Stochastic Context Free Grammars for RNA Structure Modeling

BMI/CS 776

www.biostat.wisc.edu/bmi776/

Spring 2019

Colin Dewey

colin.dewey@wisc.edu

Goals for Lecture

Key concepts

- transformational grammars
- the Chomsky hierarchy
- context free grammars
- stochastic context free grammars
- parsing ambiguity
- the Inside and Outside algorithms
- parameter learning via the Inside-Outside algorithm

Modeling RNA with Stochastic Context Free Grammars

- Consider tRNA genes
  - 274 in yeast genome, ~1500 in human genome
  - get transcribed, like protein-coding genes
  - don’t get translated, therefore base statistics much different than protein-coding genes
  - but secondary structure is conserved
- To recognize new tRNA genes, model known ones using stochastic context free grammars [Eddy & Durbin, 1994; Sakakibara et al. 1994]
- But what is a grammar?

Transformational Grammars

- A transformational grammar characterizes a set of legal strings
- The grammar consists of
  - a set of abstract *nonterminal* symbols

    $$\{s,\ c_1,\ c_2,\ c_3,\ c_4\}$$

  - a set of *terminal* symbols (those that actually appear in strings)

    $$\{\mathrm{A},\ \mathrm{C},\ \mathrm{G},\ \mathrm{U}\}$$

  - a set of *productions*

$$s \rightarrow c_1 \qquad c_1 \rightarrow \mathrm{U}c_2 \qquad c_2 \rightarrow \mathrm{A}c_3 \qquad c_3 \rightarrow \mathrm{A} \qquad c_4 \rightarrow \mathrm{A}$$

$$c_2 \rightarrow \mathrm{G}c_4 \qquad c_3 \rightarrow \mathrm{G}$$

A Grammar for Stop Codons

$$s \rightarrow c_1 \quad c_1 \rightarrow \mathrm{U}c_2 \quad c_2 \rightarrow \mathrm{A}c_3 \quad c_3 \rightarrow \mathrm{A} \quad c_4 \rightarrow \mathrm{A}$$

$$c_2 \rightarrow \mathrm{G}c_4 \quad c_3 \rightarrow \mathrm{G}$$

- This grammar can generate the 3 stop codons: UAA, UAG, UGA

- With a grammar we can ask questions like
  - what strings are derivable from the grammar?
  - can a particular string be derived from the grammar?
  - what sequence of productions can be used to derive a particular string from a given grammar?

The Derivation for UAG

$$s \to c_1 \qquad c_1 \to \mathrm{U}c_2 \qquad c_2 \to \mathrm{A}c_3 \qquad c_3 \to \mathrm{A} \qquad c_4 \to \mathrm{A}$$

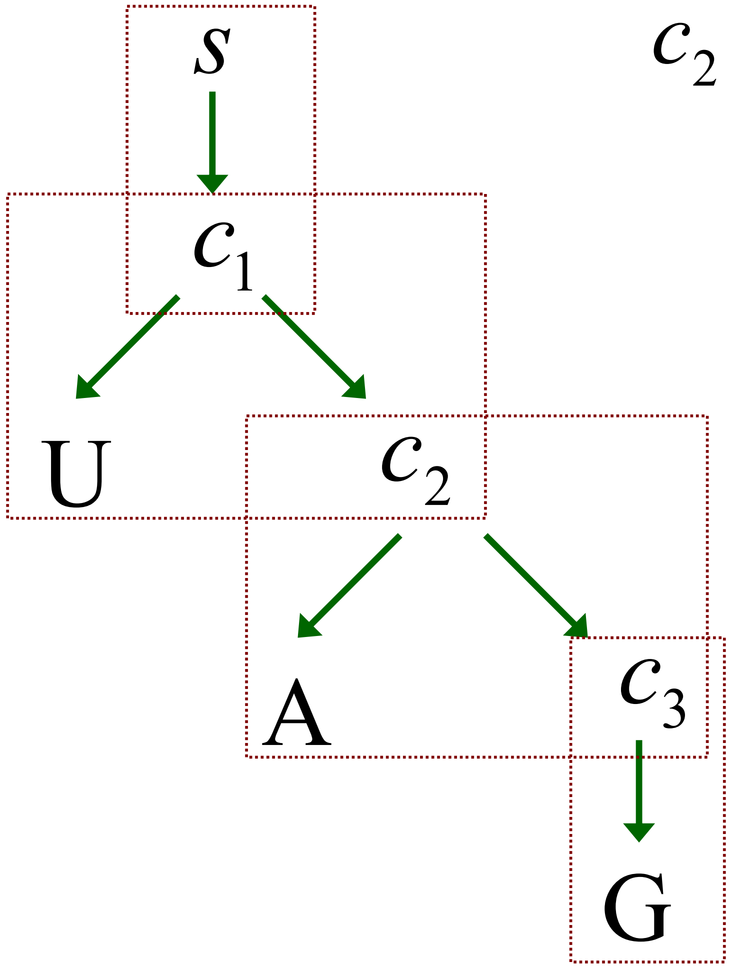
$$c_2 \to \mathrm{G}c_4 \qquad c_3 \to \mathrm{G}$$

$$s \Rightarrow c_1 \Rightarrow \mathrm{U}c_2 \Rightarrow \mathrm{UA}c_3 \Rightarrow \mathrm{UAG}$$

The Parse Tree for UAG

$s \rightarrow c_1$ $\quad c_1 \rightarrow \mathrm{U}c_2$ $\quad c_2 \rightarrow \mathrm{A}c_3$ $\quad c_3 \rightarrow \mathrm{A}$ $\quad c_4 \rightarrow \mathrm{A}$

$c_2 \rightarrow \mathrm{G}c_4$ $\quad c_3 \rightarrow \mathrm{G}$



Some Shorthand

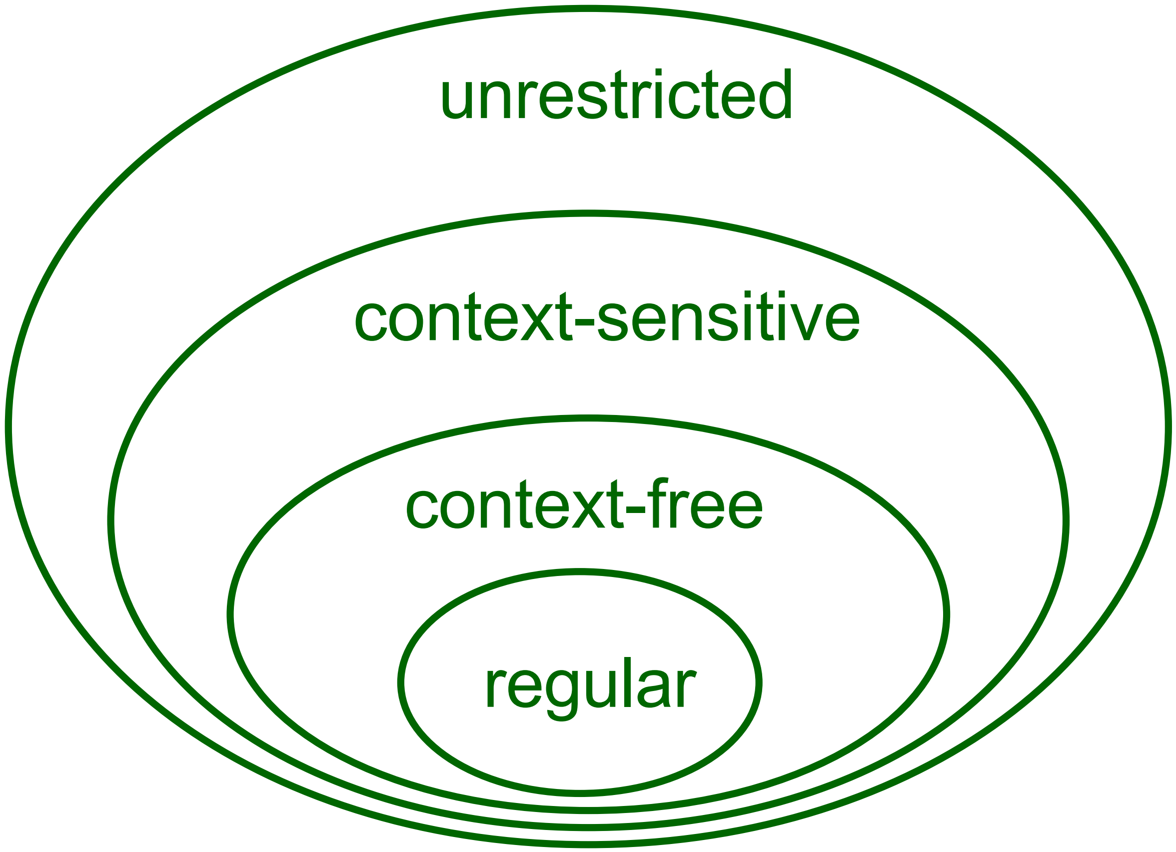
$$c_2 \to \mathrm{A}c_3$$

$$c_2 \to \mathrm{G}c_4$$

$$\Longleftrightarrow$$

$$c_2 \to \mathrm{A}c_3 \mid \mathrm{G}c_4$$

The Chomsky Hierarchy



- A hierarchy of grammars defined by restrictions on productions

The Chomsky Hierarchy

- Regular grammars $u \rightarrow \mathrm{X}v$ $u \rightarrow \mathrm{X}$
- Context-free grammars $u \rightarrow \beta$
- Context-sensitive grammars $\alpha_1 u \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$
- Unrestricted grammars $\alpha_1 u \alpha_2 \rightarrow \alpha_3$

u, v are nonterminals

X is a terminal

$\alpha_1, \alpha_2, \alpha_3$ are any sequence of terminals/nonterminals

β is any non-null sequence of terminals/nonterminals

CFGs and RNA

- Context free grammars are well suited to modeling RNA secondary structure because they can represent base pairing preferences
- A grammar for a 3-base stem with a loop of either GAAA or GCAA

$$s \rightarrow \mathrm{A}w_1\mathrm{U} \mid \mathrm{C}w_1\mathrm{G} \mid \mathrm{G}w_1\mathrm{C} \mid \mathrm{U}w_1\mathrm{A}$$

$$w_1 \rightarrow \mathrm{A}w_2\mathrm{U} \mid \mathrm{C}w_2\mathrm{G} \mid \mathrm{G}w_2\mathrm{C} \mid \mathrm{U}w_2\mathrm{A}$$

$$w_2 \rightarrow \mathrm{A}w_3\mathrm{U} \mid \mathrm{C}w_3\mathrm{G} \mid \mathrm{G}w_3\mathrm{C} \mid \mathrm{U}w_3\mathrm{A}$$

$$w_3 \rightarrow \mathrm{GAAA} \mid \mathrm{GCAA}$$

CFGs and RNA

a. Productions

$$
P = \{ \begin{array}{llll}
S_0 \rightarrow S_1, & & S_7 \rightarrow \mathrm{G}\ S_8, & \\
S_1 \rightarrow \mathrm{C}\ S_2\ \mathrm{G}, & & S_8 \rightarrow \mathrm{G}, & \\
S_1 \rightarrow \mathrm{A}\ S_2\ \mathrm{U}, & & S_8 \rightarrow \mathrm{U}, & \\
S_2 \rightarrow \mathrm{A}\ S_3\ \mathrm{U}, & & S_9 \rightarrow \mathrm{A}\ S_{10}\ \mathrm{U}, & \\
S_3 \rightarrow S_4\ \ S_9, & & S_{10} \rightarrow \mathrm{C}\ S_{10}\ \mathrm{G}, & \\
S_4 \rightarrow \mathrm{U}\ S_5\ \mathrm{A}, & & S_{10} \rightarrow \mathrm{G}\ S_{11}\ \mathrm{C}, & \\
S_5 \rightarrow \mathrm{C}\ S_6\ \mathrm{G}, & & S_{11} \rightarrow \mathrm{A}\ S_{12}\ \mathrm{U}, & \\
S_6 \rightarrow \mathrm{A}\ S_7, & & S_{12} \rightarrow \mathrm{U}\ S_{13}, & \\
S_7 \rightarrow \mathrm{U}\ S_7, & & S_{13} \rightarrow \mathrm{C} & \}
\end{array}
$$

b. Derivation

$$
\begin{array}{ll}
S_0 & \Rightarrow S_1 \Rightarrow \mathrm{C}S_2\mathrm{G} \Rightarrow \mathrm{CA}S_3\mathrm{UG} \Rightarrow \mathrm{CA}S_4S_9\mathrm{UG} \\
 & \Rightarrow \mathrm{CAU}S_5\mathrm{A}S_9\mathrm{UG} \Rightarrow \mathrm{CAUC}S_6\mathrm{GA}S_9\mathrm{UG} \\
 & \Rightarrow \mathrm{CAUCA}S_7\mathrm{GA}S_9\mathrm{UG} \Rightarrow \mathrm{CAUCAG}S_8\mathrm{GA}S_9\mathrm{UG} \\
 & \Rightarrow \mathrm{CAUCAGGGA}S_9\mathrm{UG} \Rightarrow \mathrm{CAUCAGGGAA}S_{10}\mathrm{UUG} \\
 & \Rightarrow \mathrm{CAUCAGGGAAG}S_{11}\mathrm{CUUG} \\
 & \Rightarrow \mathrm{CAUCAGGGAAGA}S_{12}\mathrm{UCUUG} \\
 & \Rightarrow \mathrm{CAUCAGGGAAGAU}S_{13}\mathrm{UCUUG} \\
 & \Rightarrow \mathrm{CAUCAGGGAAGAUCUCUUG.}
\end{array}
$$

c. Parse tree **d. Secondary Structure**

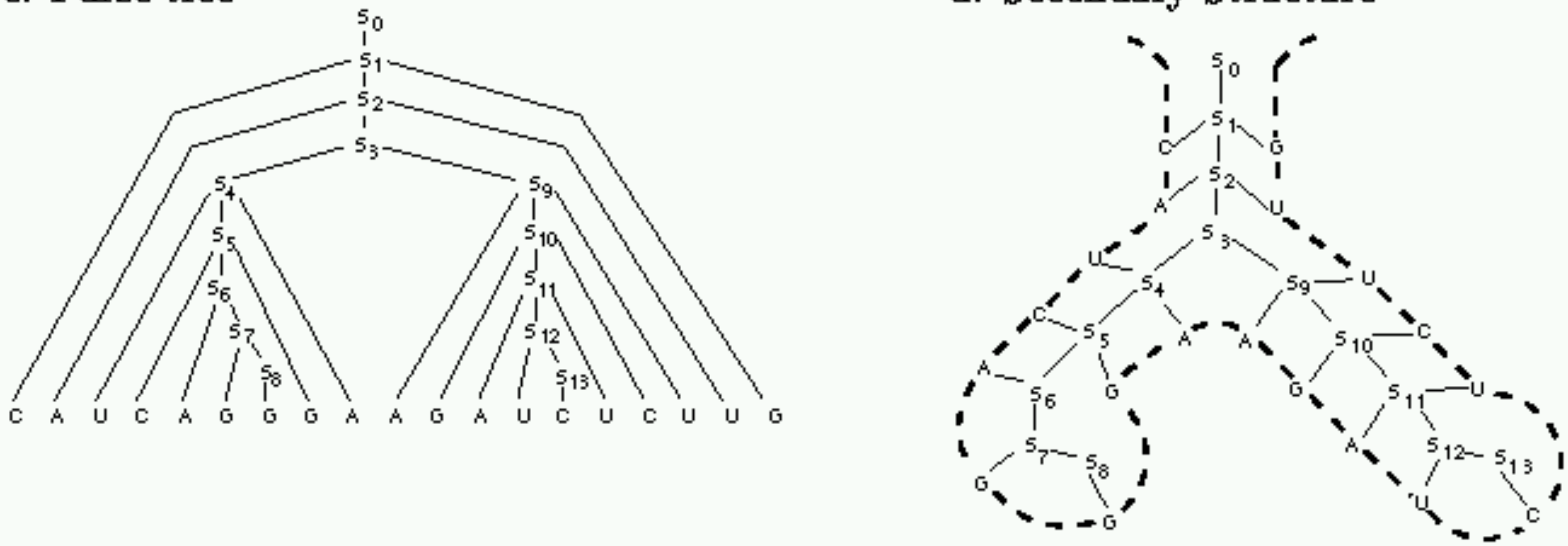
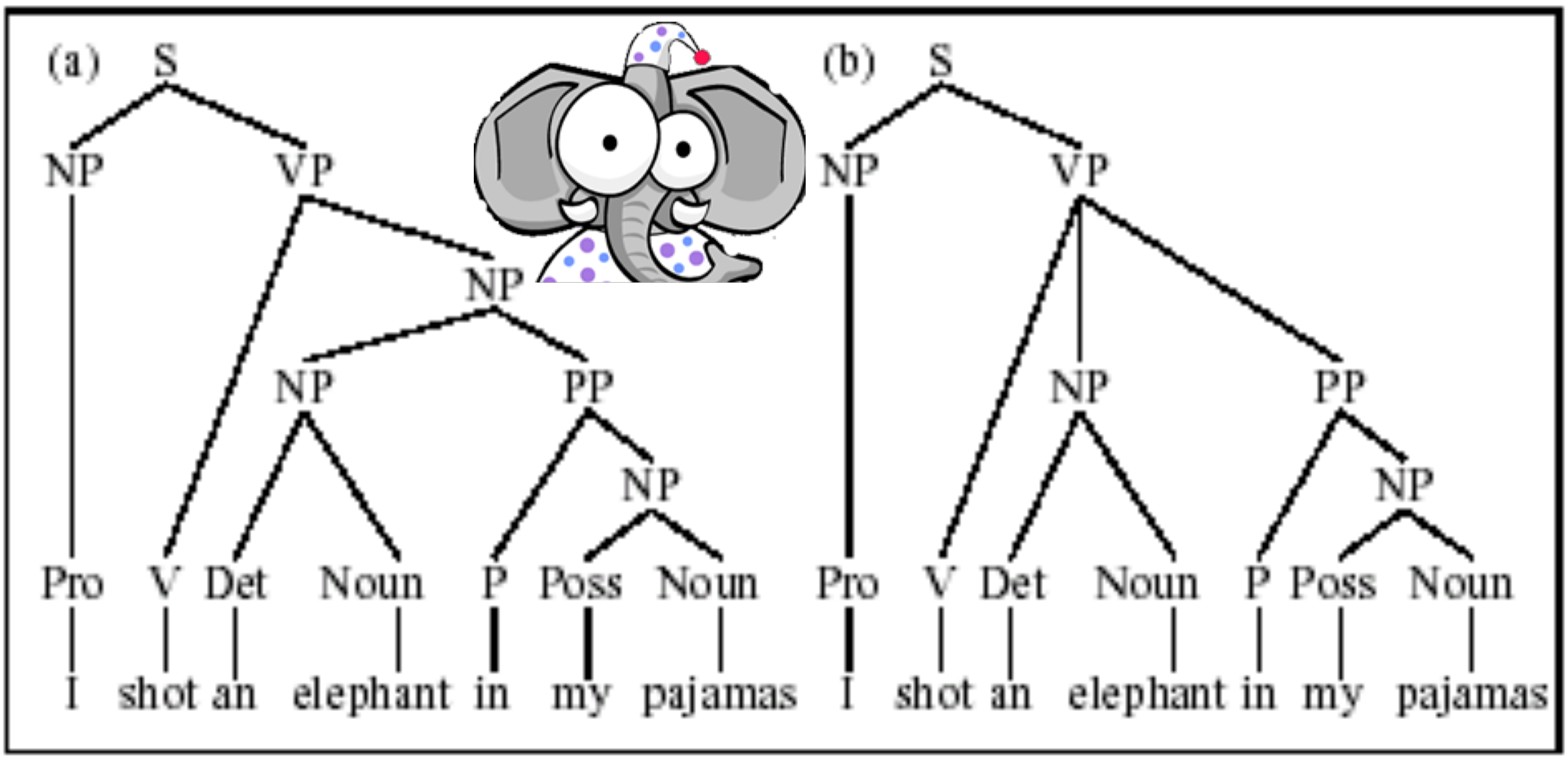


Figure from: Sakakibara et al. *Nucleic Acids Research*, 1994

Ambiguity in Parsing

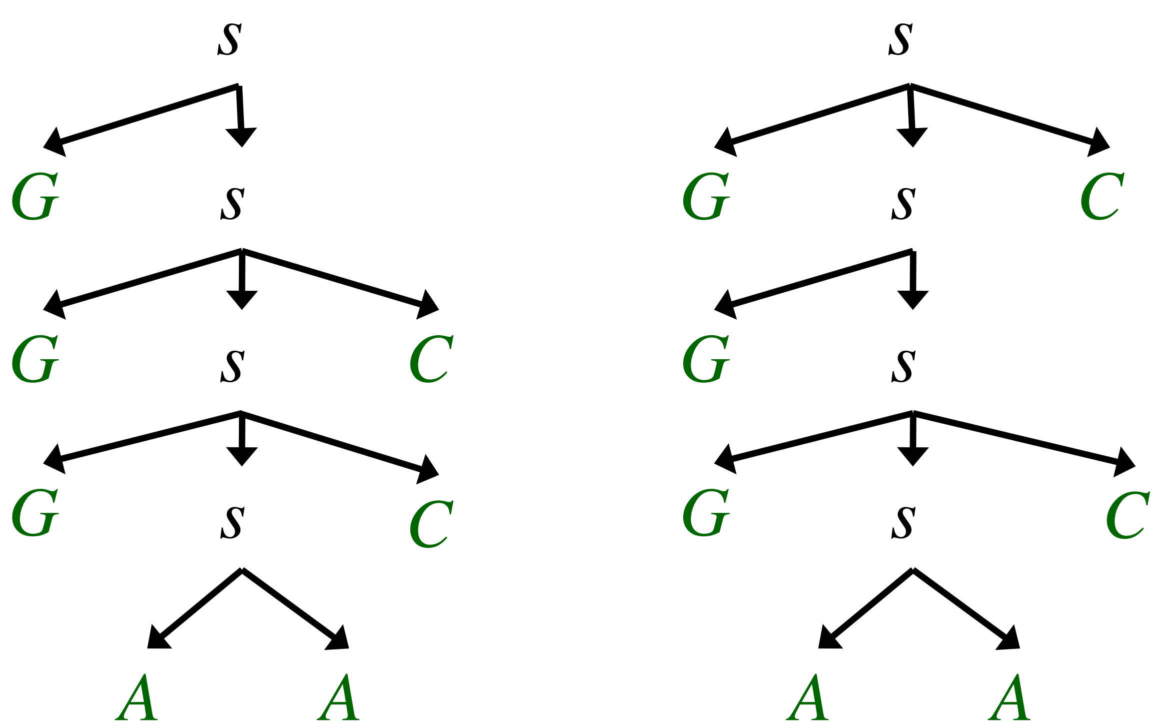
"I shot an elephant in my pajamas. How he got in my pajamas, I'll never know." – Groucho Marx



An Ambiguous RNA Grammar

$$s \rightarrow G\ s\ C$$
$$s \rightarrow G\ s$$
$$s \rightarrow A\ A$$

- With this grammar, there are 3 parses for the string *GGGAACC*



A Probabilistic Version of the Stop Codon Grammar

$$s \xrightarrow{1.0} c_1 \qquad c_1 \xrightarrow{1.0} \mathrm{U}c_2 \qquad c_2 \xrightarrow{0.7} \mathrm{A}c_3 \qquad c_3 \xrightarrow{0.2} \mathrm{A} \qquad c_4 \xrightarrow{1.0} \mathrm{A}$$

$$c_2 \xrightarrow{0.3} \mathrm{G}c_4 \qquad c_3 \xrightarrow{0.8} \mathrm{G}$$

- Each production has an associated probability
- Probabilities for productions with the same left-hand side sum to 1
- This *regular* grammar has a corresponding Markov chain model

Stochastic Context Free Grammars

(a.k.a. Probabilistic Context Free Grammars)

$$s \to \overset{0.25}{\mathrm{A}w_1\mathrm{U}} \mid \overset{0.25}{\mathrm{C}w_1\mathrm{G}} \mid \overset{0.25}{\mathrm{G}w_1\mathrm{C}} \mid \overset{0.25}{\mathrm{U}w_1\mathrm{A}}$$

$$w_1 \to \overset{0.1}{\mathrm{A}w_2\mathrm{U}} \mid \overset{0.4}{\mathrm{C}w_2\mathrm{G}} \mid \overset{0.4}{\mathrm{G}w_2\mathrm{C}} \mid \overset{0.1}{\mathrm{U}w_2\mathrm{A}}$$

$$w_2 \to \overset{0.25}{\mathrm{A}w_3\mathrm{U}} \mid \overset{0.25}{\mathrm{C}w_3\mathrm{G}} \mid \overset{0.25}{\mathrm{G}w_3\mathrm{C}} \mid \overset{0.25}{\mathrm{U}w_3\mathrm{A}}$$

$$w_3 \to \overset{0.8}{\mathrm{GAAA}} \mid \overset{0.2}{\mathrm{GCAA}}$$

Stochastic Grammars?

…the notion "probability of a sentence" is an entirely useless one, under any known interpretation of this term.

— Noam Chomsky
(famed linguist)

Every time I fire a linguist, the performance of the recognizer improves.

— Fred Jelinek
(former head of IBM speech recognition group)

Credit for pairing these quotes goes to Dan Jurafsky and James Martin, *Speech and Language Processing*

Three Key Questions

- How likely is a given sequence?

  the Inside algorithm
- What is the most probable parse for a given sequence?

  the Cocke-Younger-Kasami (CYK) algorithm
- How can we learn the SCFG parameters given a grammar and a set of sequences?

  the Inside-Outside algorithm

Chomsky Normal Form

- It is convenient to assume that our grammar is in *Chomsky Normal Form*; i.e. all productions are of the form:

$$v \to yz$$ right hand side consists of two nonterminals

$$v \to A$$ right hand side consists of a single terminal

- Any CFG can be put into Chomsky Normal Form

Converting a Grammar to CNF

$$s \to G\, s\, C$$
$$s \to G\, s$$
$$s \to A\, A$$

$$\Longrightarrow$$

$$s \to b_G\ p$$
$$p \to s\, b_C$$
$$s \to b_G\ s$$
$$s \to b_A b_A$$
$$b_G \to G$$
$$b_C \to C$$
$$b_A \to A$$

Parameter Notation

- For productions of the form $v \rightarrow yz$, we'll denote the associated probability parameters

$$t_v(y,z) \qquad \text{transition}$$

- For productions of the form $v \rightarrow A$, we'll denote the associated probability parameters

$$e_v(A) \qquad \text{emission}$$

Determining the Likelihood of a Sequence: The Inside Algorithm

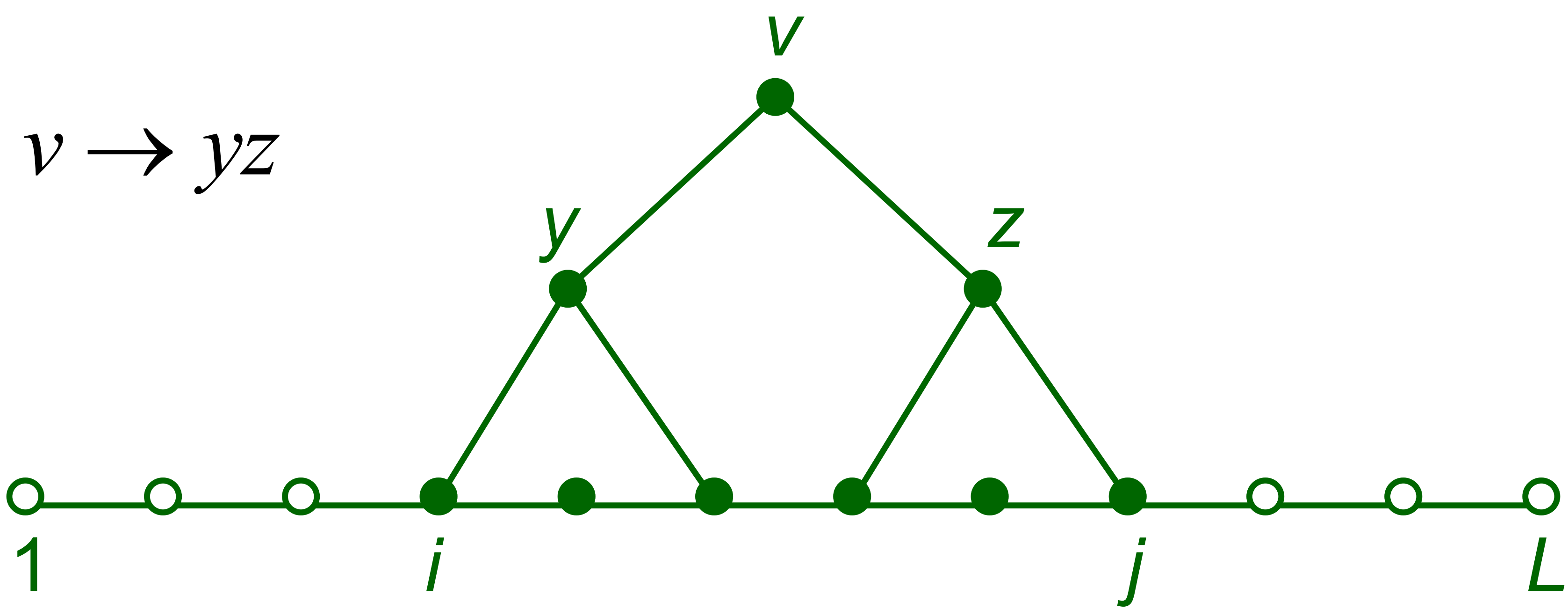
- Dynamic programming method, analogous to the Forward algorithm
- Involves filling in a 3D matrix

$$\alpha(i, j, v)$$

representing the probability of all parse subtrees rooted at nonterminal v for the subsequence from i to j

Determining the Likelihood of a Sequence: The Inside Algorithm

$v \rightarrow yz$



- $\alpha(i, j, v)$: the probability of all parse subtrees rooted at nonterminal v for the subsequence from i to j

Inside Calculation Example

$s \rightarrow b_G \; p$

$p \rightarrow s \, b_C$

$s \rightarrow b_G \; s$

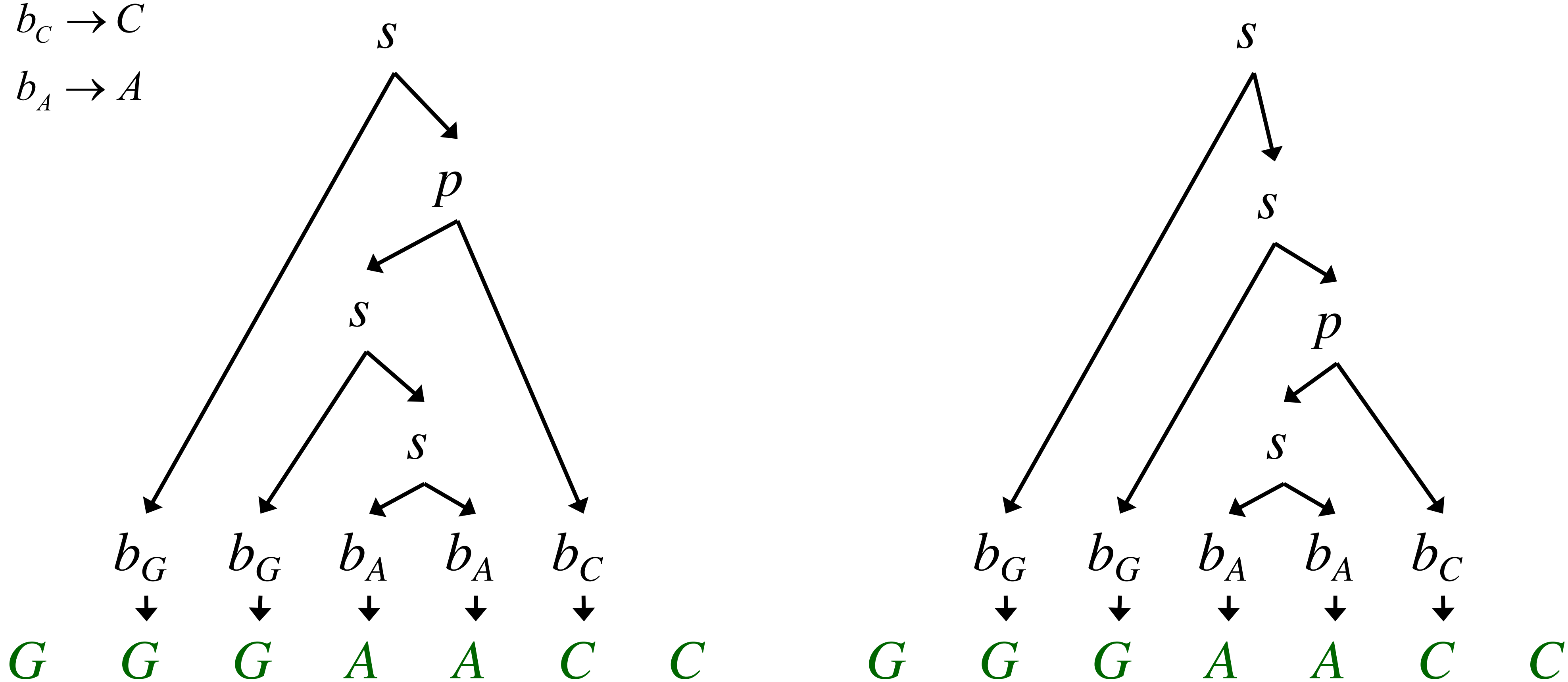
$s \rightarrow b_A \, b_A$

$b_G \rightarrow G$

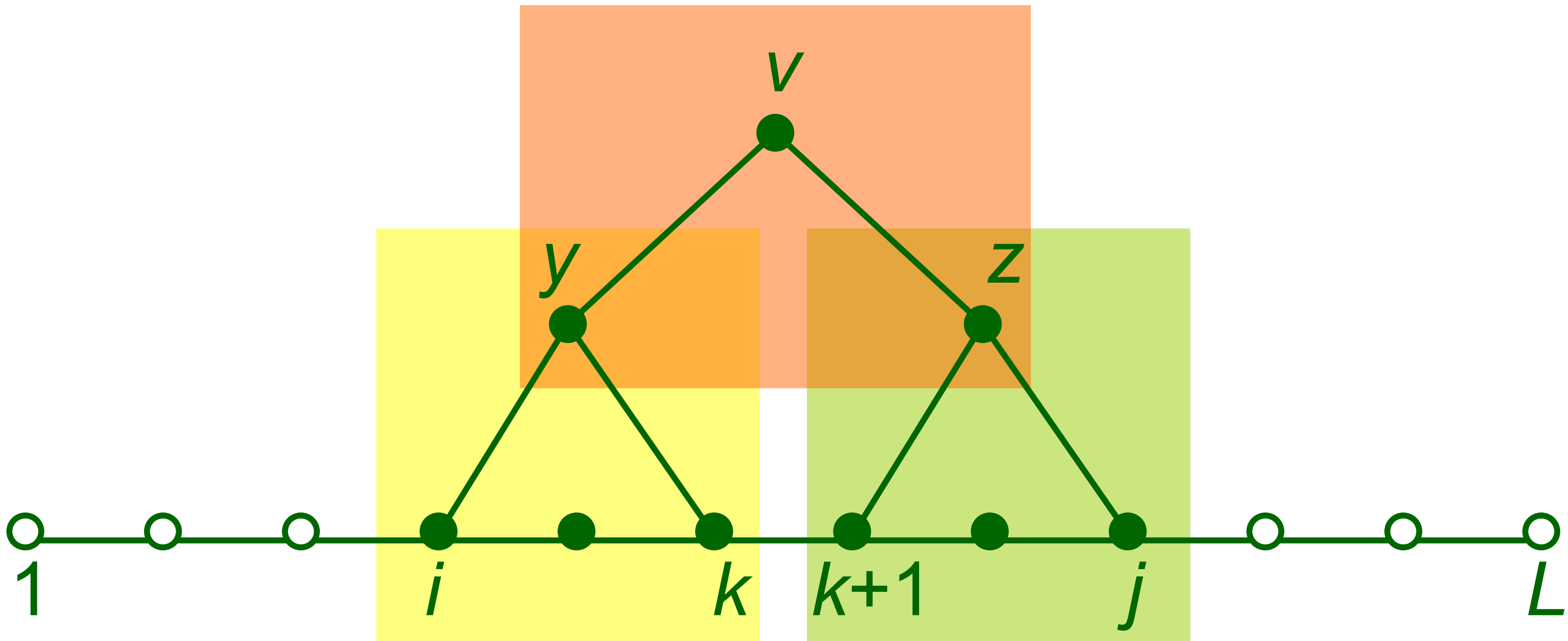
$b_C \rightarrow C$

$b_A \rightarrow A$

$$\alpha(2,6,s) = t_s(b_G, p)\,\alpha(2,2,b_G)\,\alpha(3,6,p) + t_s(b_G, s)\,\alpha(2,2,b_G)\,\alpha(3,6,s)$$



Determining the Likelihood of a Sequence: The Inside Algorithm



$$\alpha(i,j,v) = \sum_{y=1}^{M} \sum_{z=1}^{M} \sum_{k=i}^{j-1} t_v(y,z)\ \alpha(i,k,y)\ \alpha(k+1,j,z)$$

M is the number of nonterminals in the grammar

The Inside Algorithm

- Initialization (for $i = 1$ to L, $v = 1$ to M)

$$\alpha(i,i,v) = e_v(x_i)$$

- Iteration (for $i = L\text{-}1$ to 1, $j = i+1$ to L, $v = 1$ to M)

$$\alpha(i,j,v) = \sum_{y=1}^{M}\sum_{z=1}^{M}\sum_{k=i}^{j-1} t_v(y,z)\,\alpha(i,k,y)\,\alpha(k+1,j,z)$$

- Termination

$$\Pr(x) = \alpha(1,L,1)$$

start nonterminal

Learning SCFG Parameters

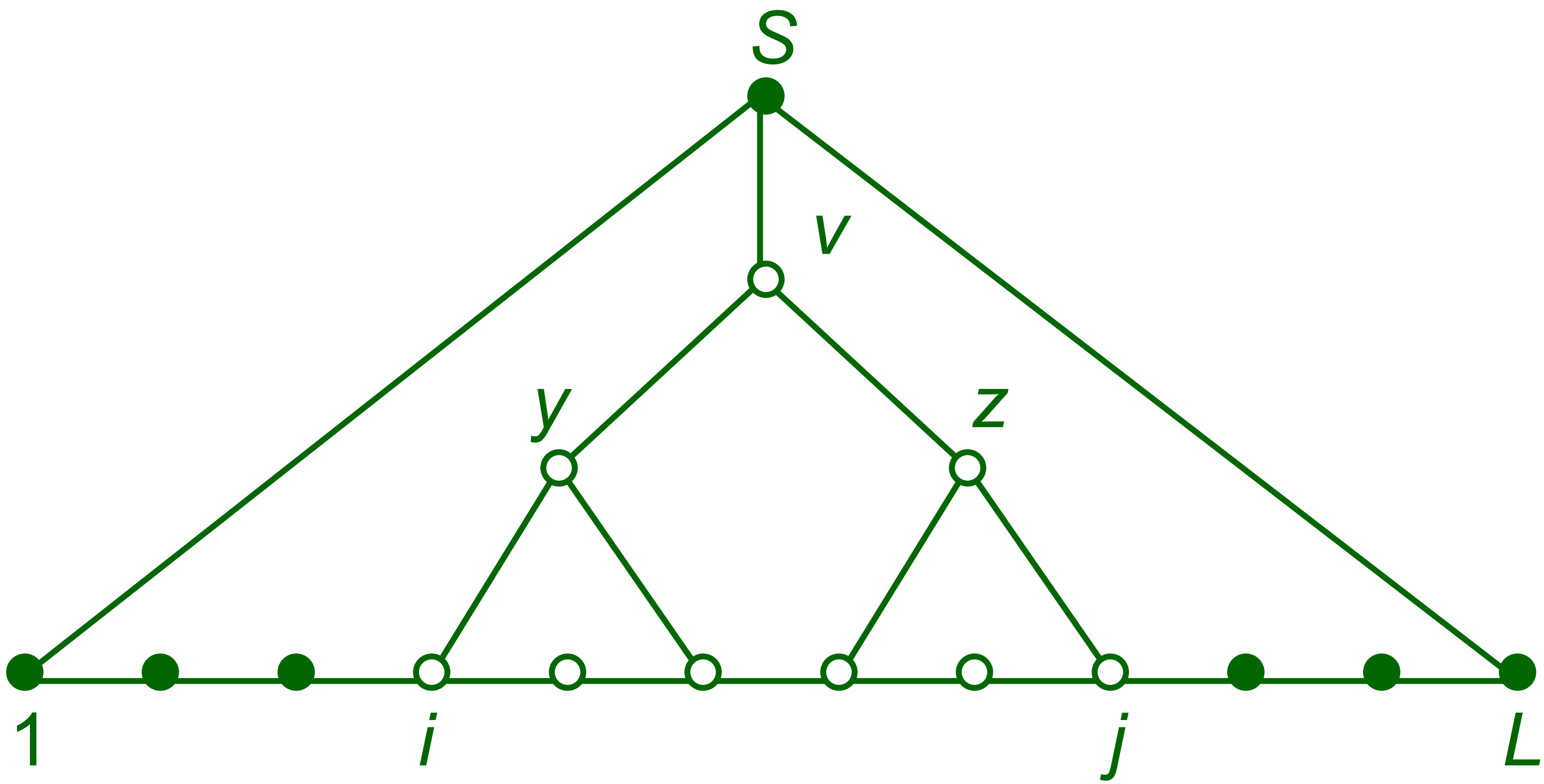
- If we know the parse tree for each training sequence, learning the SCFG parameters is simple
  - no hidden part of the problem during training
  - count how often each parameter (i.e. production) is used
  - normalize/smooth to get probabilities

- More commonly, there are many possible parse trees per sequence – we don't know which one is correct
  - thus, use an EM approach (Inside-Outside)
  - iteratively
    - determine *expected* # times each production is used
      - consider all parses
      - weight each by its probability
    - set parameters to maximize likelihood given these counts

The Inside-Outside Algorithm

- We can learn the parameters of an SCFG from training sequences using an EM approach called Inside-Outside
- In the E-step, we determine
  - the expected number of times each *nonterminal* is used in parses $c(v)$
  - the expected number of times each *production* is used in parses
$$c(v \rightarrow yz)$$
$$c(v \rightarrow A)$$
- In the M-step, we update our production probabilities

The Outside Algorithm



- $\beta(i, j, v)$: the probability of parse trees rooted at the start nonterminal, excluding the probability of all subtrees rooted at nonterminal v covering the subsequence from i to j

Outside Calculation Example

$s \rightarrow b_G\ p$

$p \rightarrow s\ b_C$

$s \rightarrow b_G\ s$

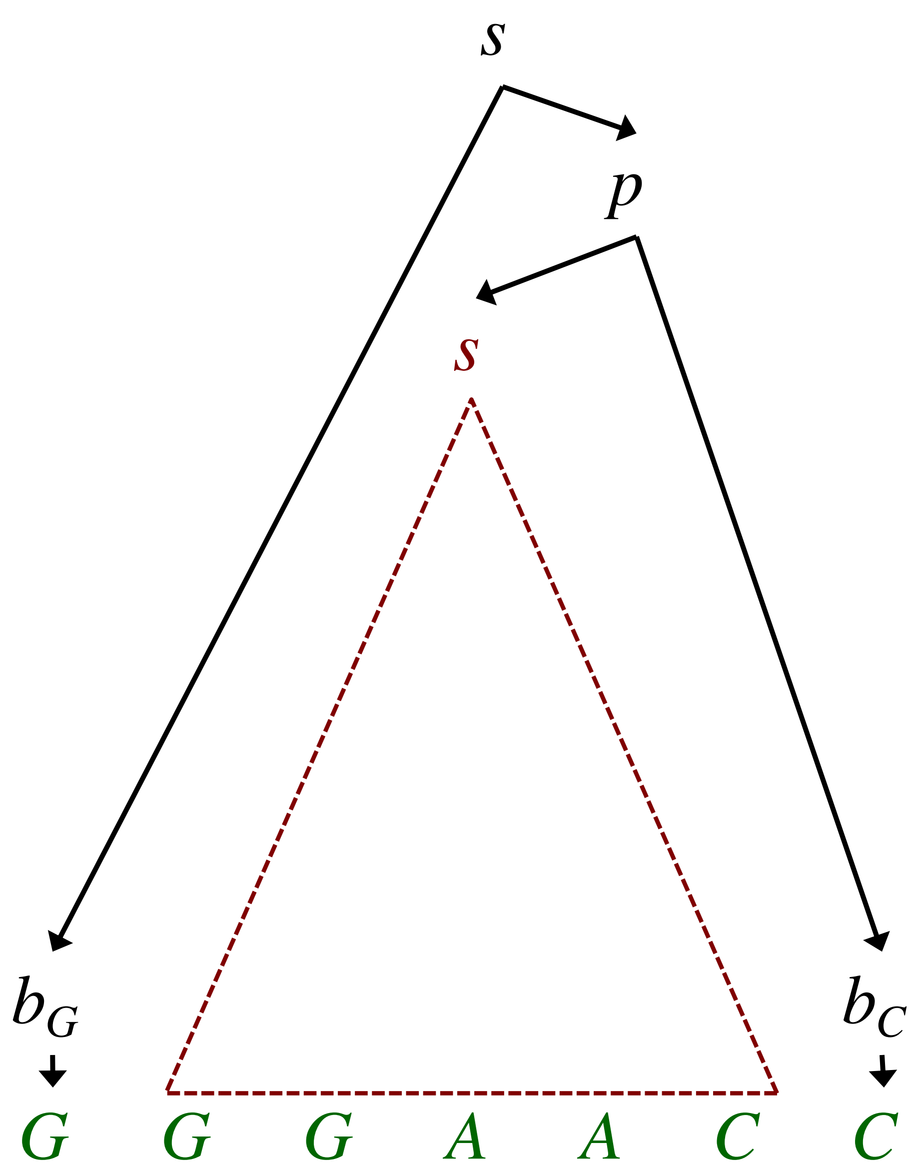
$s \rightarrow b_A\ b_A$

$b_G \rightarrow G$

$b_C \rightarrow C$

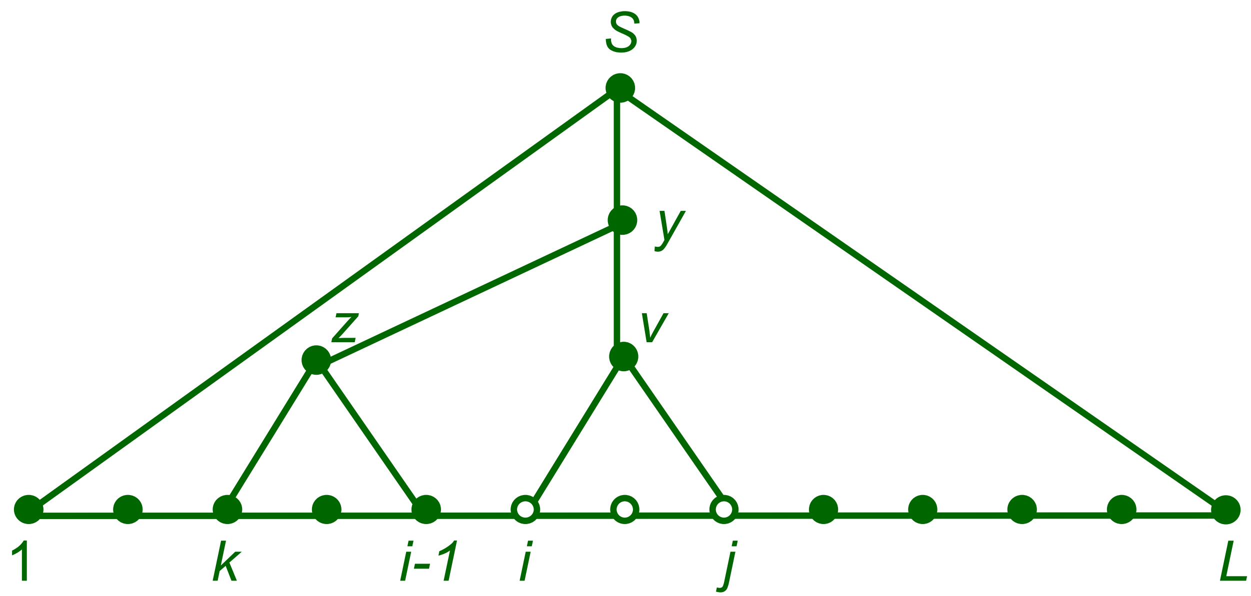
$b_A \rightarrow A$

$$\beta(2,6,s) = t_p(s, b_C)\alpha(7,7,b_C)\beta(2,7,p)$$



The Outside Algorithm

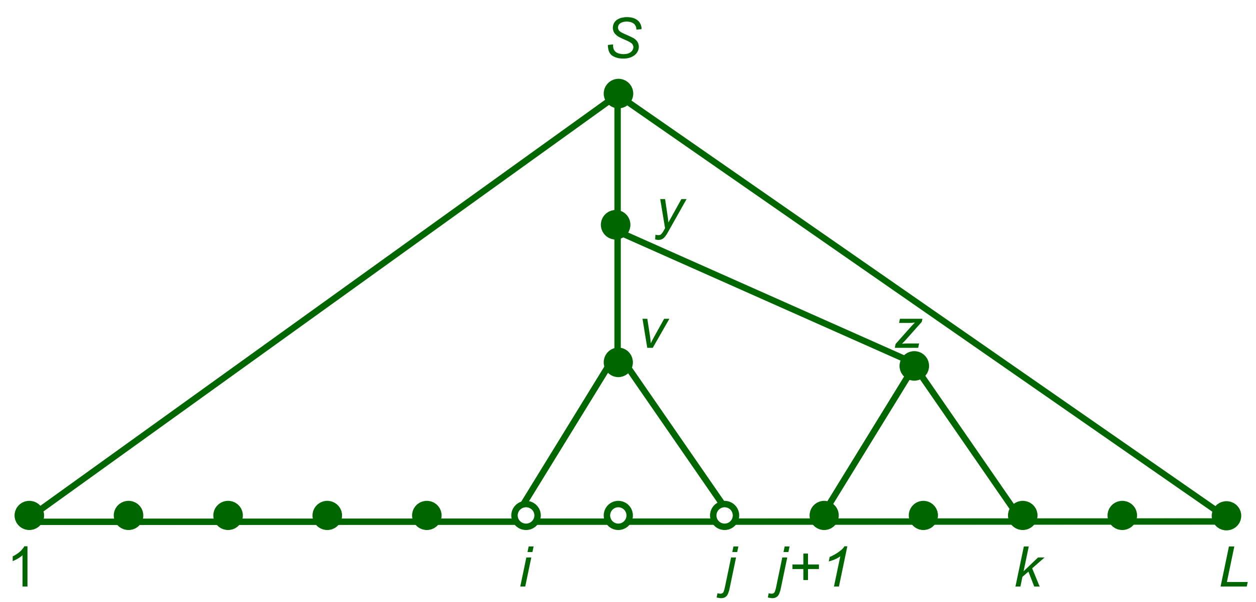
- We can recursively calculate $\beta(i, j, v)$ from β values we've calculated for y
- The first case we consider is where v is used in productions of the form: $y \rightarrow zv$



$$\sum_{y=1}^{M} \sum_{z=1}^{M} \sum_{k=1}^{i-1} t_y(z,v)\, \alpha(k, i-1, z)\, \beta(k, j, y)$$

The Outside Algorithm

- The second case we consider is where v is used in productions of the form: $y \rightarrow vz$



$$\sum_{y=1}^{M}\sum_{z=1}^{M}\sum_{k=j+1}^{L} t_y(v,z)\,\alpha(j+1,k,z)\,\beta(i,k,y)$$

The Outside Algorithm

- Initialization

$$\beta(1, L, 1) = 1 \qquad \text{(the } \mathit{start} \text{ nonterminal)}$$

$$\beta(1, L, v) = 0 \qquad \text{for } v = 2 \text{ to } M$$

- Iteration (for *i* = 1 to *L*, *j* = *L* to *i*, *v* = 1 to *M*)

$$\beta(i, j, v) = \sum_{y=1}^{M} \sum_{z=1}^{M} \sum_{k=1}^{i-1} t_y(z, v)\, \alpha(k, i-1, z)\, \beta(k, j, y) +$$

$$\sum_{y=1}^{M} \sum_{z=1}^{M} \sum_{k=j+1}^{L} t_y(v, z)\, \alpha(j+1, k, z)\, \beta(i, k, y)$$

The Inside-Outside Algorithm

- We can learn the parameters of an SCFG from training sequences using an EM approach called Inside-Outside
- In the E-step, we determine
  - the expected number of times each *nonterminal* is used in parses $c(v)$
  - the expected number of times each *production* is used in parses
$$c(v \to yz)$$
$$c(v \to A)$$
- In the M-step, we update our production probabilities

The Inside-Outside Algorithm

- The EM re-estimation equations (for 1 sequence) are:

$$\hat{e}_v(A) = \frac{c(v \rightarrow A)}{c(v)} = \frac{\sum_{i \mid x_i = A} \beta(i,i,v) e_v(A)}{\sum_{i=1}^{L} \sum_{j=i}^{L} \beta(i,j,v) \alpha(i,j,v)}$$

cases where v used to generate *A*

cases where v used to generate any subsequence

$$\hat{t}_v(y,z) = \frac{c(v \rightarrow yz)}{c(v)}$$

$$= \frac{\sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=i}^{j-1} t_v(y,z)\, \beta(i,j,v)\, \alpha(i,k,y)\, \alpha(k+1,j,z)}{\sum_{i=1}^{L} \sum_{j=i}^{L} \beta(i,j,v)\, \alpha(i,j,v)}$$

Finding the Most Likely Parse: The CYK Algorithm

- Involves filling in a 3D matrix

$$\gamma(i, j, v)$$

representing the most probable parse subtree rooted at nonterminal v for the subsequence from i to j

- and a matrix for the traceback

$$\tau(i, j, v)$$

storing information about the production at the top of this parse subtree

The CYK Algorithm

- Initialization (for $i = 1$ to L, $v = 1$ to M)

$$\gamma(i,i,v) = \log e_v(x_i)$$

$$\tau(i,i,v) = (0,0,0)$$

- Iteration (for $i = 1$ to $L - 1$, $j = i{+}1$ to L, $v = 1$ to M)

$$\gamma(i,j,v) = \max_{\substack{y,z \\ k=i\ldots j-1}} \{\gamma(i,k,y) + \gamma(k+1,j,z) + \log t_v(y,z)\}$$

$$\tau(i,j,v) = \arg\max_{\substack{y,z \\ k=i\ldots j-1}} \{\gamma(i,k,y) + \gamma(k+1,j,z) + \log t_v(y,z)\}$$

- Termination

$$\log P(x,\hat{\pi} \mid \theta) = \gamma(1,L,1)$$

start nonterminal

The CYK Algorithm Traceback

- Initialization:

  push ($1, L, 1$) on the stack

- Iteration:

  pop (i, j, v) // pop subsequence/nonterminal pair

  (y, z, k) = τ(i, j, v) // get best production identified by CYK

  if (y, z, k) == ($0,0,0$) // indicating a leaf

  attach x_i as the child of v

  else

  attach y, z to parse tree as children of v

  push(i, k, y)

  push($k+1,\ j, z$)

Comparison of SCFG Algorithms to HMM Algorithms

	HMM algorithm	SCFG algorithm
optimal alignment	Viterbi	CYK
probability of sequence	forward	inside
EM parameter estimation	forward-backward	inside-outside
memory complexity	$O(LM)$	$O(L^2M)$
time complexity	$O(LM^2)$	$O(L^3M^3)$