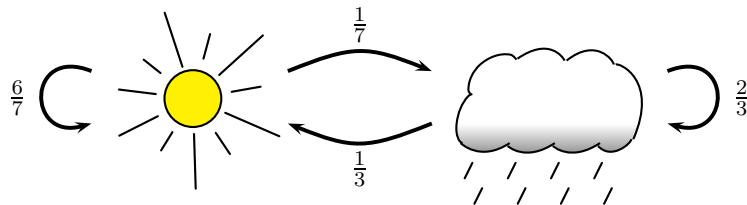
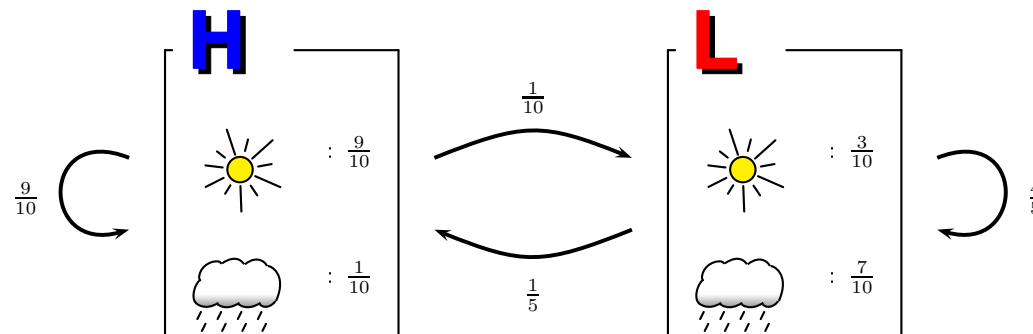


Hidden Markov Models



Markov model: Move from state to state according to probability distribution of each state and emit states visited:



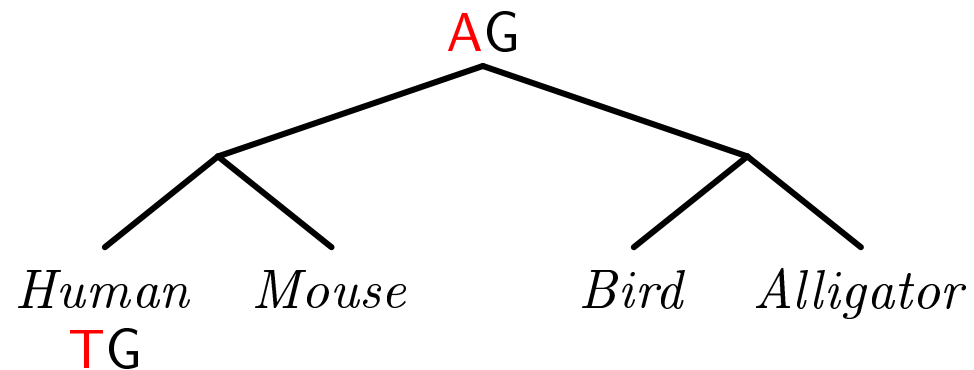
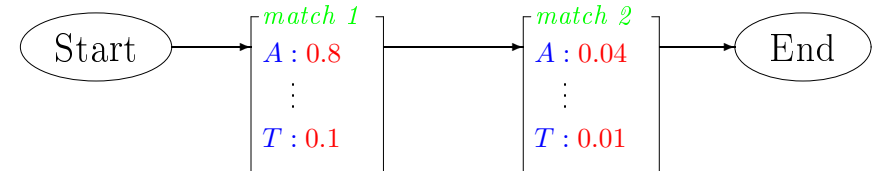
Hidden Markov model: Move from state to state in the same way, but emit a symbol according to probability distribution instead:



Note: In many situations we are only interested in finite sequences of observations – thus a hidden Markov model often have special start and end states

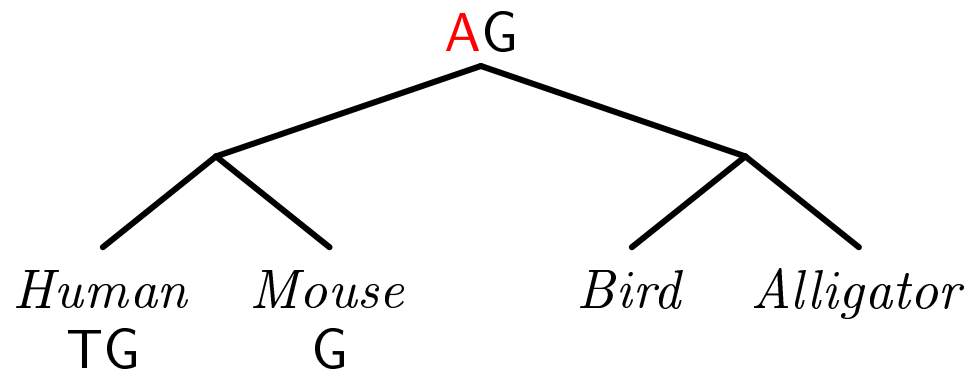
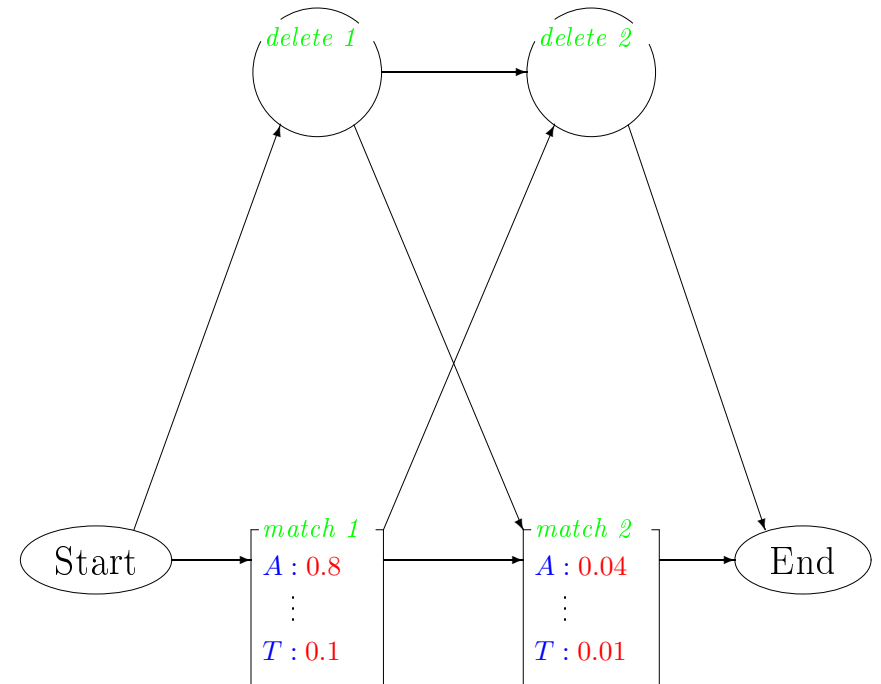
HMMs in Bioinformatics

- Sequence family modelling



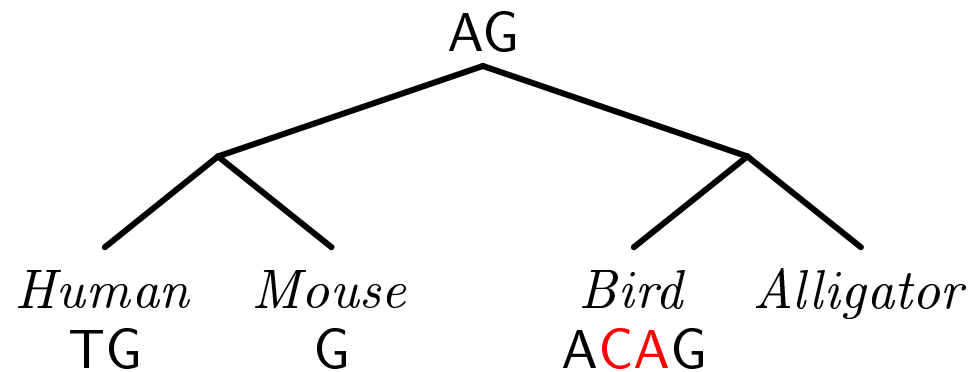
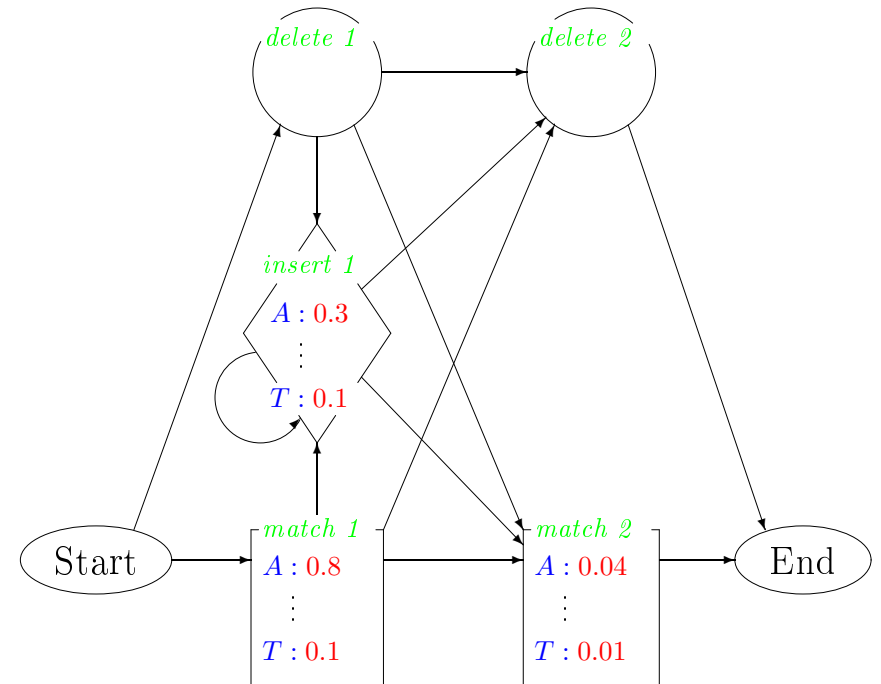
HMMs in Bioinformatics

- Sequence family modelling



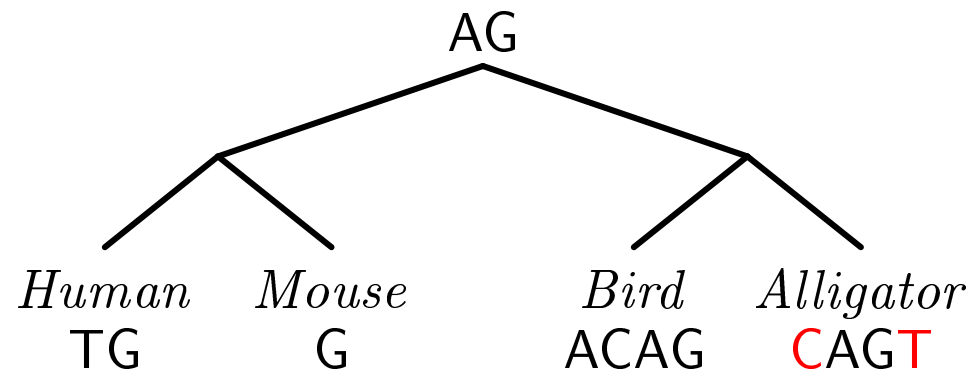
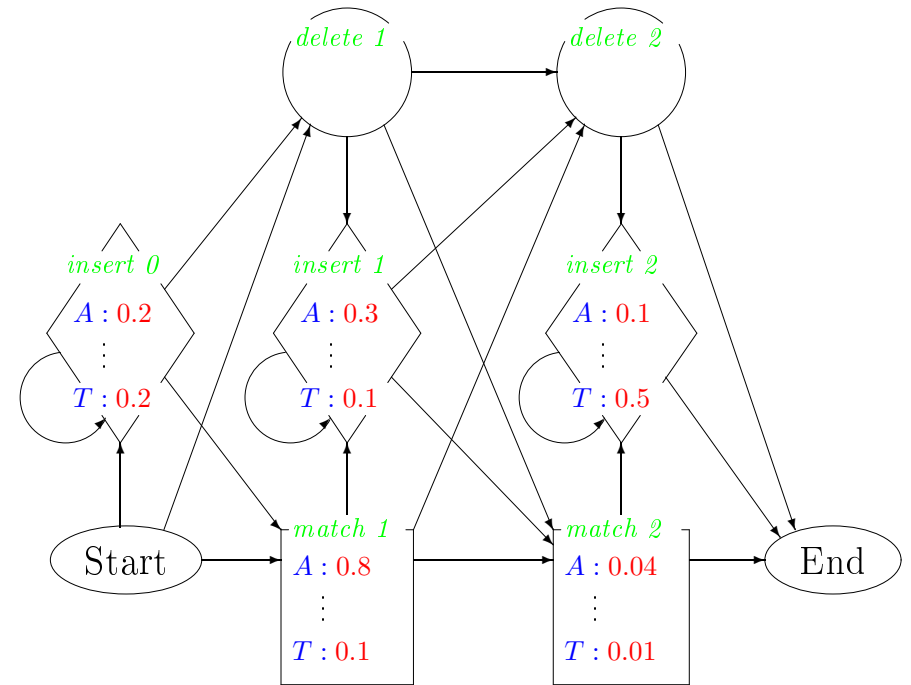
HMMs in Bioinformatics

- Sequence family modelling



HMMs in Bioinformatics

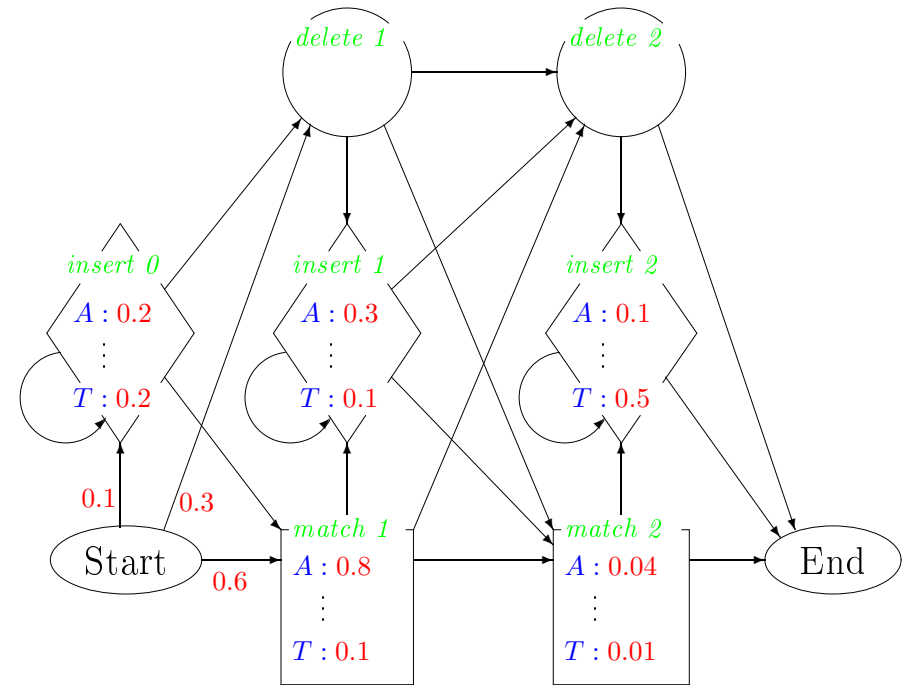
- Sequence family modelling



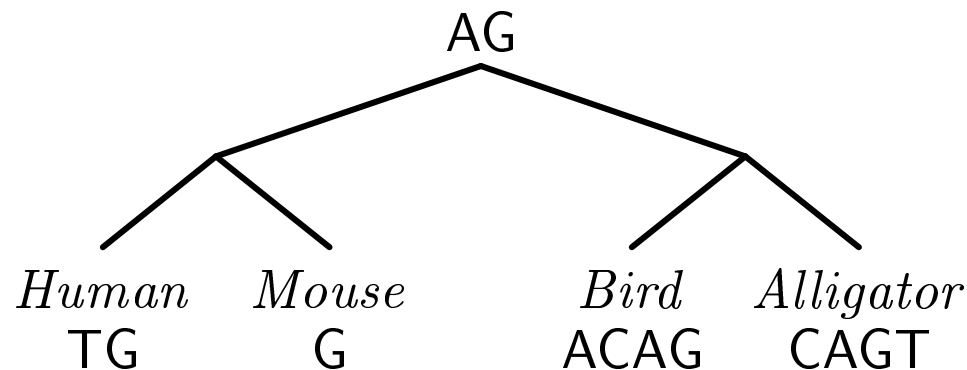
HMMs in Bioinformatics

- Sequence family modelling
- Gene finding
- Secondary structure prediction
- Transmembrane helix prediction

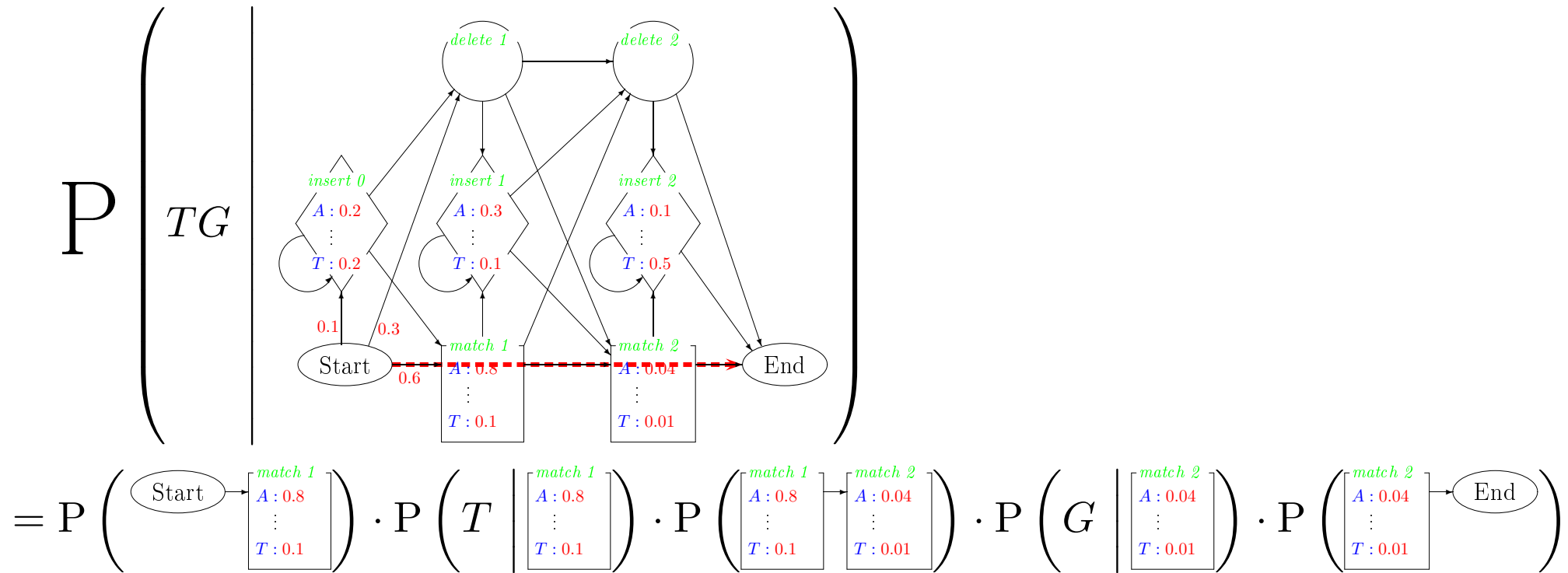
⋮



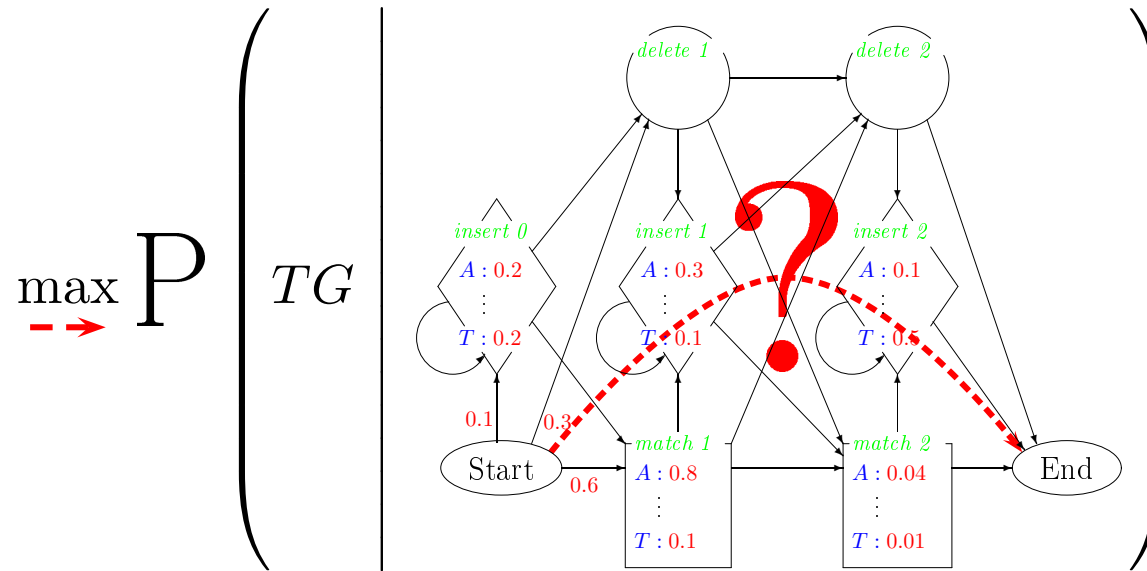
Profile HMM



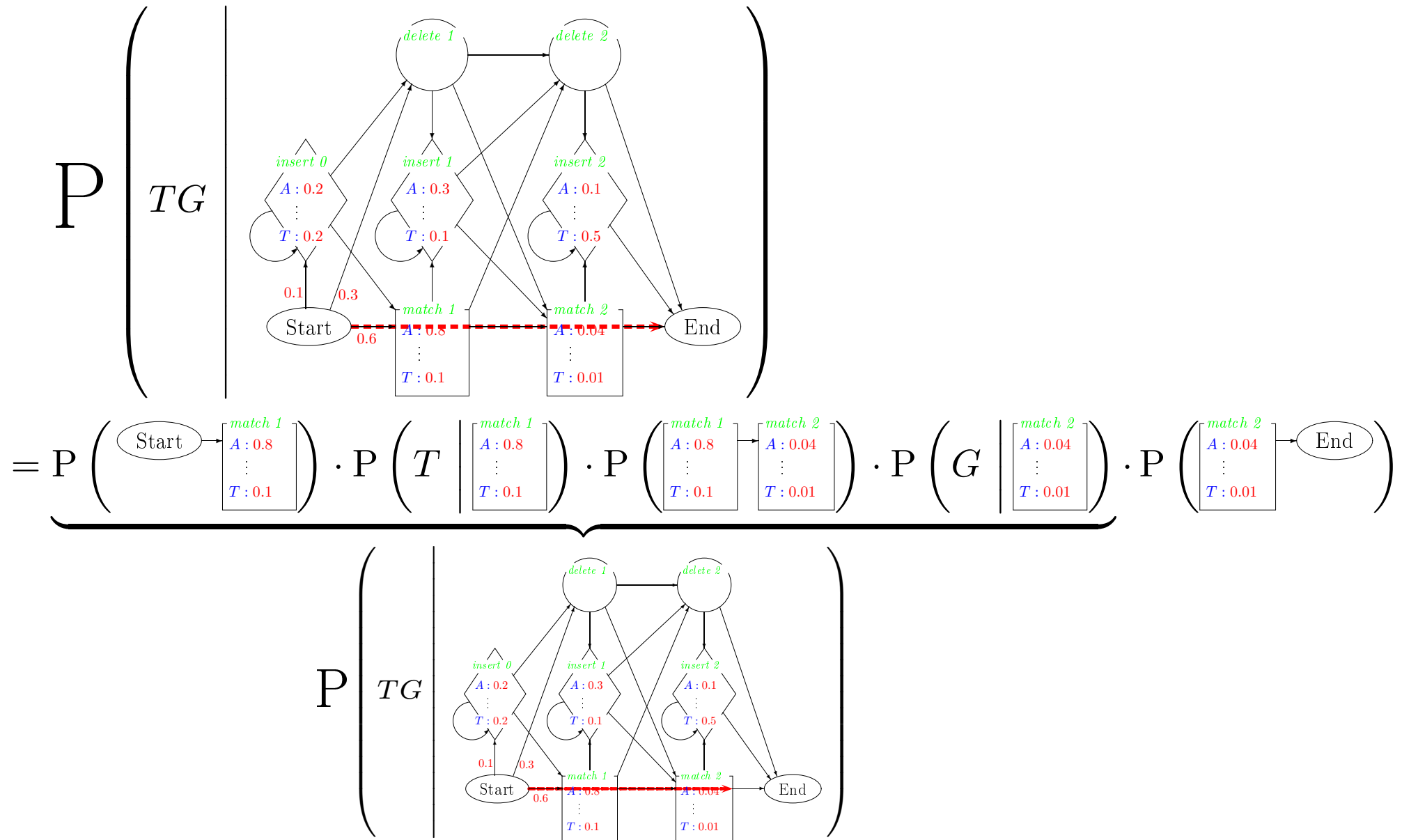
The Viterbi Algorithm



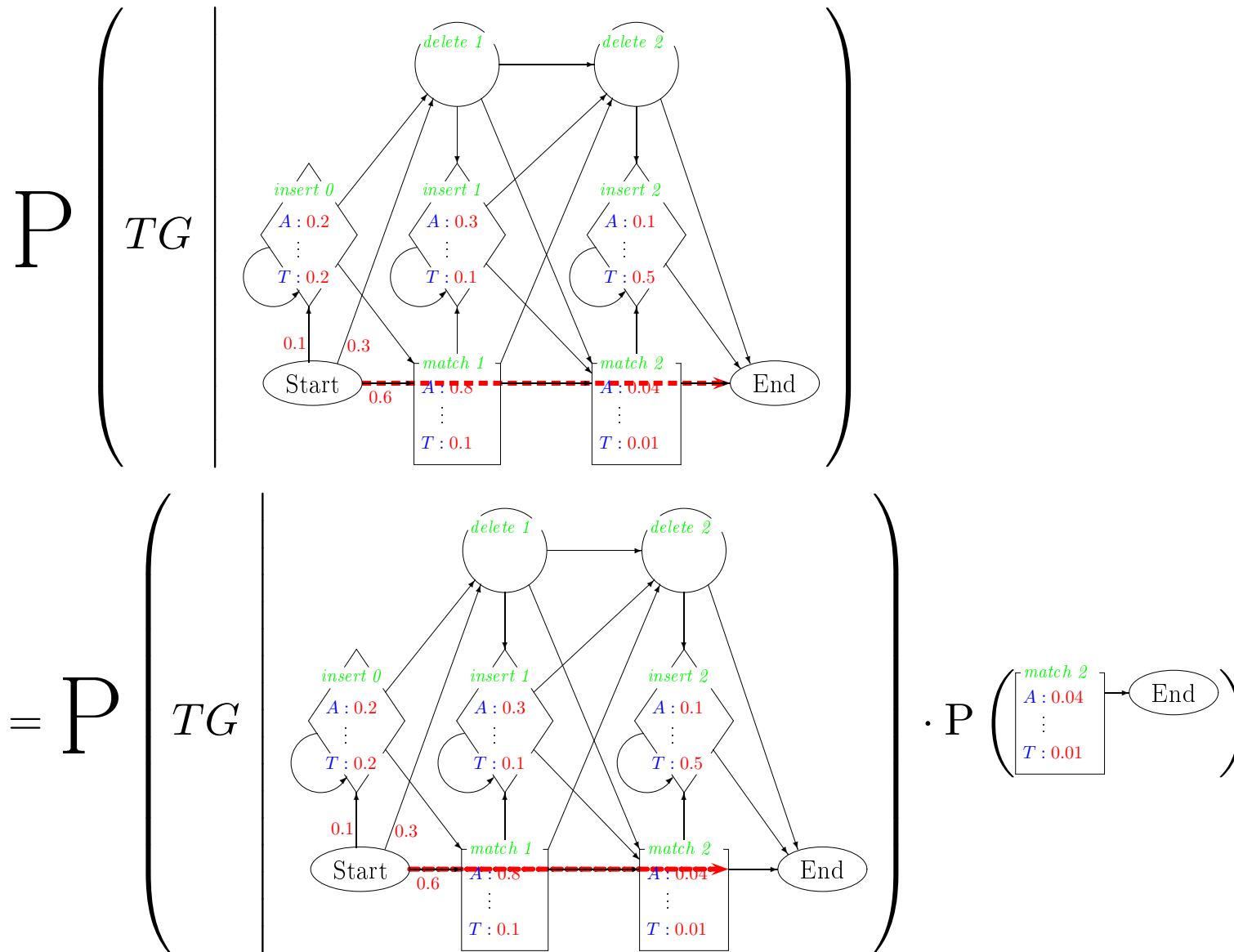
The Viterbi Algorithm



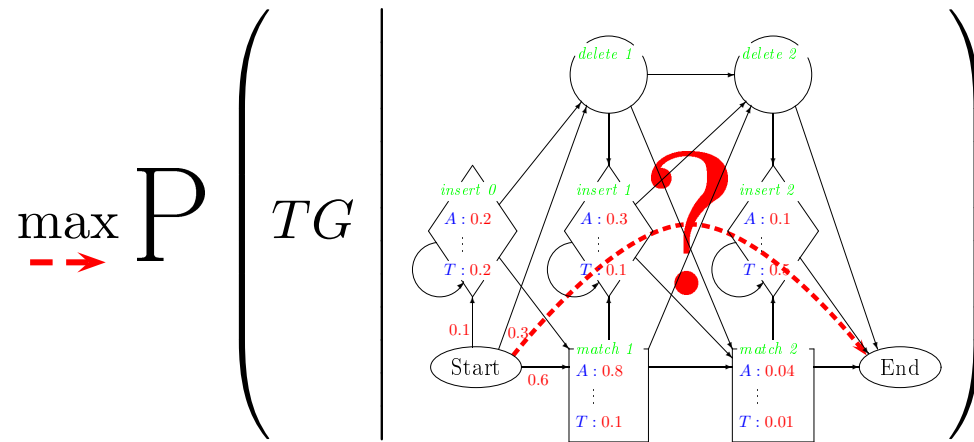
The Viterbi Algorithm



The Viterbi Algorithm

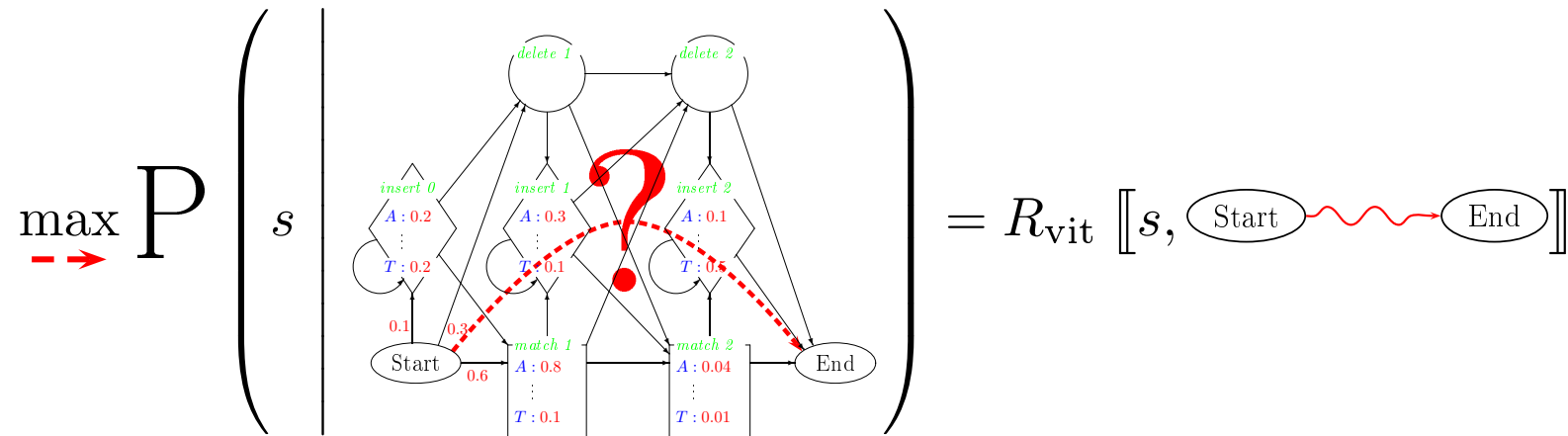


The Viterbi Algorithm



$$= \max \left\{ \begin{array}{l}
 \max_{\text{---}} P \left(TG \mid \begin{array}{c} \text{delete 1} \text{ delete 2} \\ \text{insert 0} \text{ insert 1} \text{ insert 2} \\ \text{Start} \text{ match 1} \text{ match 2} \text{ End} \end{array} \right) \cdot P \left(\begin{array}{c} \text{match 2} \\ A: 0.04 \\ \vdots \\ T: 0.01 \end{array} \mid \text{End} \right) \\
 \max_{\text{---}} P \left(TG \mid \begin{array}{c} \text{delete 1} \text{ delete 2} \\ \text{insert 0} \text{ insert 1} \text{ insert 2} \\ \text{Start} \text{ match 1} \text{ match 2} \text{ End} \end{array} \right) \cdot P \left(\begin{array}{c} \text{insert 2} \\ A: 0.1 \\ \vdots \\ T: 0.5 \end{array} \mid \text{End} \right) \\
 \max_{\text{---}} P \left(TG \mid \begin{array}{c} \text{delete 1} \text{ delete 2} \\ \text{insert 0} \text{ insert 1} \text{ insert 2} \\ \text{Start} \text{ match 1} \text{ match 2} \text{ End} \end{array} \right) \cdot P \left(\begin{array}{c} \text{delete 2} \end{array} \mid \text{End} \right)
 \end{array} \right.$$

The Viterbi Algorithm



where

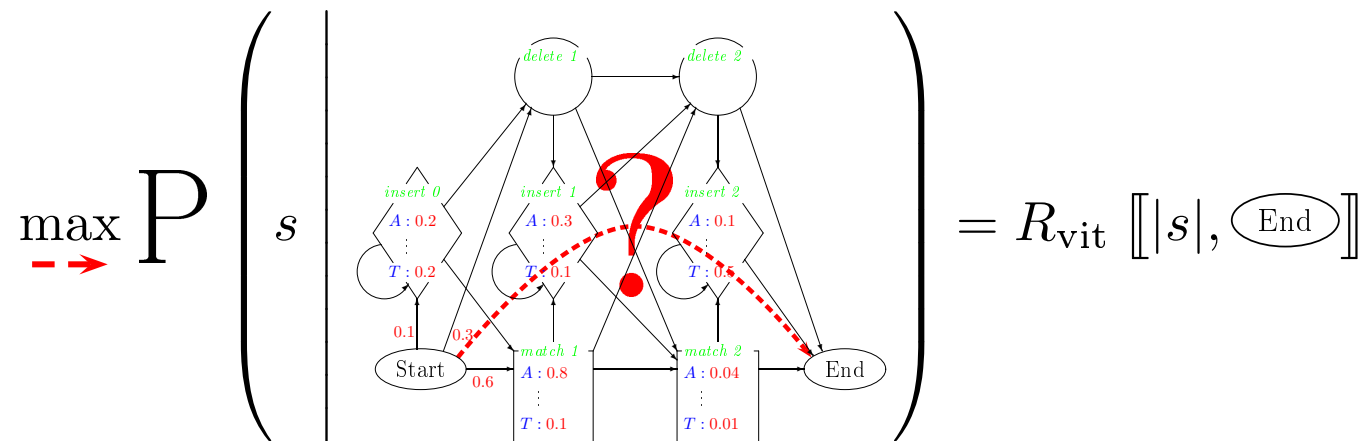
$$R_{vit} \llbracket s[1..i], \text{Start} \rightsquigarrow \text{p} \rrbracket = \max_{\text{q}} \left\{ R_{vit} \llbracket s[1..i], \text{Start} \rightsquigarrow \text{q} \rrbracket \cdot P(\text{q} \rightarrow \text{p}) \right\}$$

$$R_{vit} \llbracket s[1..i], \text{Start} \rightsquigarrow \text{p} \rrbracket = \max_{\text{q}} \left\{ R_{vit} \llbracket s[1..i-1], \text{Start} \rightsquigarrow \text{q} \rrbracket \cdot P(\text{q} \rightarrow \text{p}) \right\} \cdot P(s[i] | \text{q})$$

Legend:

● Non-silent state
 ● Silent state
 ● Any state

The Viterbi Algorithm



Legend:

- Non-silent state
- Silent state
- Any state

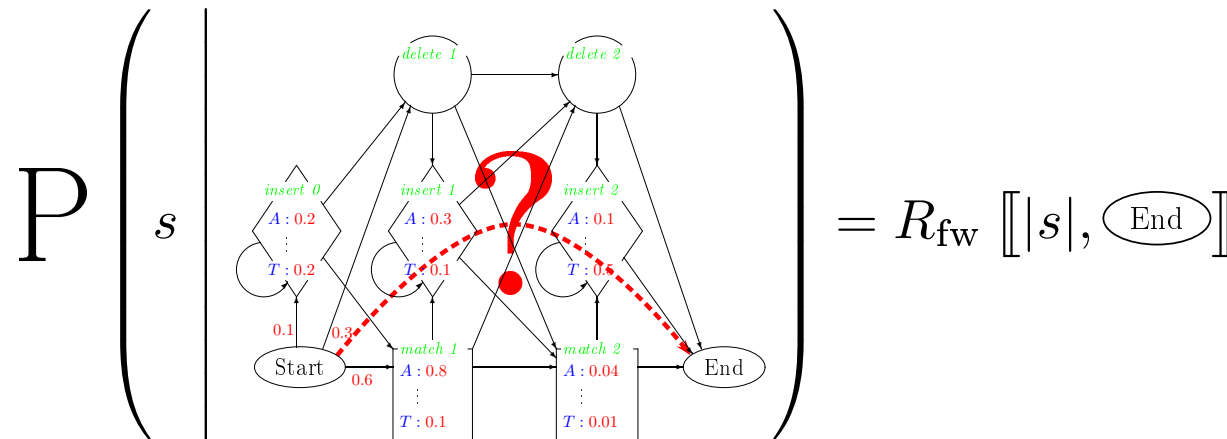
where

$$R_{vit} [i, \text{p}] = \max_{\text{q}} \{ R_{vit} [i, \text{q}] \cdot P(\text{q} \rightarrow \text{p}) \}$$

$$R_{vit} [i, \text{p}] = \max_{\text{q}} \{ R_{vit} [i - 1, \text{q}] \cdot P(\text{q} \rightarrow \text{p}) \} \cdot P(s[i] | \text{q})$$

Recursions yield Viterbi algorithm that compute maximum likelihood explanation in time $O(|s| \cdot \text{\#transitions})$ and space $O(|s| \cdot \text{\#states})$

The Forward Algorithm



Legend:

- Non-silent state
- Silent state
- Any state

where

$$R_{\text{fw}} \left[i, \text{p} \right] = \sum_{\text{q}} \left\{ R_{\text{fw}} \left[i, \text{q} \right] \cdot P \left(\text{q} \rightarrow \text{p} \right) \right\}$$

$$R_{\text{fw}} \left[i, \text{p} \right] = \sum_{\text{q}} \left\{ R_{\text{fw}} \left[i - 1, \text{q} \right] \cdot P \left(\text{q} \rightarrow \text{p} \right) \right\} \cdot P \left(s[i] \mid \text{q} \right)$$

Recursions yield forward algorithm that compute sequence probability in time $O(|s| \cdot \# \text{transitions})$ and space $O(|s| \cdot \# \text{states})$

Stochastic Context-Free Grammars

$$S_X \xrightarrow{\frac{1}{4}} X \text{ ✈️ } Y \quad S_Y \text{ ✈️ } X$$

$$\xrightarrow{\frac{1}{6}} X \text{ 🚂 } Y \quad S_Y \text{ 🚂 } X$$

$$\xrightarrow{\frac{1}{3}} C_{X,X}$$

$$\xrightarrow{\frac{1}{6}} S_X S_X$$

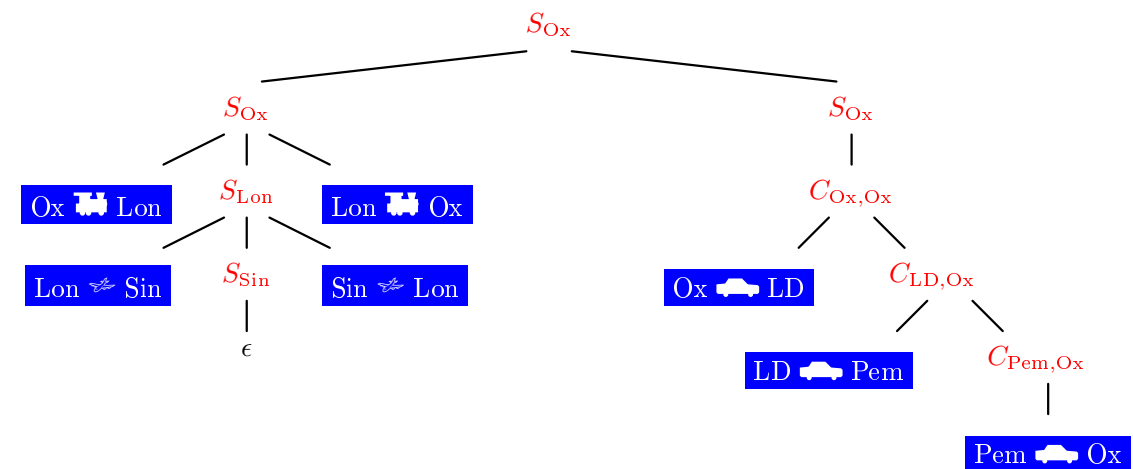
$$\xrightarrow{\frac{1}{12}} \epsilon$$

$$C_{X,Y} \xrightarrow{\frac{1}{2}} X \text{ 🚗 } Y$$

$$\xrightarrow{\frac{1}{3}} X \text{ 🚗 } Z \quad C_{Z,Y}$$

$$\xrightarrow{\frac{1}{6}} S_X C_{X,Y}$$

Replace non-terminals according to probability distribution until only terminal symbols are left. Extends on the capabilities of an HMM by allowing the modelling of nested dependencies:



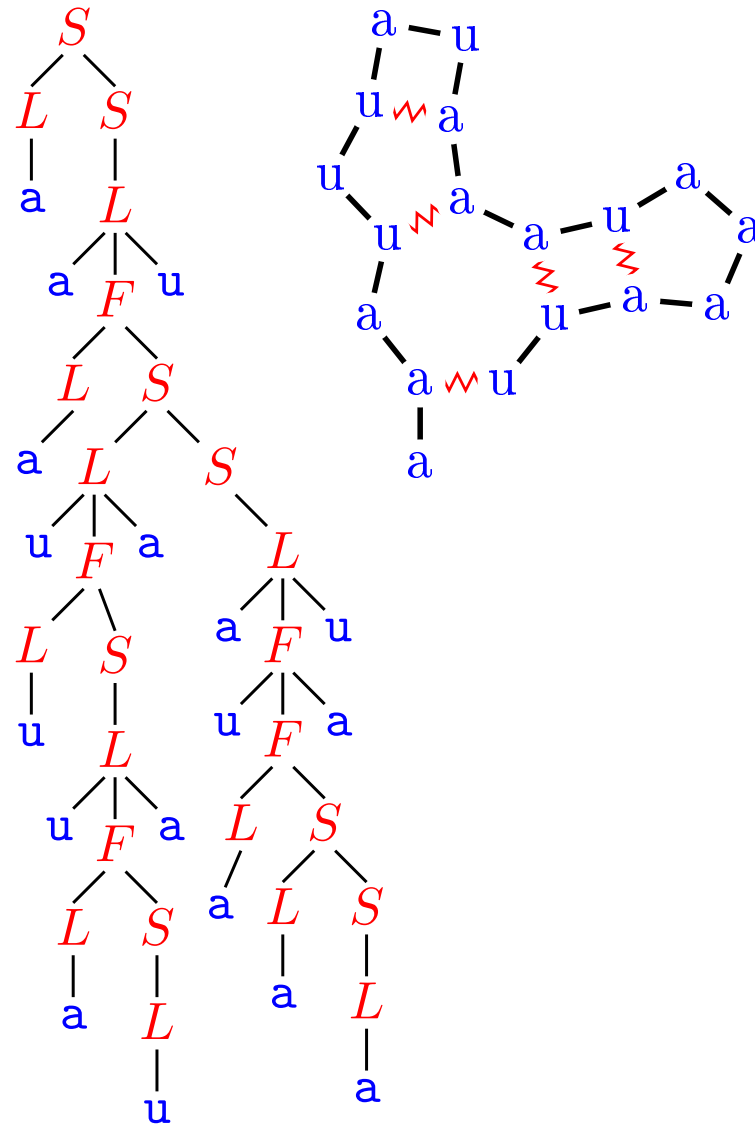
SCFGs in Bioinformatics

- RNA secondary structure prediction/modelling

$$S \rightarrow LS \mid L$$

$$F \rightarrow aFu \mid uFa \mid LS$$

$$L \rightarrow a \mid u \mid aFu \mid uFa$$



Chomsky Normal Form

In a grammar on Chomsky normal form, non-terminals are replaced by either

- the empty string ($X \rightarrow \epsilon$),
- a terminal symbol ($X \rightarrow \sigma$),
- or two non-terminals ($X \rightarrow YZ$).

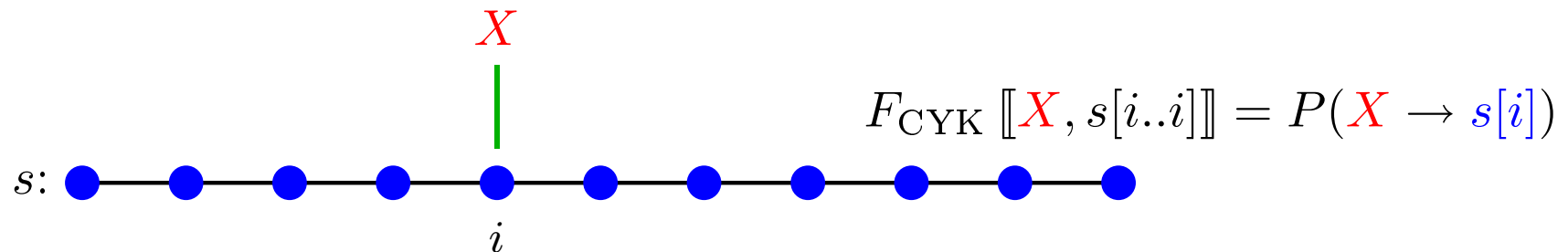
All (stochastic) context-free grammars can be converted to Chomsky normal form:

$$\begin{array}{l} S \rightarrow LS \mid L \\ F \rightarrow aFu \mid uFa \mid LS \\ L \rightarrow a \mid u \mid aFu \mid uFa \end{array} \Rightarrow \begin{array}{l} S \rightarrow LS \mid LE \\ F \rightarrow AF_1 \mid UF_2 \mid LS \\ L \rightarrow a \mid u \mid AF_1 \mid UF_2 \\ E \rightarrow \epsilon \end{array} \quad \begin{array}{l} A \rightarrow a \\ U \rightarrow u \\ F_1 \rightarrow FU \\ F_2 \rightarrow FA \end{array}$$

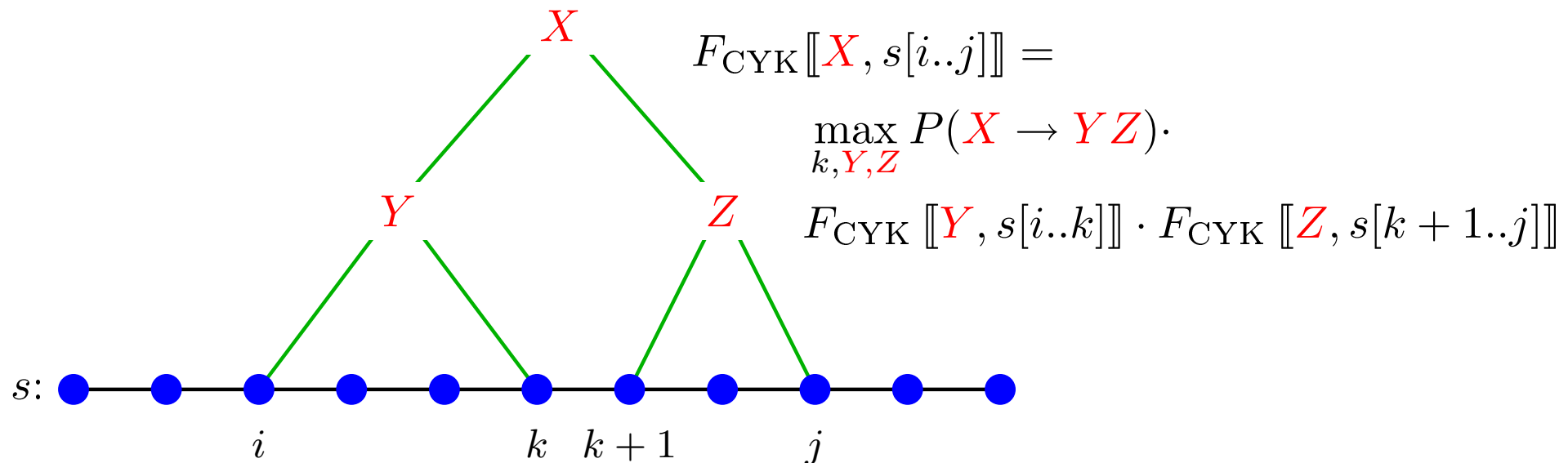
CYK Algorithm

Assume grammar G is CNF (and only ϵ replacement is $S \rightarrow \epsilon$).

Basis: Find probabilities that non-terminals generate single symbols of the string.



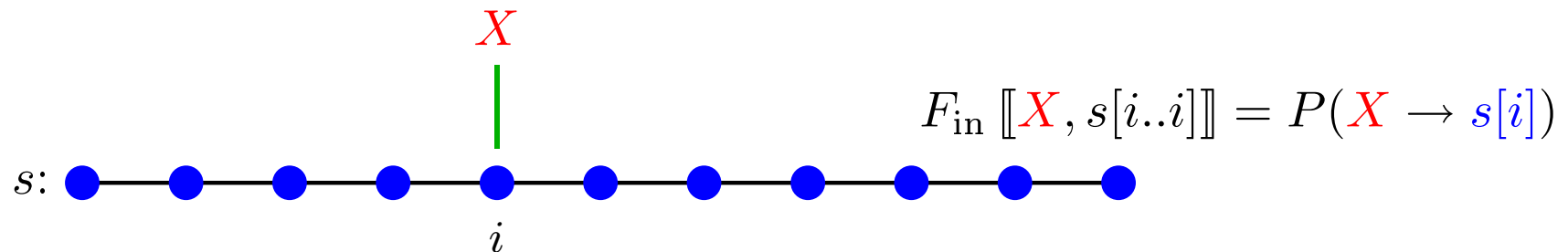
Recursion: Combine parses of neighbouring substrings.



Inside Algorithm

Assume grammar G is CNF (and only ϵ replacement is $S \rightarrow \epsilon$).

Basis: Find probabilities that non-terminals generate single symbols of the string.



Recursion: Combine parses of neighbouring substrings.

