

# Grammatical Modelling of Hidden Structures

## HMMs and Grammars

Rune Lyngsø

University of Iceland, 2<sup>nd</sup> of June 2009

# Outline

Hidden Markov Models

Regular Grammars

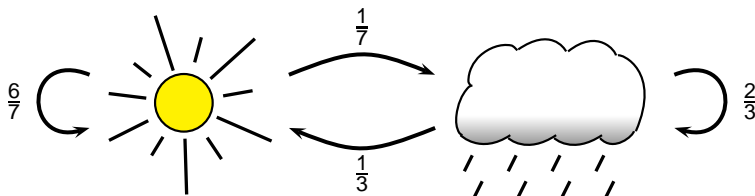
Context Free Grammars

The Rest of the Grammars

# Markov Models

## Empirical Fact

Weather tomorrow is most likely to be the same as today

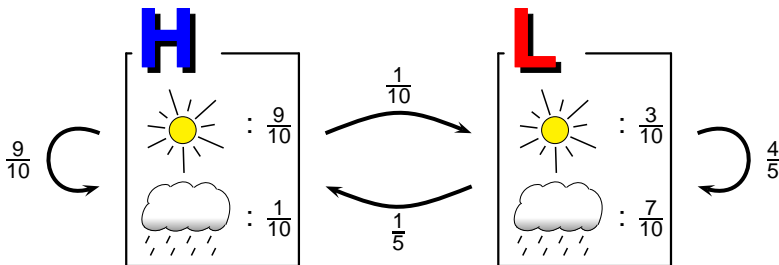


This is typical example of Markov behaviour

# Hidden Markov Models

## More realism

Underlying unobserved phenomena may be the conserved factor

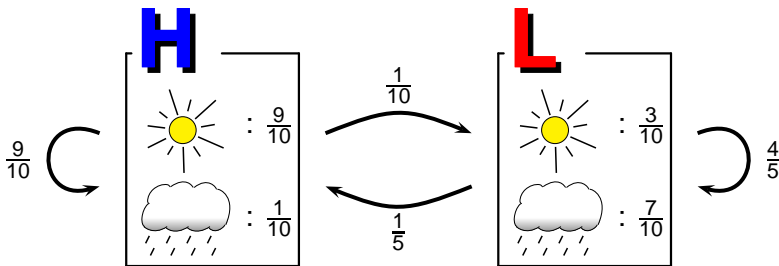


States of the Markov chain are not observed directly – observed characters are emitted from the hidden chain of states

# Hidden Markov Models

## More realism

Underlying unobserved phenomena may be the conserved factor



States of the Markov chain are not observed directly – observed characters are emitted from the hidden chain of states

## Other types of states

Silent states have no emissions, and special start and end states results in distribution over *finite* observed sequences

# Common HMM Uses

## Annotation



# Common HMM Uses

## Annotation

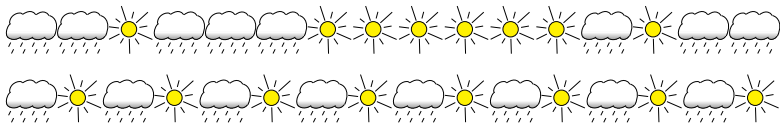


# Common HMM Uses

## Annotation



## Classification

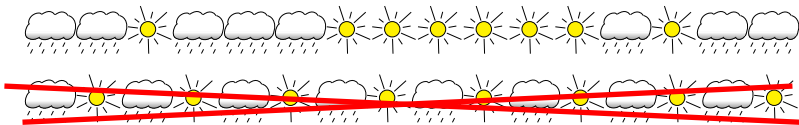


# Common HMM Uses

## Annotation















## Classification



# Annotation – Naïve Approach

Observed ☁☁☀ – enumerate all possibilities:

Path	$Pr(\text{Path})$	$Pr(\text{Data} \mid \text{Path})$	$Pr(\text{Data})$
	$\pi$  $a$   $a$  	$e$   $e$   $e$  	0.04704

# Annotation – Naïve Approach

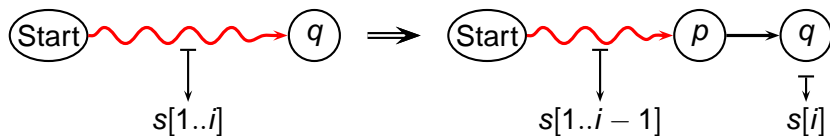
Observed ☁☁☀ – enumerate all possibilities:

Path	$Pr(\text{Path})$	$Pr(\text{Data} \mid \text{Path})$	$Pr(\text{Data})$
	$\pi_{L,L,L} a_{L,L,L} a_{L,L,L}$	$e_{L,\text{☁}} e_{L,\text{☁}} e_{L,\text{☀}}$	0.04704
	$\pi_{L,L,H} a_{L,L,H} a_{L,H,H}$	$e_{L,\text{☁}} e_{L,\text{☁}} e_{H,\text{☀}}$	0.03528
	$\pi_{L,H,L} a_{L,H,L} a_{H,L,L}$	$e_{L,\text{☁}} e_{H,\text{☁}} e_{L,\text{☀}}$	0.00021
	$\pi_{L,H,H} a_{L,H,H} a_{H,H,H}$	$e_{L,\text{☁}} e_{H,\text{☁}} e_{H,\text{☀}}$	0.00567
	$\pi_{H,L,L} a_{H,L,L} a_{L,L,L}$	$e_{H,\text{☁}} e_{L,\text{☁}} e_{L,\text{☀}}$	0.00084
	$\pi_{H,H,L} a_{H,H,L} a_{L,L,H}$	$e_{H,\text{☁}} e_{L,\text{☁}} e_{H,\text{☀}}$	0.00063
	$\pi_{H,H,L} a_{H,H,L} a_{L,L,H}$	$e_{H,\text{☁}} e_{H,\text{☁}} e_{L,\text{☀}}$	0.00014
	$\pi_{H,H,H} a_{H,H,H} a_{H,H,H}$	$e_{H,\text{☁}} e_{H,\text{☁}} e_{H,\text{☀}}$	0.00365

Number of paths grows exponentially!

# Recursion...

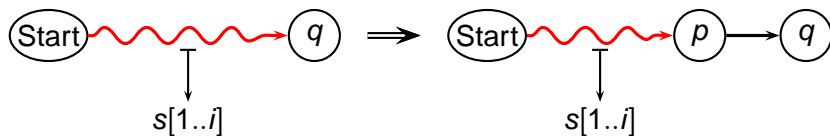
Last state is non-silent



$$V(q, i) = \max_p \{ V(p, i - 1) \cdot Pr(p \rightarrow q) \cdot Pr(s[i] | q) \}$$

# Recursion...

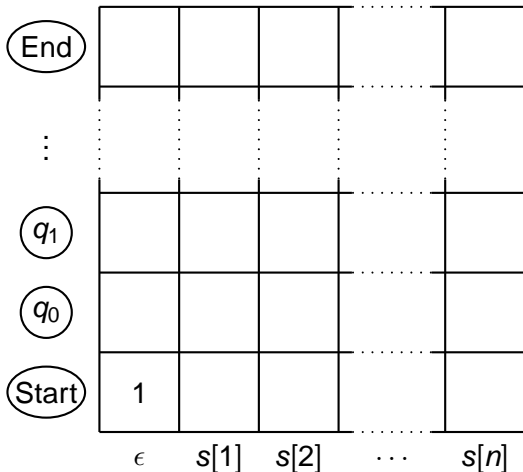
Last state is silent



$$V(q, i) = \max_p \{ V(p, i) \cdot Pr(p \rightarrow q) \}$$

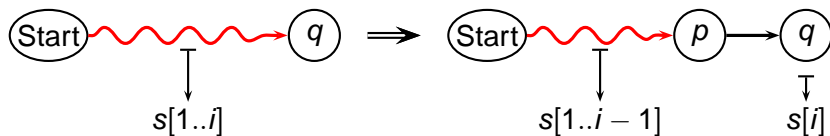
## ... and a Table

Can compute probabilities efficiently by dynamic programming



# Forward Algorithm

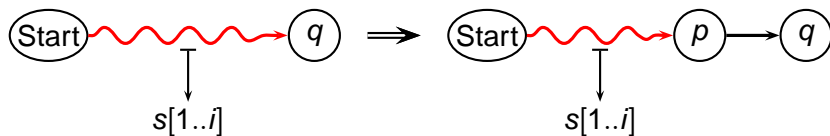
Last state is non-silent



$$F(q, i) = \sum_p F(p, i-1) \cdot P(p \rightarrow q) \cdot P(s[i] | q)$$

# Forward Algorithm

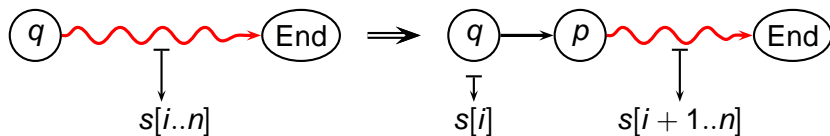
Last state is silent



$$F(q, i) = \sum_p F(p, i) \cdot P(p \rightarrow q)$$

# Backward Algorithm

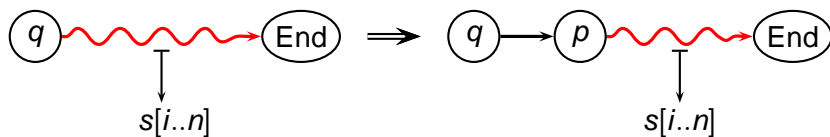
First state is non-silent



$$B(q, i) = \sum_p B(p, i + 1) \cdot P(q \rightarrow p) \cdot P(s[i] | q)$$

# Backward Algorithm

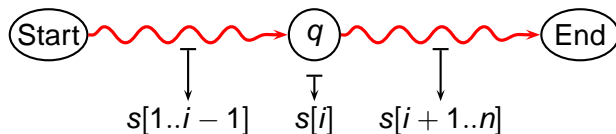
First state is non-silent



$$B(q, i) = \sum_p B(p, i) \cdot P(q \rightarrow p)$$

## Maximum A Posteriori Decoding

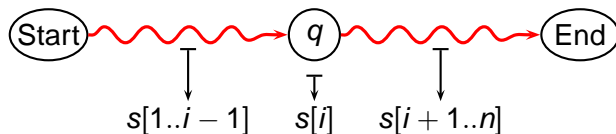
Forward and backward algorithms give  $P(q \text{ emits } s[i])$ :



$$P(q \text{ emits } s[i]) = \frac{P(q \text{ emits } s[i], s)}{P(s)} = \frac{F(q, i)B(q, i)}{F(\text{End}, |s|)P(s[i] | q)}$$

## Maximum A Posteriori Decoding

Forward and backward algorithms give  $P(q \text{ emits } s[i])$ :



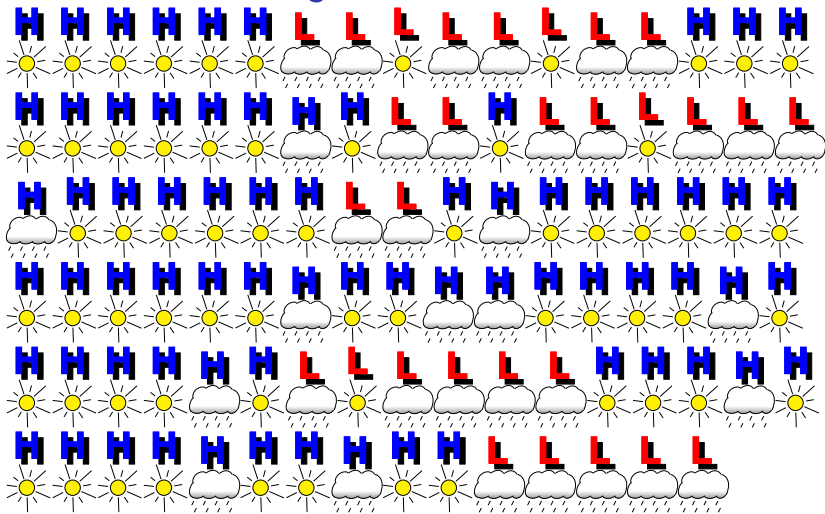
$$P(q \text{ emits } s[i]) = \frac{P(q \text{ emits } s[i], s)}{P(s)} = \frac{F(q, i)B(q, i)}{F(\text{End}, |s|)P(s[i] | q)}$$

### MAP Decoding Path

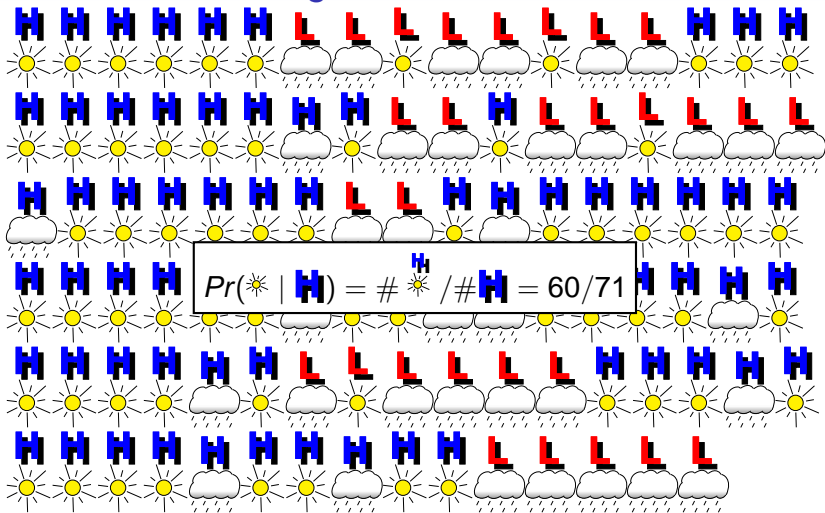
$$A(q, i) = \max_{p: P(p \rightarrow q) > 0} \{A(p, i-1) + P(q \text{ emits } s[i])\}$$

Gives annotation with most expected correct positions

## Training – Annotated Data



# Training – Annotated Data



# Training – Unannotated Data

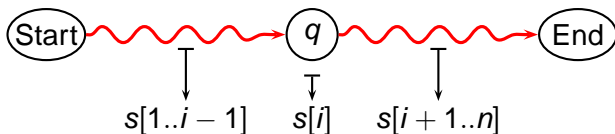
Use expectations instead of observed frequencies:

$$Pr(\odot | \mathbf{H}) = \mathbb{E} \left[ \# \odot \right] / \mathbb{E} [\# \mathbf{H}]$$

## Training – Unannotated Data

Use expectations instead of observed frequencies:

$$Pr(\odot \mid \mathbf{H}) = \mathbb{E} \left[ \# \odot \right] / \mathbb{E} \left[ \# \mathbf{H} \right]$$



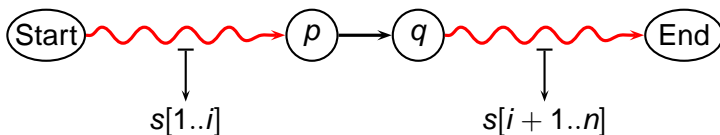
$$\mathbb{E} \left[ \# \sigma^q \right] = \sum_{i: s[i]=\sigma} P(q \text{ emits } s[i])$$

$$\mathbb{E} \left[ \# q \right] = \sum_i P(q \text{ emits } s[i])$$

## Training – Unannotated Data

Use expectations instead of observed frequencies:

$$Pr(\mathbf{H} \rightarrow \mathbf{L}) = \mathbb{E} [\# (\mathbf{H} \rightarrow \mathbf{L})] / \mathbb{E} [\#\mathbf{H}]$$



$$\mathbb{E} [\# (p \rightarrow q)] = \sum_i \frac{F(p, i)B(q, i + 1)}{F(\text{End}, |s|)}$$

$$\mathbb{E} [\#q] = \sum_i P(q \text{ emits } s[i])$$

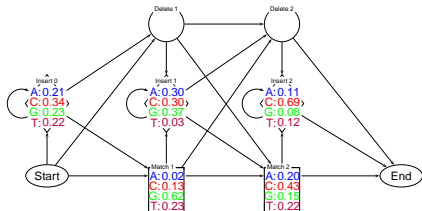
# HMM Applications

## Bioinformatics

- Sequence alignment
- Gene finding
- Protein secondary structure prediction
- Genetic variation
- Inference of genealogical histories
- Homology modelling

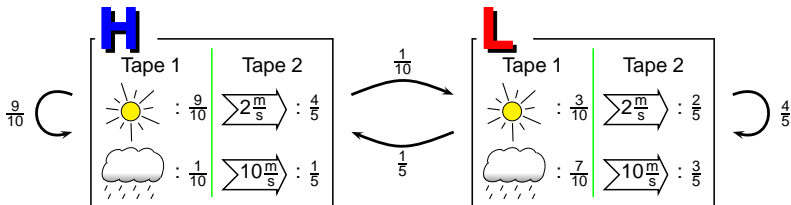
## Speech Recognition

## Signal Processing



## Generating Several Outputs

There may be more than one type of observables for each state

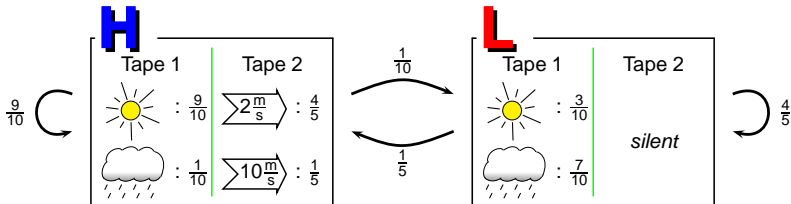


**States all-silent or all-emitting**

Really just a single output with a large alphabet

## Generating Several Outputs

There may be more than one type of observables for each state

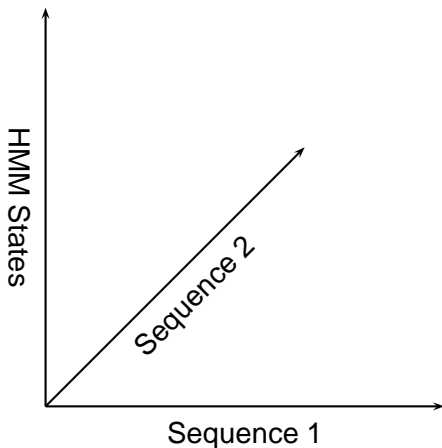


**States can be partially silent**

More complex – track progress in all outputs simultaneously

## Algorithms for Multitape HMMs

Same as for normal HMMs, just with more dimensions:

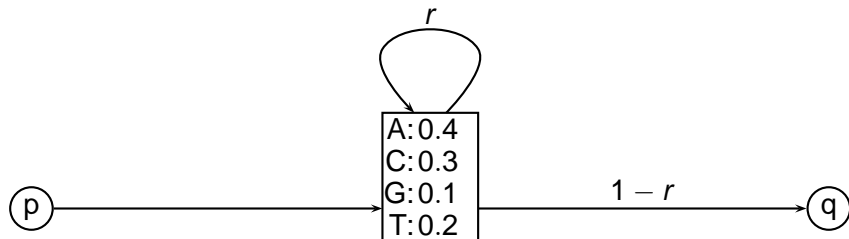


$$V(q, i, j) = \max_p \{ V(p, i-1, j-1) \cdot P(p \rightarrow q) P(s[i], t[j] | q) \}$$

# Non-geometric Waiting Times

Waiting times in states are geometric

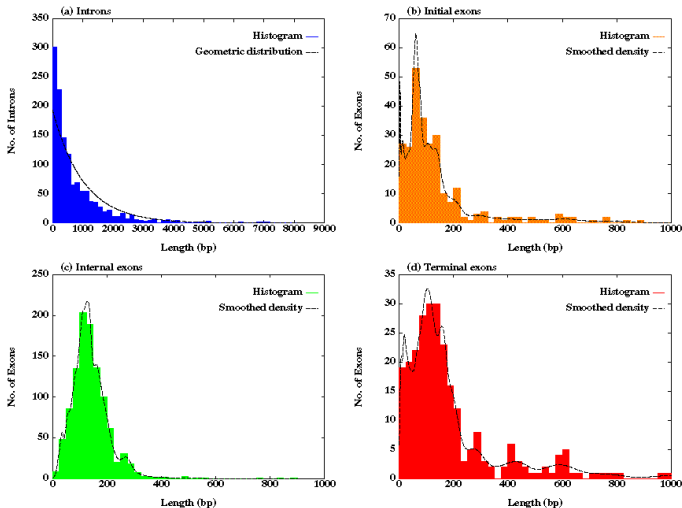
$$P(\text{emit } n \text{ symbols before moving on}) = r^{n-1}(1 - r)$$



# Non-geometric Waiting Times

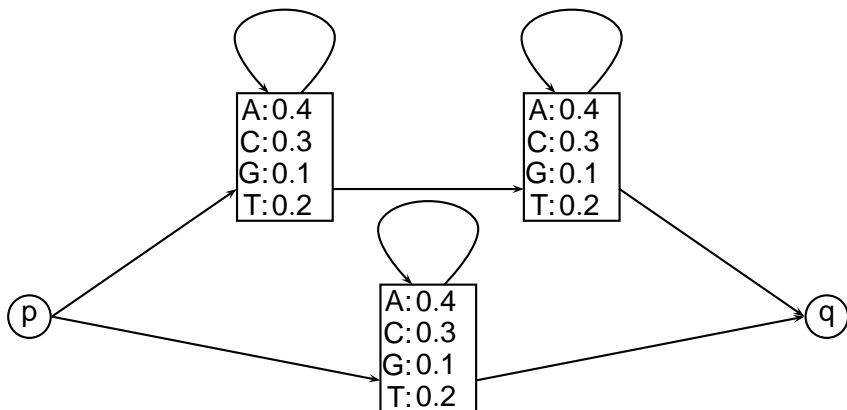
Waiting times in states are geometric

Length distributions of human introns and initial, internal and terminal exons



## Non-geometric Waiting Times

Waiting times in states are geometric, but can be made more general by replicating states:



# Class HMMs

In a class HMM

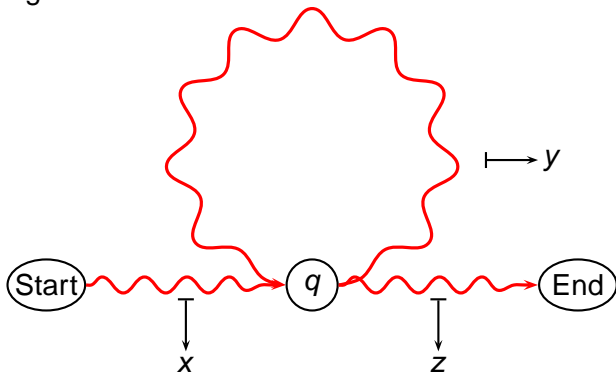
- all states belong to a class, e.g. insertion
- a class may have several states belonging to it

Annotation problem for class HMMs is finding annotation with classes such that *sum over all paths* yielding that annotation is maximised

Viterbi algorithm not guaranteed to work – problem is in general hard but efficiently solvable for most class HMMs published (Brejova, Brown, Vinar, 2007)

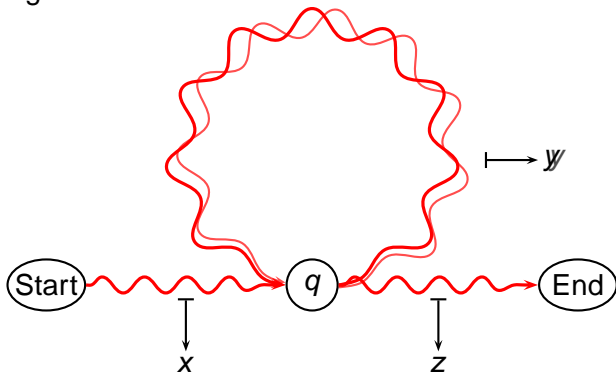
## Limitations of Hidden Markov Models

For sufficiently long sequences, there will be a cycle in any path generating it from a fixed HMM



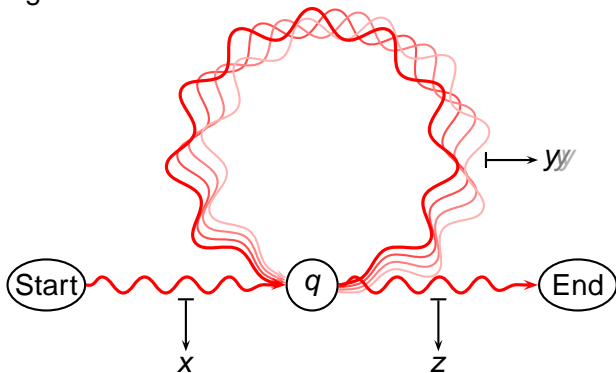
## Limitations of Hidden Markov Models

For sufficiently long sequences, there will be a cycle in any path generating it from a fixed HMM



## Limitations of Hidden Markov Models

For sufficiently long sequences, there will be a cycle in any path generating it from a fixed HMM



**Consequence:** Cannot generate e.g. palindromes and

$\{\text{☀}^i \text{☁}^i \mid i > 0\}$

# Maintaining the State of an HMM

We can maintain progress of an HMM as pair:

$$(\epsilon, \mathbf{H}) \Rightarrow (\text{☀}, \mathbf{H}) \Rightarrow (\text{☀☀}, \mathbf{L}) \Rightarrow (\text{☀☀☁}, \mathbf{H}) \Rightarrow \dots$$

## Maintaining the State of an HMM

We can maintain progress of an HMM as pair:

$$(\epsilon, \mathbf{H}) \Rightarrow (\text{☀}, \mathbf{H}) \Rightarrow (\text{☀☀}, \mathbf{L}) \Rightarrow (\text{☀☀☁}, \mathbf{H}) \Rightarrow \dots$$

We could also just keep the state at the end of the sequence:

$$\mathbf{H} \Rightarrow \text{☀} \mathbf{H} \Rightarrow \text{☀☀} \mathbf{L} \Rightarrow \text{☀☀☁} \mathbf{H} \Rightarrow \dots$$

At each update we replace a state with an observation and a new state

# Fundamentals of Grammars

A formal grammar consists of

- A set of variables  $\mathbf{V}$ , e.g.  $\{\mathbf{H}, \mathbf{L}\}$
- A set of terminal symbols  $\Sigma$ , e.g.  $\{\text{☀}, \text{☁}\}$
- A set of productions  $\mathbf{P}$  of the form  $x \rightarrow y$  where  $x, y \in (\mathbf{V} \cup \Sigma)^*$  and  $x$  contains at least one element from  $\mathbf{V}$ , e.g.  $\mathbf{H} \rightarrow \text{☀} \mathbf{H} \in \mathbf{P}$
- A special start variable  $S \in \mathbf{V}$ , e.g.  $S = \mathbf{H}$

Derivations start from the single start variable and keeps applying a suitable production until there are no variables left in the string

# Stochastic Grammars

## Basics

- Probability associated with each production
- Probabilities of productions replacing variable sums to 1

## Example: Grammar probabilities from weather HMM

$$Pr(\mathbf{H} \rightarrow \odot \mathbf{L}) = Pr(\mathbf{H} \rightarrow \mathbf{L}) \cdot Pr(\odot \mid \mathbf{L})$$

## Probability of a...

- Derivation is product of probabilities of productions applied
- String is sum of probabilities of derivations generating it

## Variations of Regular Grammars

**Left regular:** productions  $U \rightarrow \sigma V$  or  $U \rightarrow \sigma$  ( $U, V \in \mathbf{V}$  and  $\sigma \in \Sigma$ )

**Extended left regular:** productions  $U \rightarrow xV$  or  $U \rightarrow x$  ( $U \in \mathbf{V}$  and  $x \in \Sigma^*$ )

**Right regular:** productions  $U \rightarrow V\sigma$  or  $U \rightarrow \sigma$  ( $U, V \in \mathbf{V}$  and  $\sigma \in \Sigma$ )

**Extended right regular:** productions  $U \rightarrow Vx$  or  $U \rightarrow x$  ( $U \in \mathbf{V}$  and  $x \in \Sigma^*$ )

# Context Free Grammars

All productions on the form  $U \rightarrow x$  where  $U \in \mathbf{V}$  and  $x \in (\mathbf{V} \cup \Sigma)^*$

Example:

$$S \rightarrow aSa$$

$$S \rightarrow bSb$$

$$S \rightarrow \epsilon$$

# Context Free Grammars

All productions on the form  $U \rightarrow x$  where  $U \in \mathbf{V}$  and  $x \in (\mathbf{V} \cup \Sigma)^*$

Example:

$$S \rightarrow aSa \mid bSb \mid \epsilon$$

# Context Free Grammars

All productions on the form  $U \rightarrow x$  where  $U \in \mathbf{V}$  and  $x \in (\mathbf{V} \cup \Sigma)^*$

Example:

$$S \rightarrow aSa \mid bSb \mid \epsilon$$

Derivation of palindrome *abba*:

**S**

# Context Free Grammars

All productions on the form  $U \rightarrow x$  where  $U \in \mathbf{V}$  and  $x \in (\mathbf{V} \cup \Sigma)^*$

Example:

$$S \rightarrow aSa \mid bSb \mid \epsilon$$

Derivation of palindrome *abba*:

$$S \Rightarrow aSa$$

# Context Free Grammars

All productions on the form  $U \rightarrow x$  where  $U \in \mathbf{V}$  and  $x \in (\mathbf{V} \cup \Sigma)^*$

Example:

$$S \rightarrow aSa \mid bSb \mid \epsilon$$

Derivation of palindrome *abba*:

$$S \Rightarrow aSa \Rightarrow abSba$$

# Context Free Grammars

All productions on the form  $U \rightarrow x$  where  $U \in \mathbf{V}$  and  $x \in (\mathbf{V} \cup \Sigma)^*$

Example:

$$S \rightarrow aSa \mid bSb \mid \epsilon$$

Derivation of palindrome *abba*:

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abba$$

## Another Example – Parentheses

Balanced string of parentheses consist of

- An empty string
- A balanced string enclosed between a ( and a )
- Two balanced strings concatenated

## Another Example – Parentheses

Balanced string of parentheses consist of

- An empty string
- A balanced string enclosed between a ( and a )
- Two balanced strings concatenated

$$S \rightarrow \epsilon \mid (S) \mid SS$$

## Another Example – Parentheses

Balanced string of parentheses consist of

- An empty string
- A balanced string enclosed between a ( and a )
- Two balanced strings concatenated

$$S \rightarrow \epsilon \mid (S) \mid SS$$

String  $()()$  can be derived e.g. as

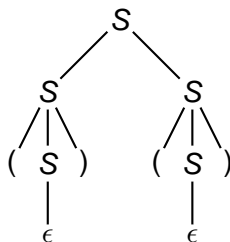
$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$$

## Parse Trees

An alternative to derivation sequences are parse trees.  
For example

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$$

would be represented by the parse tree



# Ambiguous Grammars

Context-free grammars introduce choice of variable to apply production to:

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$$

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow (S)() \Rightarrow ()()$$

These share the same parse tree we have already seen

## Ambiguous Grammars

Context-free grammars introduce choice of variable to apply production to:

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$$

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow (S)() \Rightarrow ()()$$

These share the same parse tree we have already seen

$$S \Rightarrow \epsilon$$

$$S \Rightarrow SS \Rightarrow S \Rightarrow \epsilon$$

Only derivations with different parse trees are distinct.

# Chomsky Normal Form

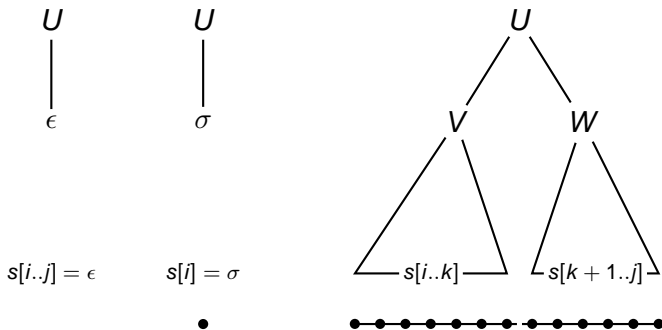
A grammar is in Chomsky Normal Form (CNF) if all productions are

- $S \rightarrow \epsilon$
- $U \rightarrow \sigma, U \in \mathbf{V}$  and  $\sigma \in \Sigma$
- $U \rightarrow VW, U \in \mathbf{V}$  and  $V, W \in \mathbf{V} \setminus \{S\}$

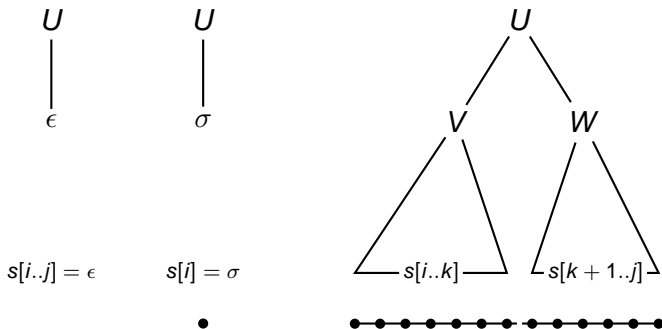
Any context-free grammar can be converted to normal form

But there are issues with probabilities when converting stochastic context-free grammars

# Cocke-Younger-Kasami Algorithm

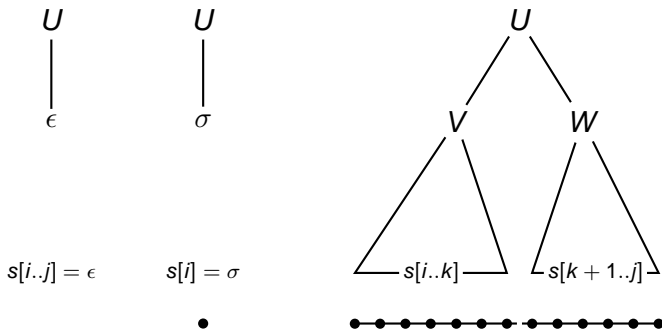


# Cocke-Younger-Kasami Algorithm



$$C(U, i, j) = Pr(S \rightarrow \epsilon) \text{ if } U = S \text{ and } j = i - 1$$

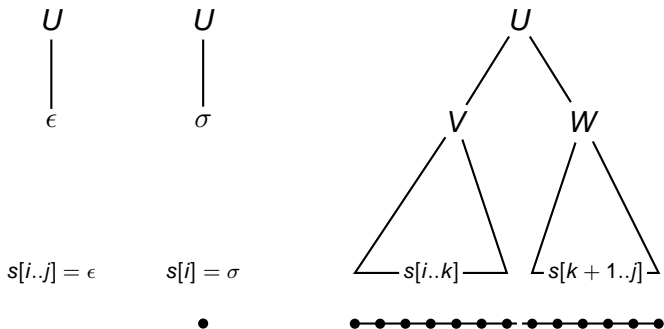
# Cocke-Younger-Kasami Algorithm



$$C(U, i, j) = Pr(U \rightarrow s[i]) \text{ if } i = j$$



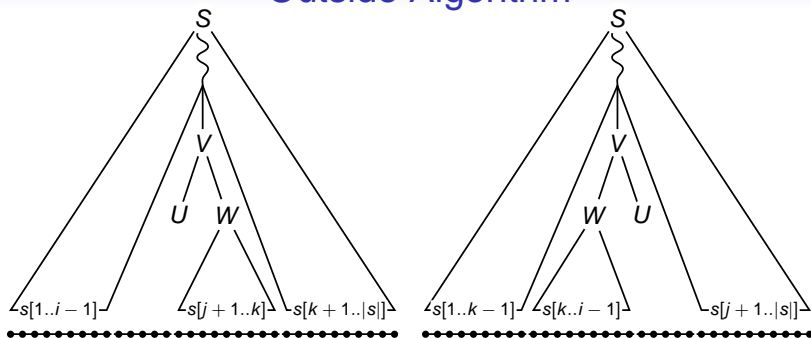
## Cocke-Younger-Kasami Algorithm



Dynamic programming once again allows efficient computation (though cubic in sequence length, compared to linear for HMMs)



# Outside Algorithm



$$O(U, i, j) = 1 \text{ if } U = S, i = 1, \text{ and } j = |s|$$

$$O(U, i, j) = \sum_{\substack{V \rightarrow UW \in P \\ k > j}} \Pr(V \rightarrow UW) O(V, i, k) I(W, j+1, k)$$

$$+ \sum_{\substack{V \rightarrow WU \in P \\ k < i}} \Pr(V \rightarrow WU) O(V, k, j) I(W, k, i-1) \text{ if } i \leq j$$

# Parameter Estimation

## Annotated Data

Each production probability is set to the frequency with which it is used to replace the variable on the its left hand side

## Unannotated Data

Instead of observed frequencies, expected frequencies are computed using the inside/outside algorithms

In both cases a small pseudocount may be added to observed/expected counts

# Bioinformatics Example I: RNA Secondary Structures

Two bases introduced by the same production are paired, all other are unpaired

$S \rightarrow L$	one structural component
$ LS$	at least two structural components
$L \rightarrow dFd$	beginning of stem
$ s$	unpaired base
$F \rightarrow dFd$	continuation of stem
$ LS$	end of stem

Despite its simplicity works reasonably well for a single sequence, and is very powerful for alignments of RNA (Knudsen & Hein, 2003; Dowell & Eddy, 2004)

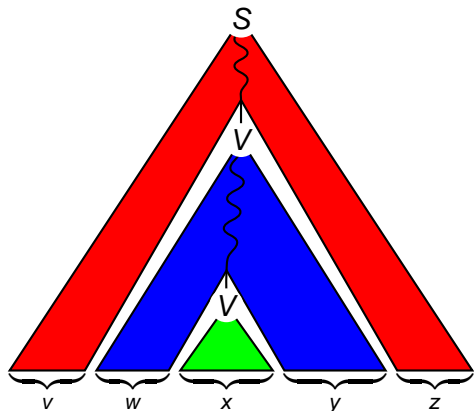
# Bioinformatics Example II: Alignment with Inversions

In addition to substitutions, insertions and deletions, also allow inversions:

$$\begin{array}{c}
 \begin{array}{cc}
 \bullet & \bullet \\
 \color{blue}{\longrightarrow} & \color{blue}{\longrightarrow} \\
 k & l \\
 \bullet & \bullet \\
 \color{green}{\longrightarrow} & \color{green}{\longrightarrow} \\
 i & j \\
 \color{red}{A(i, j; k, l)}
 \end{array}
 & = \min \left\{ \begin{array}{l}
 \min \left\{ 1 + \begin{array}{cc}
 \bullet & \bullet \\
 \color{blue}{\longrightarrow} & \color{blue}{\longrightarrow} \\
 k & l \\
 \bullet & \bullet \\
 \color{green}{\longrightarrow} & \color{green}{\longrightarrow} \\
 i & j-1
 \end{array}, 1 + \begin{array}{cc}
 \bullet & \bullet \\
 \color{blue}{\longrightarrow} & \color{blue}{\longrightarrow} \\
 k & l-1 \\
 \bullet & \bullet \\
 \color{green}{\longrightarrow} & \color{green}{\longrightarrow} \\
 i & j
 \end{array} \right\} \\
 \delta(s[i], s[j]) + \begin{array}{cc}
 \bullet & \bullet \\
 \color{blue}{\longrightarrow} & \color{blue}{\longrightarrow} \\
 k & l-1 \\
 \bullet & \bullet \\
 \color{green}{\longrightarrow} & \color{green}{\longrightarrow} \\
 i & j-1
 \end{array} \right\} \\
 \min_{a,b} \left\{ 1 + \begin{array}{cc}
 \bullet & \bullet \\
 \color{blue}{\longrightarrow} & \color{blue}{\longrightarrow} \\
 k & b \\
 \bullet & \bullet \\
 \color{green}{\longrightarrow} & \color{green}{\longrightarrow} \\
 i & a
 \end{array} + \begin{array}{cc}
 \bullet & \bullet \\
 \color{blue}{\longleftarrow} & \color{blue}{\longleftarrow} \\
 l & b+1 \\
 \bullet & \bullet \\
 \color{green}{\longrightarrow} & \color{green}{\longrightarrow} \\
 a+1 & j
 \end{array} \right\}
 \end{array} \right.
 \end{array}$$

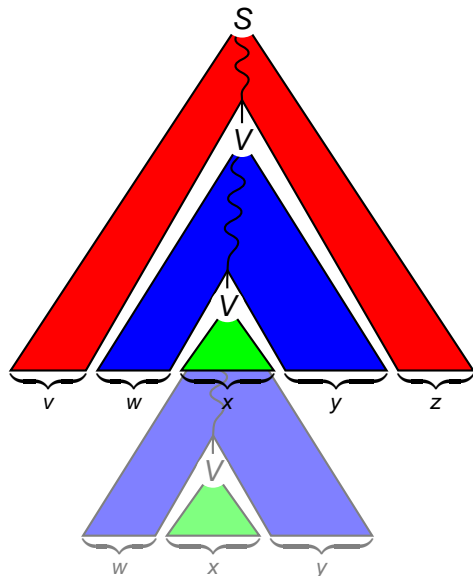
Can be captured by a context-free grammar (but not by an HMM)

# Limitations of Context-Free Grammars



When generating sufficiently long strings, there is a variable that is visited twice from root to a leaf

# Limitations of Context-Free Grammars



When generating sufficiently long strings, there is a variable that is visited twice from root to a leaf

Cannot generate  
 e.g.  $\{xx \mid x \in \Sigma^*\}$  or  
 $\{a^i b^i c^i \mid i > 0\}$

# Rest of the Chomsky Hierarchy

## Context Sensitive Grammars

- Productions  $\alpha V \beta \rightarrow \alpha x \beta$  with  $\alpha, \beta, x \in (\mathbf{V} \cup \Sigma)^*$  and  $x \neq \epsilon$

# Rest of the Chomsky Hierarchy

## Context Sensitive (Non-Contracting) Grammars

- Productions  $\alpha V \beta \rightarrow \alpha \mathbf{x} \beta$  with  $\alpha, \beta, \mathbf{x} \in (\mathbf{V} \cup \Sigma)^*$  and  $\mathbf{x} \neq \epsilon$
- Equivalent to  $\alpha V \beta \rightarrow \mathbf{x}$  with  $|\alpha V \beta| \leq |\mathbf{x}|$

Analyse  $s$  by considering all sequences not longer than  $s$

## Rest of the Chomsky Hierarchy

### Context Sensitive (Non-Contracting) Grammars

- Productions  $\alpha V\beta \rightarrow \alpha x\beta$  with  $\alpha, \beta, x \in (\mathbf{V} \cup \Sigma)^*$  and  $x \neq \epsilon$
- Equivalent to  $\alpha V\beta \rightarrow x$  with  $|\alpha V\beta| \leq |x|$

Analyse  $s$  by considering all sequences not longer than  $s$  CSG for  $\{xx \mid x \in \{a, b\}^*\}$ :

$$S \rightarrow aAS \mid bBS \mid C$$

$$Aa \rightarrow aA$$

$$Ab \rightarrow bA$$

$$Ba \rightarrow aB$$

$$Bb \rightarrow bB$$

$$AC \rightarrow CA$$

$$BC \rightarrow CB$$

$$CA \rightarrow aC$$

$$CB \rightarrow bC$$

$$C \rightarrow \epsilon$$

## Rest of the Chomsky Hierarchy

### Context Sensitive (Non-Contracting) Grammars

- Productions  $\alpha V \beta \rightarrow \alpha x \beta$  with  $\alpha, \beta, x \in (\mathbf{V} \cup \Sigma)^*$  and  $x \neq \epsilon$
- Equivalent to  $\alpha V \beta \rightarrow x$  with  $|\alpha V \beta| \leq |x|$

Analyse  $s$  by considering all sequences not longer than  $s$

### Unrestricted Grammar

- Productions  $\alpha V \beta \rightarrow x$  with  $\alpha, \beta, x \in (\mathbf{V} \cup \Sigma)^*$

Cannot guarantee that  $s$  can be analysed in general

## Rest of the Chomsky Hierarchy

### Context Sensitive (Non-Contracting) Grammars

- Productions  $\alpha V \beta \rightarrow \alpha x \beta$  with  $\alpha, \beta, x \in (\mathbf{V} \cup \Sigma)^*$  and  $x \neq \epsilon$
- Equivalent to  $\alpha V \beta \rightarrow x$  with  $|\alpha V \beta| \leq |x|$

Analyse  $s$  by considering all sequences not longer than  $s$

### Unrestricted Grammar

- Productions  $\alpha V \beta \rightarrow x$  with  $\alpha, \beta, x \in (\mathbf{V} \cup \Sigma)^*$

Cannot guarantee that  $s$  can be analysed in general

### Chomsky Hierarchy

Proposed by Noam Chomsky in 1956 – consists of Regular, Context Free, Context Sensitive, and Unrestricted Grammars

# Other Suspects

## Linear Grammars

- Productions  $U \rightarrow \sigma$ ,  $U \rightarrow V\sigma$ , and  $U \rightarrow \sigma V$

## Other Suspects

### Linear Grammars

- Productions  $U \rightarrow \sigma$ ,  $U \rightarrow V\sigma$ , and  $U \rightarrow \sigma V$

Can generate palindromes:

$$S \rightarrow aA \mid bB \mid \epsilon$$

$$A \rightarrow Sa$$

$$B \rightarrow Sb$$

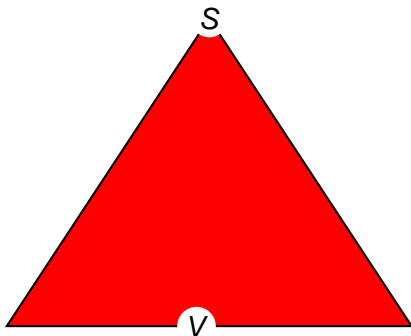
## Other Suspects

### Linear Grammars

- Productions  $U \rightarrow \sigma$ ,  $U \rightarrow V\sigma$ , and  $U \rightarrow \sigma V$

### Tree Adjoining Grammars

Context free grammars can only expand at leaves



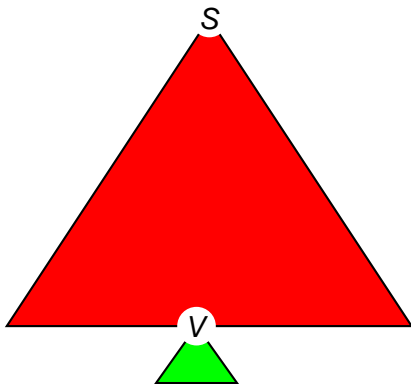
## Other Suspects

### Linear Grammars

- Productions  $U \rightarrow \sigma$ ,  $U \rightarrow V\sigma$ , and  $U \rightarrow \sigma V$

### Tree Adjoining Grammars

Context free grammars can only expand at leaves



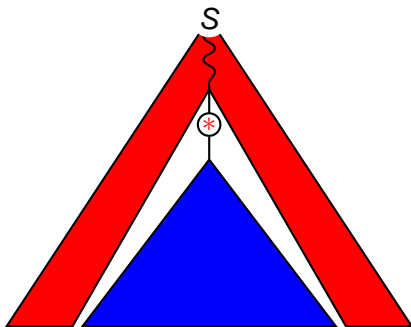
## Other Suspects

### Linear Grammars

- Productions  $U \rightarrow \sigma$ ,  $U \rightarrow V\sigma$ , and  $U \rightarrow \sigma V$

### Tree Adjoining Grammars

Tree adjoining grammars can expand at edges



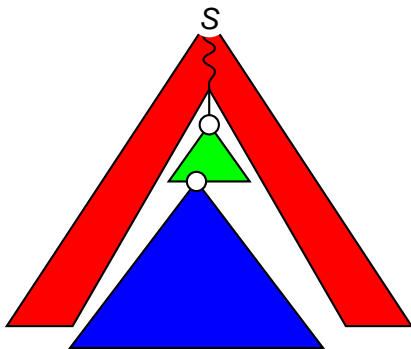
## Other Suspects

### Linear Grammars

- Productions  $U \rightarrow \sigma$ ,  $U \rightarrow V\sigma$ , and  $U \rightarrow \sigma V$

### Tree Adjoining Grammars

Tree adjoining grammars can expand at edges



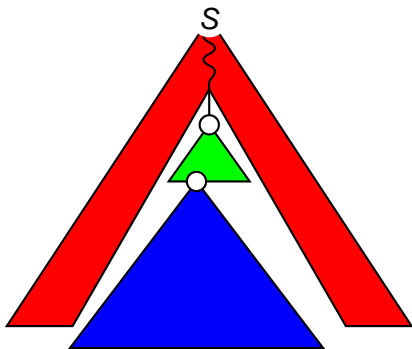
## Other Suspects

### Linear Grammars

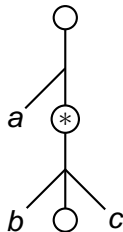
- Productions  $U \rightarrow \sigma$ ,  $U \rightarrow V\sigma$ , and  $U \rightarrow \sigma V$

### Tree Adjoining Grammars

Tree adjoining grammars can expand at edges

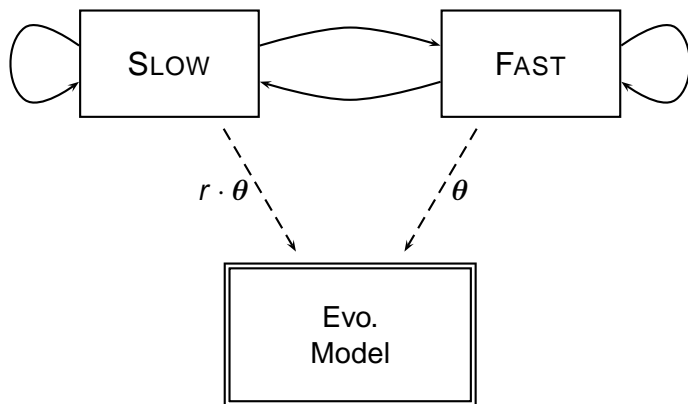


Tree for  $\{a^i b^j c^i \mid i > 0\}$



# Hierarchical Modelling – Example I

## Conserved regions in a genome



# Hierarchical Modelling – Example II

## Conserved RNA Secondary Structure

$$S \rightarrow L \mid LS, L \rightarrow dFd \mid s, F \rightarrow dFd \mid LS$$



( ( ( ( ( ( . . . . . ( ( ( ( ( ( . . . . . ) ) ) ) ) ) . . . . . )  
 GGUUA - AGGCAUUGCCACCUACUUCGUGGC A - - - UC  
 UGGUUAAGGCAUUGCCACCUACUUCGUGGC AUCUCU  
 GGUCG - - - - - GCUAGGCACAGACAAAAGC - - - -  
 GGUU - AAGGCUCUCCACA - UUCG - - UGUGGAUCUA -

Single Nuc.  
Evo. Model

Paired  
Evo. Model