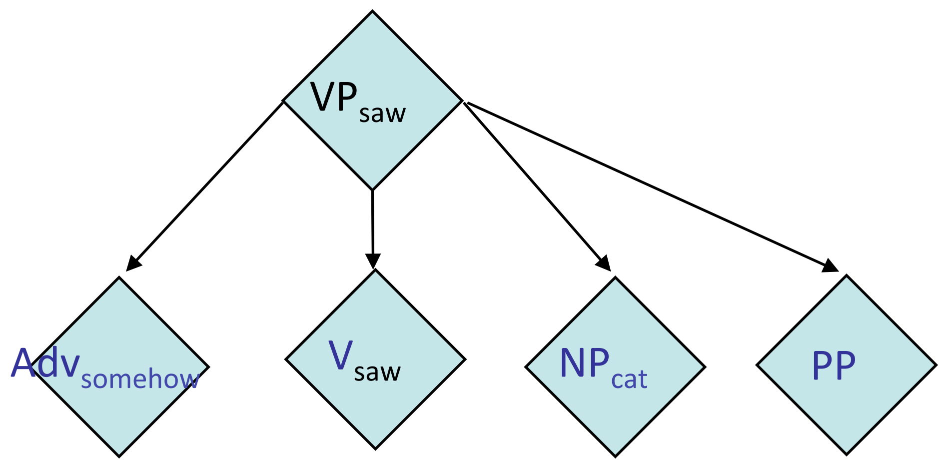
Charniak (1997) - in brief

$p(\text{somehow} \mid VP_{saw}, Adv)$



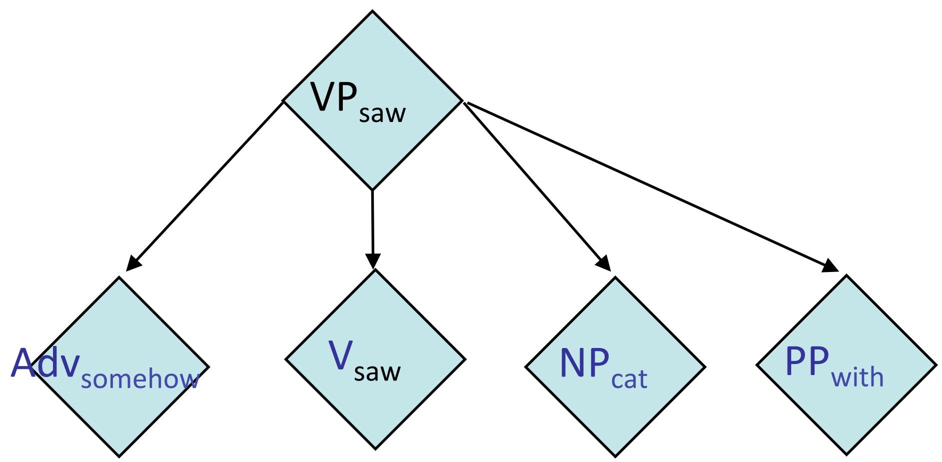
Charniak (1997) - in brief

p(cat | VP_{saw}, NP)



Charniak (1997) - in brief

$p(\text{with} \mid VP_{saw}, PP)$



Charniak (2000)

- Uses grandparents (Johnson ’98 transformation)
- Markovized children (like Collins)
- Bizarre probability model:
  - Smoothed estimates at many backoff levels
  - Multiply them together
  - “Maximum entropy inspired”
  - Kind of a product of experts (untrained)

Comparison

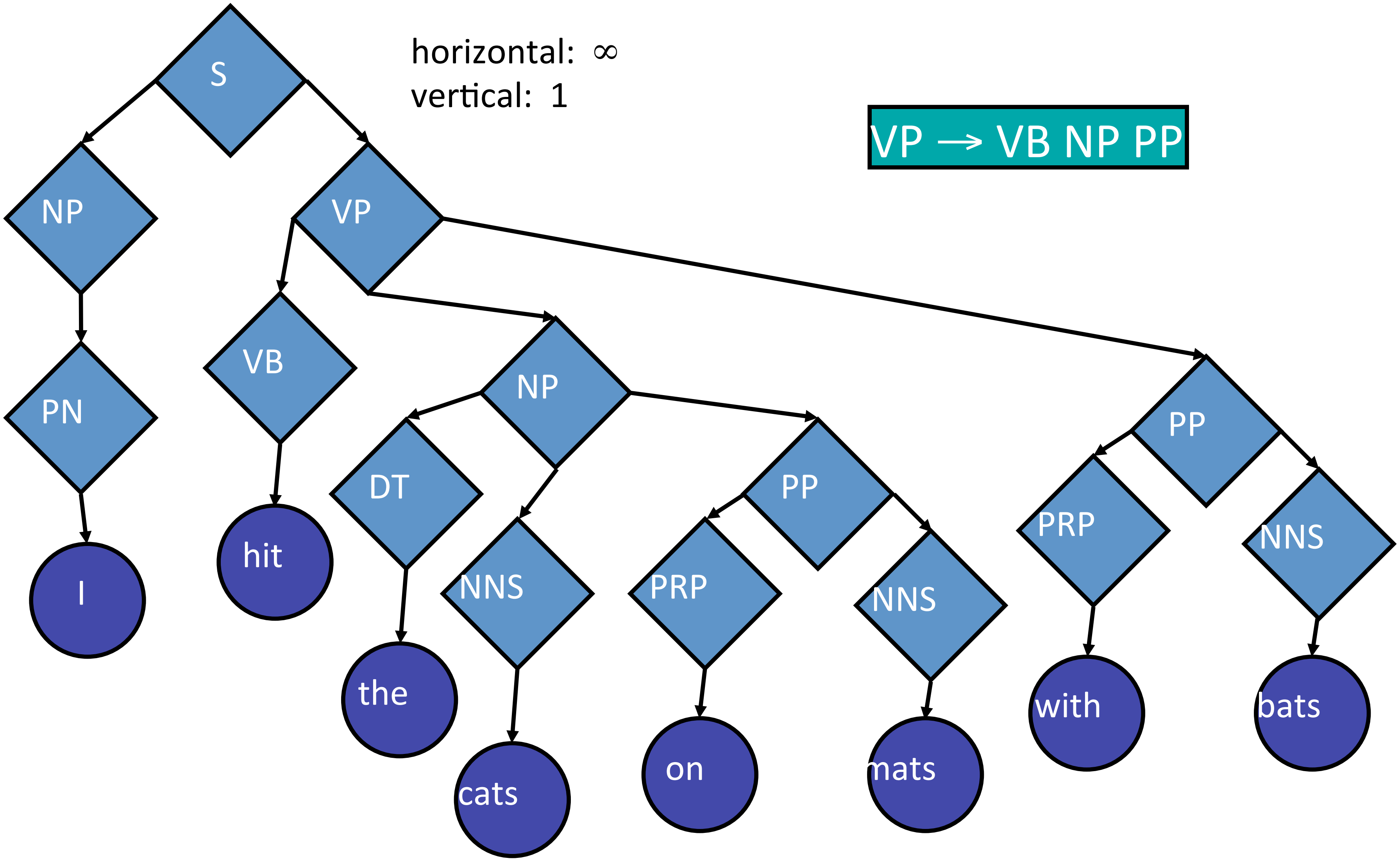
		labeled recall	labeled precision	average crossing brackets
Collins	Model 1	87.5	87.7	1.09
	Model 2	88.1	88.3	1.06
	Model 3	88.0	88.3	1.05
Charniak	1997	86.7	86.6	1.20
	2000	89.6	89.5	0.88

Klein and Manning (2003)

- By now, lexicalization was kind of controversial
  - So many probabilities, such expensive parsing: is it necessary?
- Goal: reasonable unlexicalized baseline
  - What tree transformations make sense?
  - Markovization (what order?)
  - Add all kinds of information to each node in the treebank
- Performance close to Collins model, much better than earlier unlexicalized models

Markovization

horizontal: ∞
vertical: 1



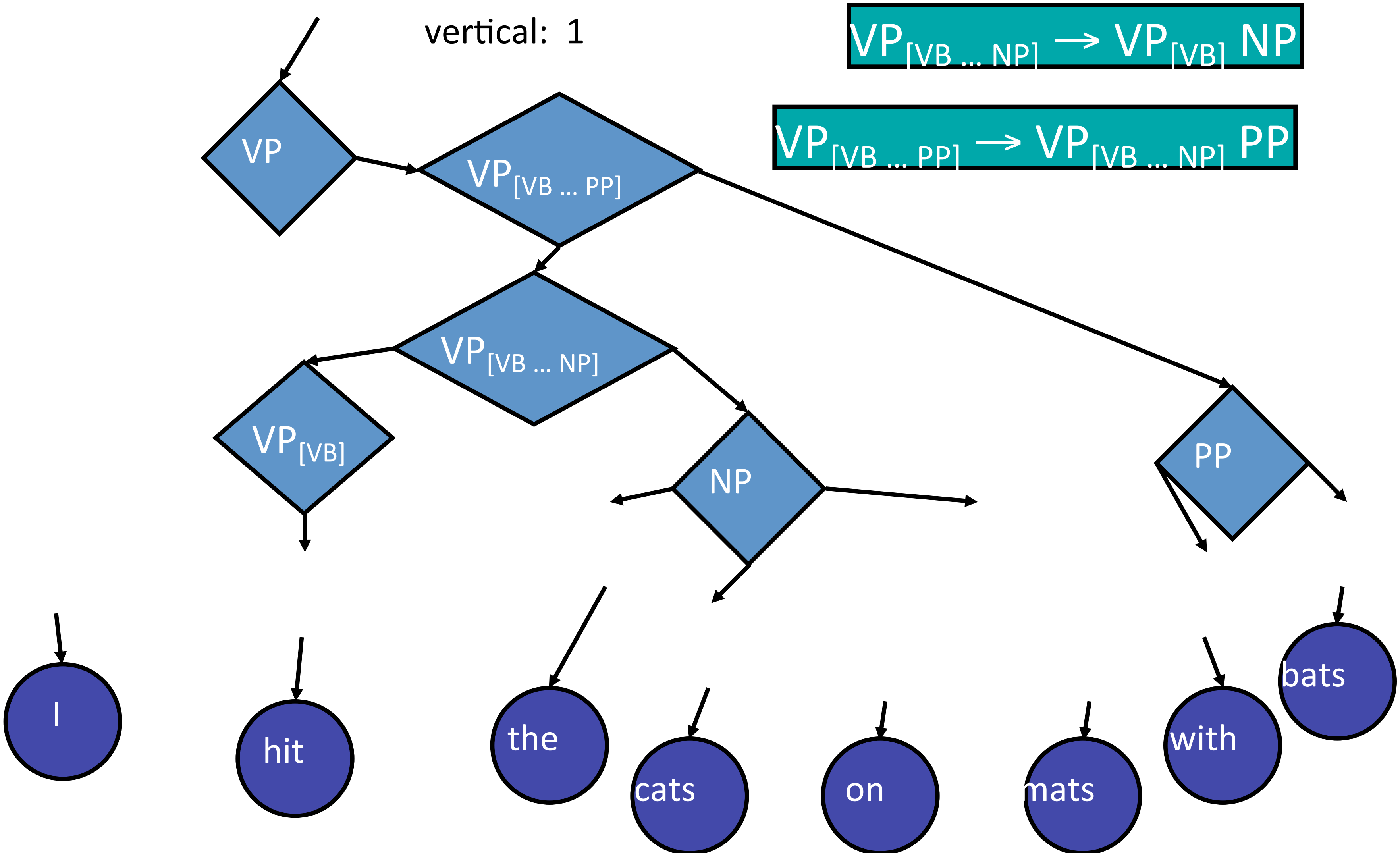
Markovization

$VP_{[VB]} \rightarrow VB$

$VP_{[VB\ \ldots\ NP]} \rightarrow VP_{[VB]}\ NP$

$VP_{[VB\ \ldots\ PP]} \rightarrow VP_{[VB\ \ldots\ NP]}\ PP$

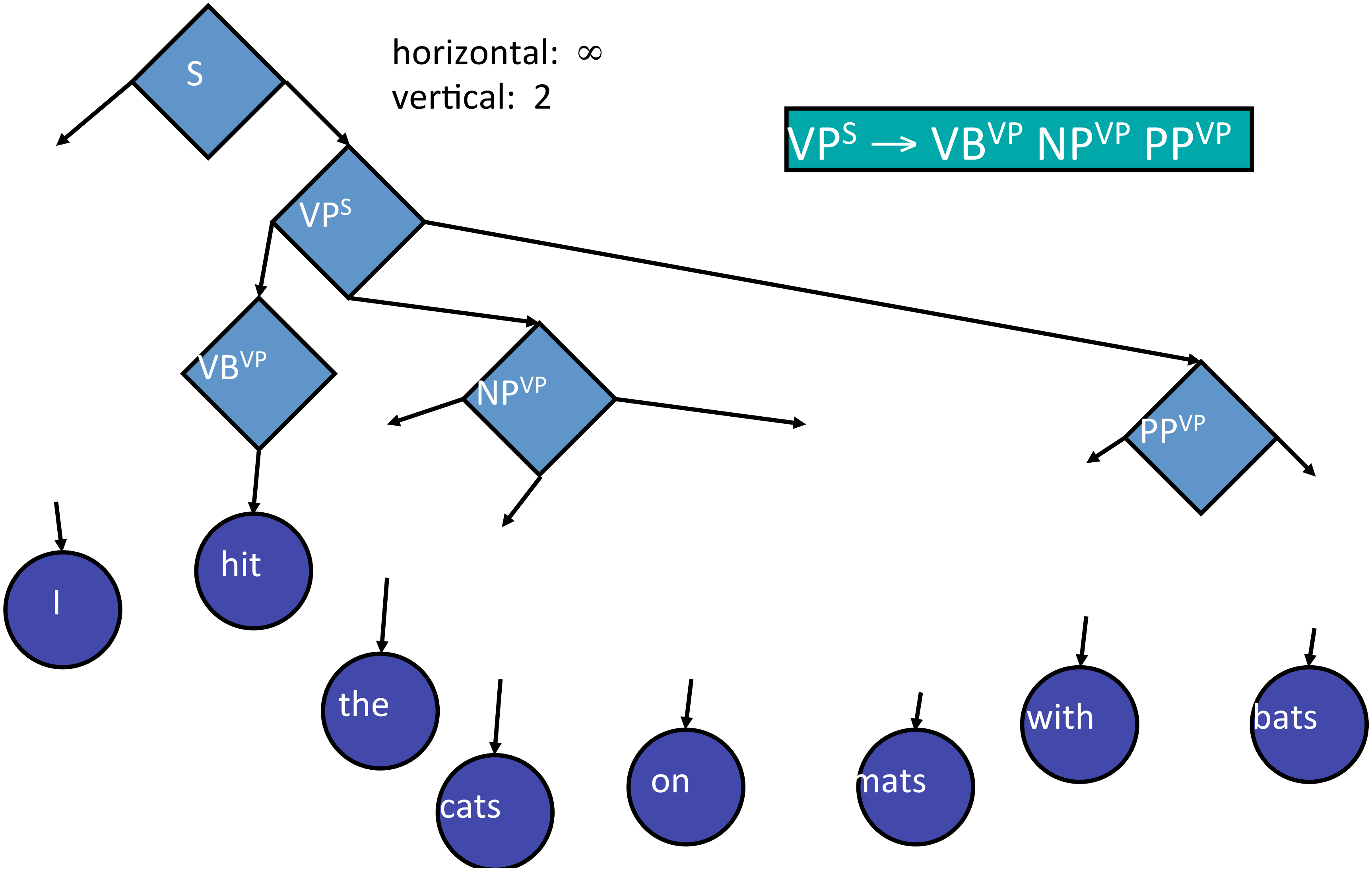
horizontal: 1
vertical: 1



Markovization

horizontal: ∞
vertical: 2

$VP^{S} \rightarrow VB^{VP}\ NP^{VP}\ PP^{VP}$



Markovization

- More vertical Markovization is better
  – Consistent with Johnson (1998)
- Horizontal 1 or 2 beats 0 or ∞
- Used (2, 2), but if sparse "back off" to 1

Other Tree Decorations

- Mark nodes with only 1 child as UNARY
- Mark DTs (determiners), RBs (adverbs) when they are only children
- Annotate POS tags with their parents
- Split IN (prepositions; 6 ways), AUX, CC, %
- NPs: temporal, possessive, base
- VPs annotated with head tag (finite vs. others)
- DOMINATES-V
- RIGHT-RECURSIVE NP

Comparison

		labeled recall	labeled precision	average crossing brackets
Collins	Model 1	87.5	87.7	1.09
	Model 2	88.1	88.3	1.06
	Model 3	88.0	88.3	1.05
Charniak	1997	86.7	86.6	1.20
	2000	89.6	89.5	0.88
K&M	2003	86.3	85.1	1.31