

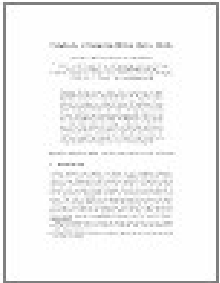
# Comparison of Hidden Markov Models

Christian Nørgaard Storm Pedersen  
Bioinformatics Research Center (BiRC)  
Department of Computer Science

# Comparison of Hidden Markov Models



**Metrics and similarity measures for hidden Markov models**  
Rune B. Lyngsø, Christian N. S. Pedersen, Henrik Nielsen  
In Proceedings of ISMB'99

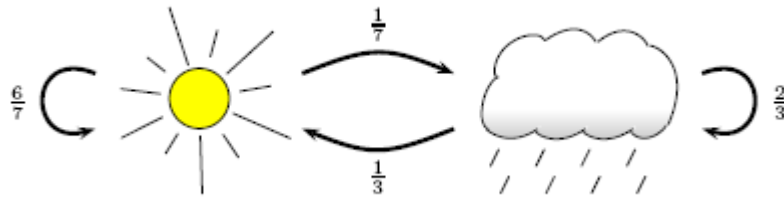


**Complexity of comparing hidden Markov models**  
Rune B. Lyngsø, Christian N. S. Pedersen  
In Proceedings of ISAAC'01

# Markov models

**Markov model:** Moves from state to state according to a probability distribution of each state (transition probabilities) and emits the states visited ...

Model  $M$ :

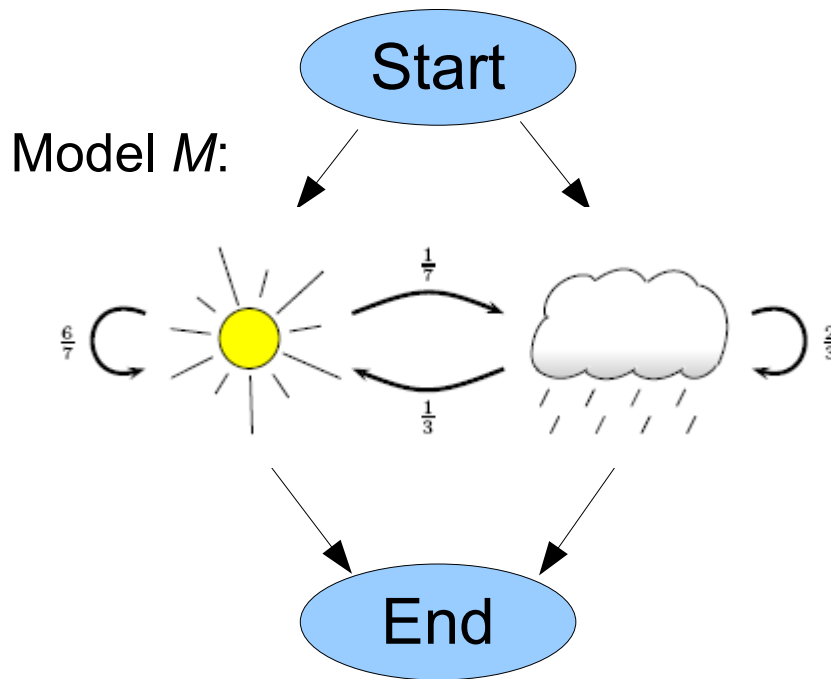


A **run** emits a sequence of states:



# Markov models

**Markov model:** Moves from state to state according to a probability distribution of each state (transition probabilities) and emits the states visited ...



A **run** emits a sequence of states:

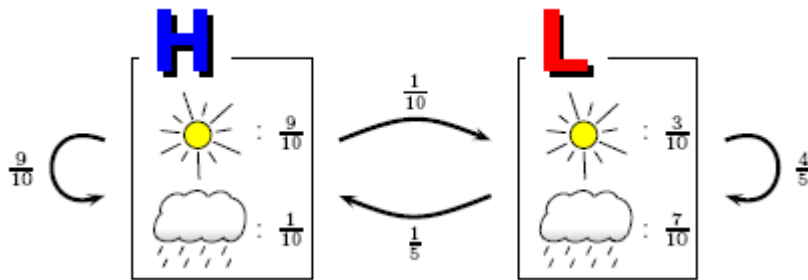


Special **Start**- and **End**-states can be added to generate finite output

# Hidden Markov models

**Hidden Markov model:** Moves from state to state in the same way as a Markov model, but **emits a symbol** (from a finite alphabet of the model) from each state visited (except silent states) **according to a probability distribution of the state** (emission probabilities) ...

Model  $M$ :



A **run** follows a sequence of states:

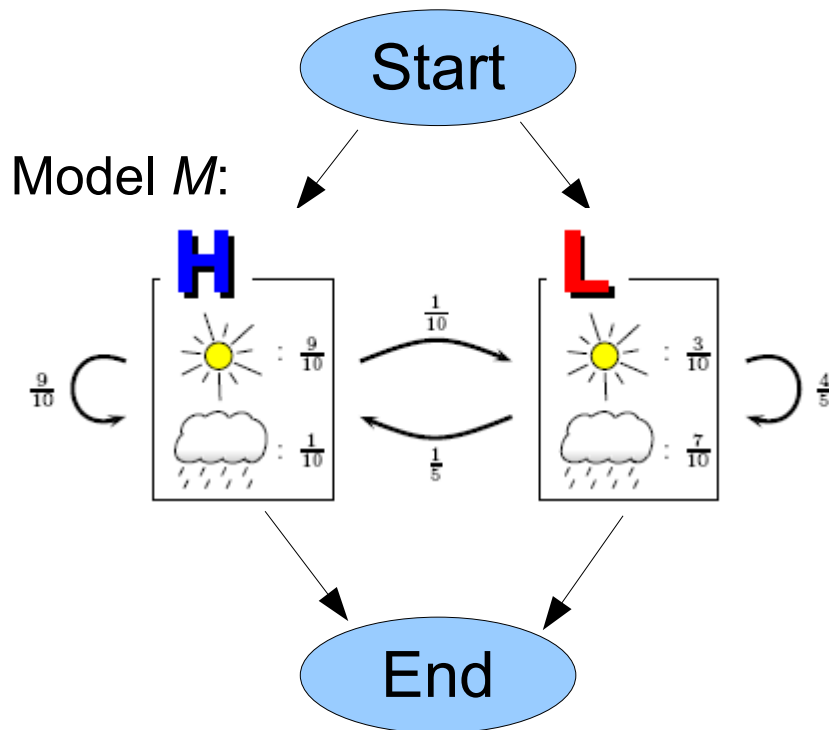
H H L L H

And **emits** a sequence of symbols:



# Hidden Markov models

**Hidden Markov model:** Moves from state to state in the same way as a Markov model, but **emits a symbol** (from a finite alphabet of the model) from each state visited (except silent states) **according to a probability distribution of the state** (emission probabilities) ...



A **run** follows a sequence of states:

H H L L H

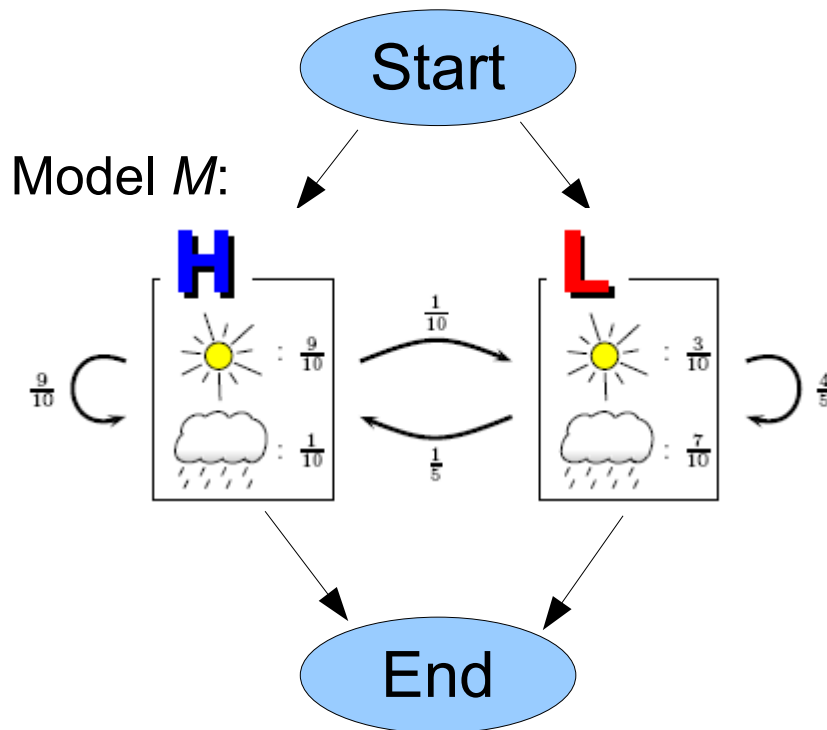
And **emits** a sequence of symbols:



Special **Start**- and **End**-states can be added to generate finite output

# Hidden Markov models

**Hidden Markov model:** Moves from state to state in the same way as a Markov model, but **emits a symbol** (from a finite alphabet of the model) from each state visited (except silent states) **according to a probability distribution of the state** (emission probabilities) ...



A **run** follows a sequence of states:

H H L L H

And **emits** a sequence of symbols:



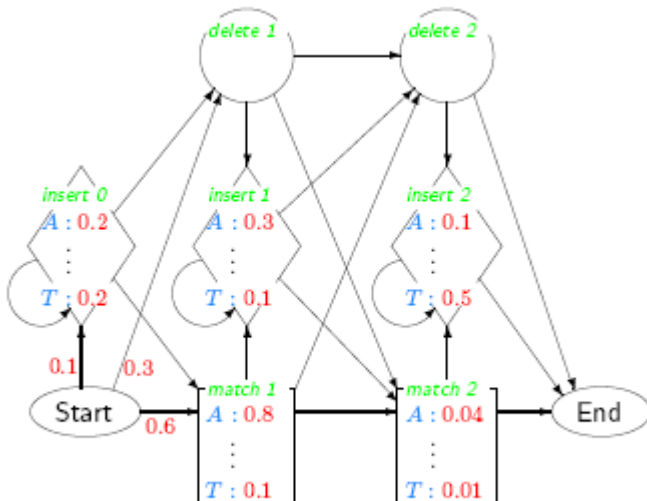
Special **Start**- and **End**-states can be added to generate finite output

# Hidden Markov models

Developed in speech recognition in the late 1960s ...

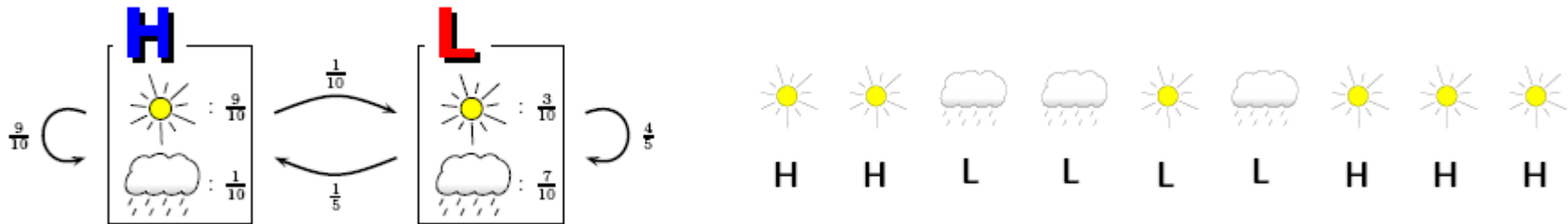
A HMM  $M$  (with start- and end-states) defines a regular language  $L_M$  of all the strings that  $M$  can generate, i.e.

$$L_M = \{S \mid P_M(S) > 0\}$$



A run in model  $M$  follows a Markovian path of states and generates a string  $S$  over a finite alphabet with probability  $P_M(S)$

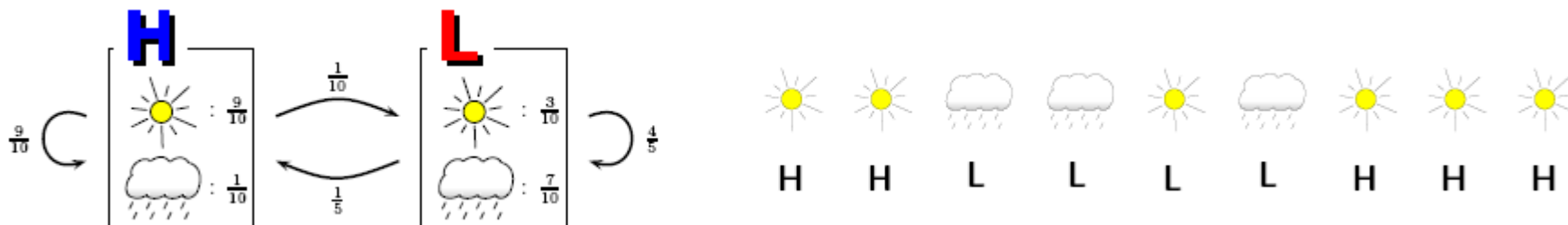
# Typical HMM problems



**Annotation:** Given a model  $M$  and an observed string  $S$ , what is the most probable path through  $M$  that generates/outputs  $S$

*The Viterbi algorithm. Running time  $O(|M||S|)$*

# Typical HMM problems



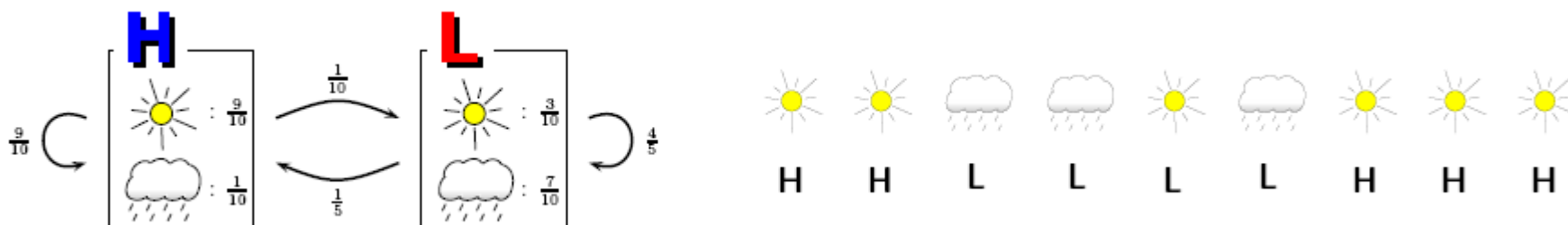
**Annotation:** Given a model  $M$  and an observed string  $S$ , what is the most probable path through  $M$  that generates/outputs  $S$

*The Viterbi algorithm. Running time  $O(|M||S|)$*

**Classification:** Given a model  $M$  and an observed string  $S$ , what is the total probability  $P_M(S)$  of  $M$  generating  $S$

*The forward algorithm. Running time  $O(|M||S|)$*

# Typical HMM problems



**Annotation:** Given a model  $M$  and an observed string  $S$ , what is the most probable path through  $M$  that generates/outputs  $S$

*The Viterbi algorithm. Running time  $O(|M||S|)$*

**Classification:** Given a model  $M$  and an observed string  $S$ , what is the total probability  $P_M(S)$  of  $M$  generating  $S$

*The forward algorithm. Running time  $O(|M||S|)$*

**Training:** Given a set of training strings and a model structure, find transition and emission probabilities that make the training set probable

*Hard! Use Baum-Welch or Viterbi iterative training methods ...*

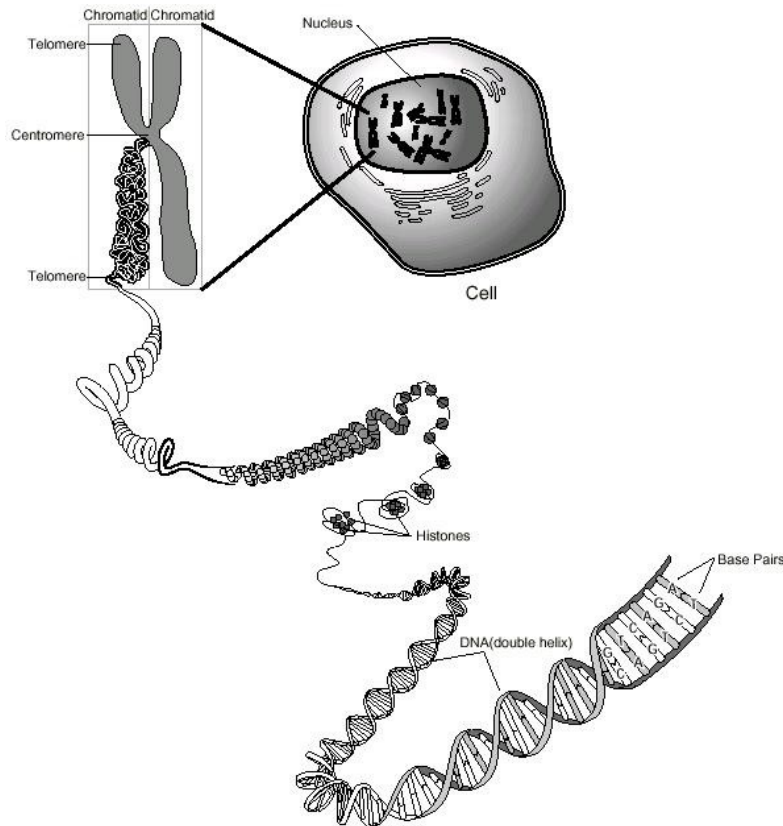
# DNA sequences

## The human genome:

About  $100 \cdot 10^{15}$  cells, each containing 23 pairs of chromosomes, DNA molecules, storing genetic information ...

```
GGCCTAAAGGCGCCGGTCTTTCGTACCCCAAATCTC  
GGCATTTTAAGATAAGTGAGTGTTGCGTTACACTAGC  
GATCTACCGCGTCTTATACTTAAGCGTATGCCAGAT  
CTGACTAATCGTGCCCCCGGATTAGACGGGCTTGATG  
GGAAAGAACAGCTCGTCTGTTTACGTATAAACAGAAT  
CGCCTGGGTTA...
```

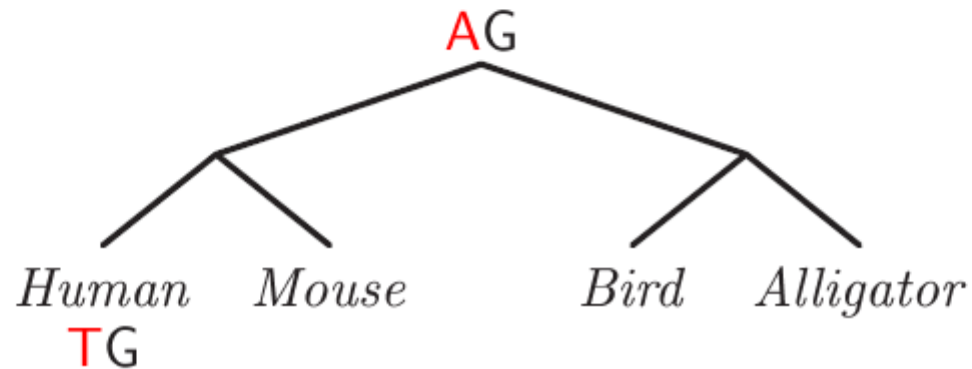
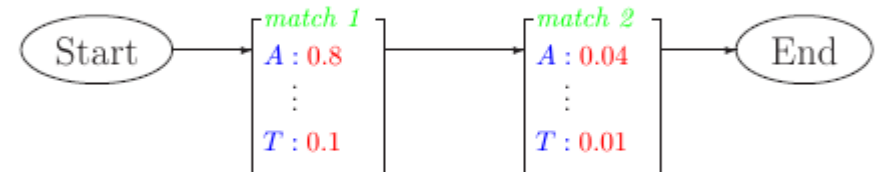
**In total: 3.100.000.000 ACGT's**



Bioinformatics is about developing and applying models, methods, and tools, for accessing and analyzing biological data, e.g. sequences ...

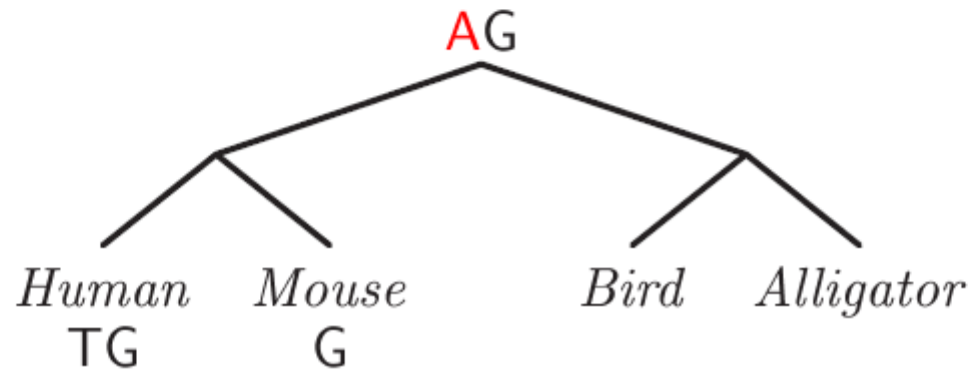
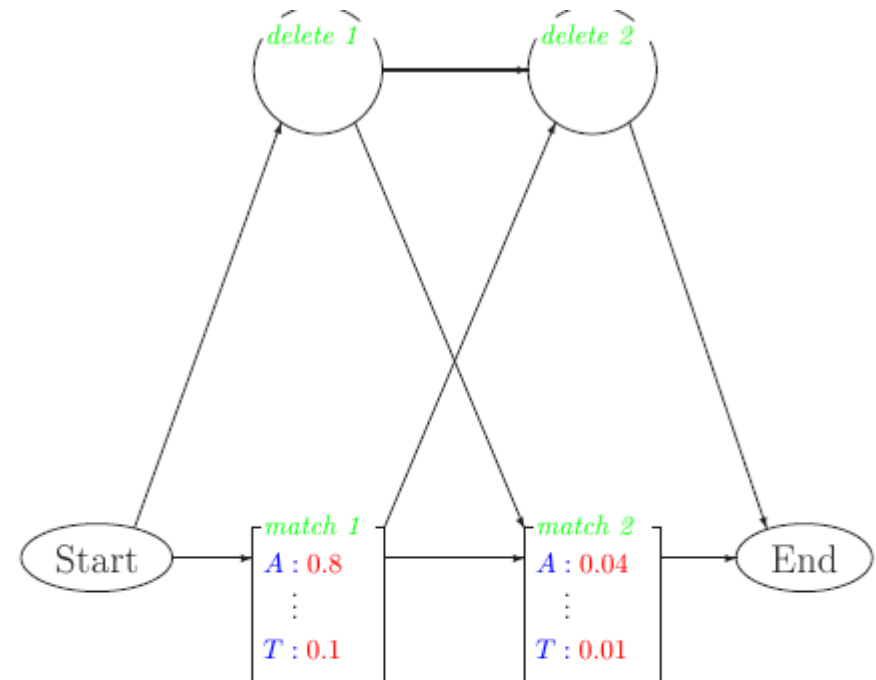
# HMMs in bioinformatics

Sequence family modeling ...



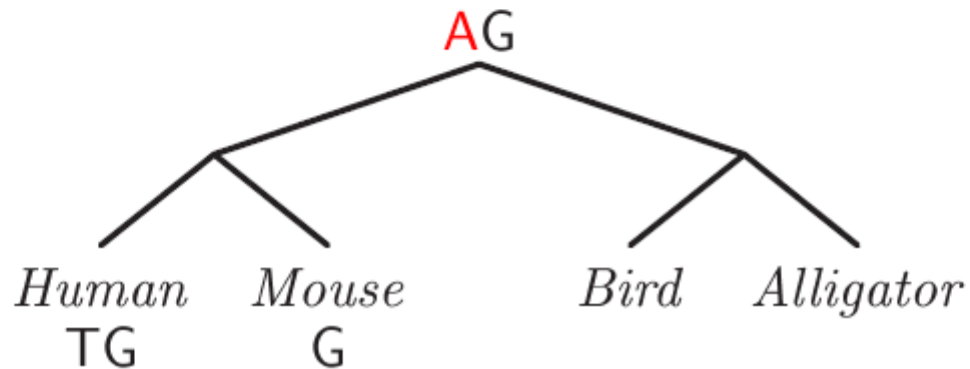
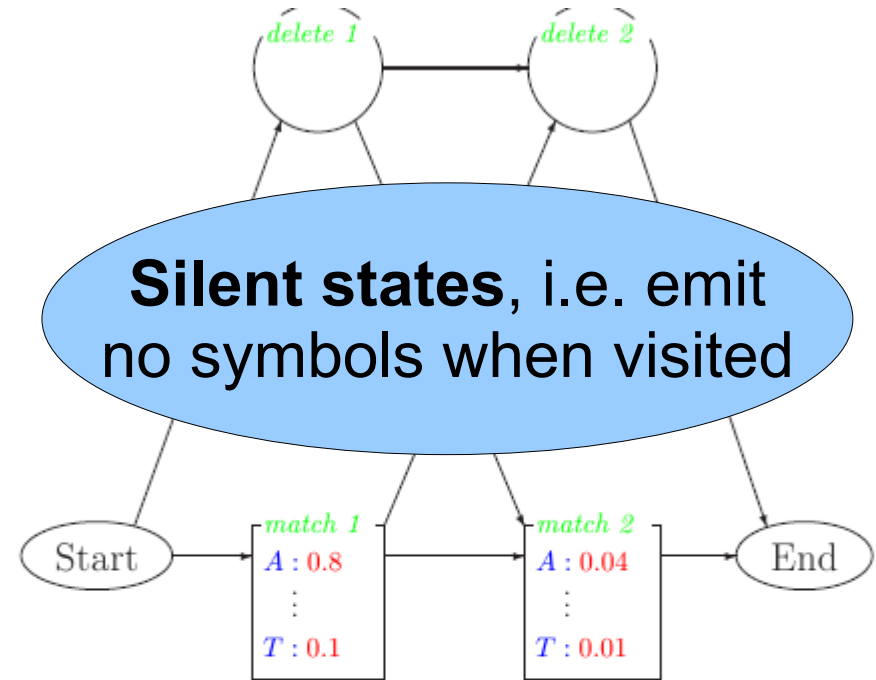
# HMMs in bioinformatics

Sequence family modeling ...



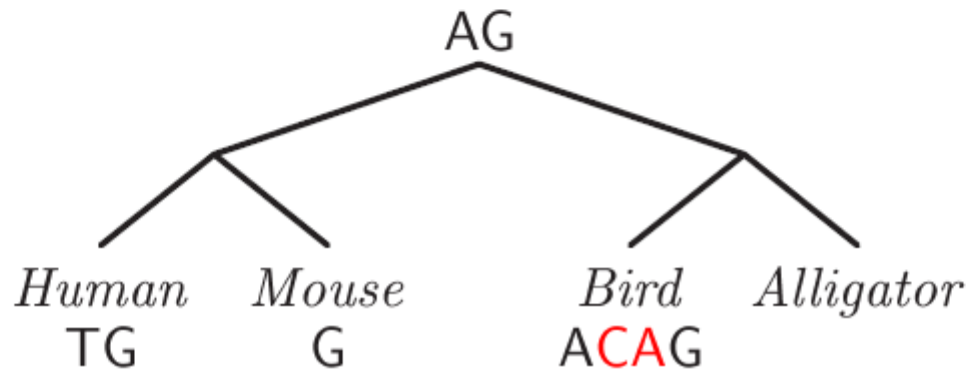
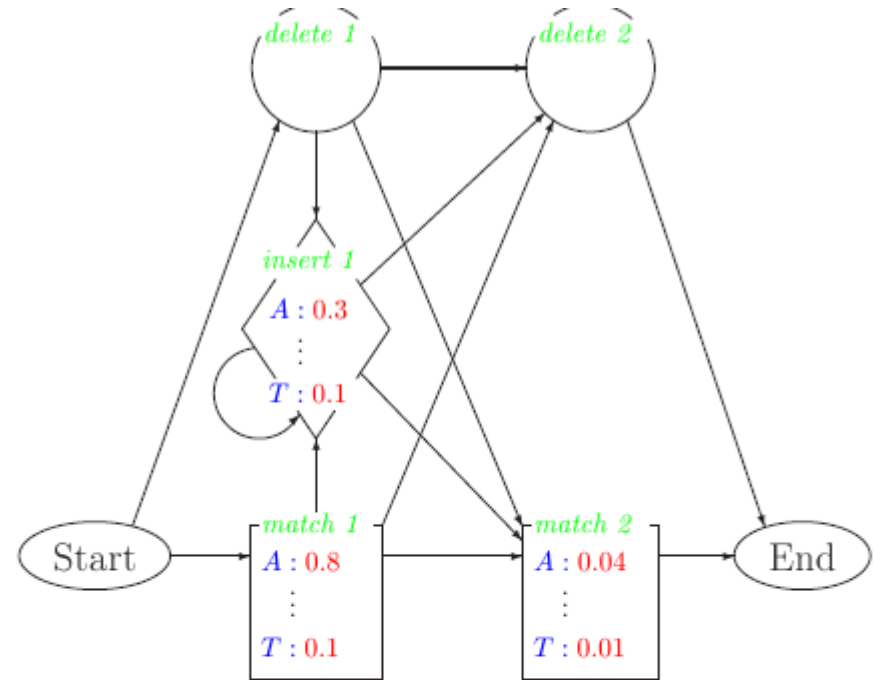
# HMMs in bioinformatics

Sequence family modeling ...



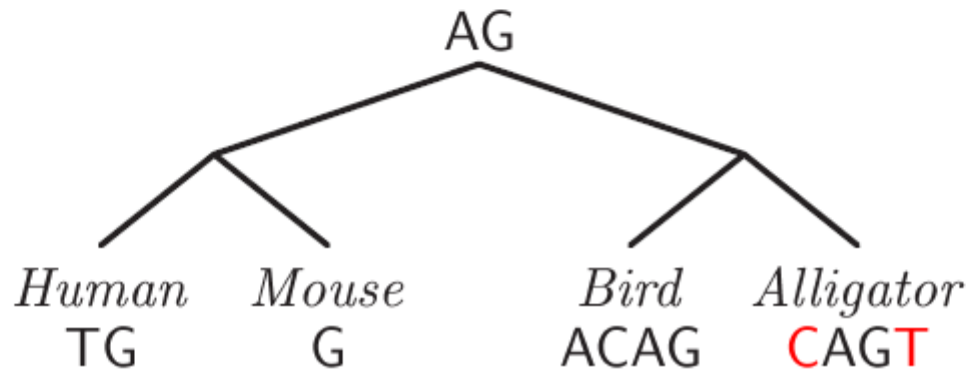
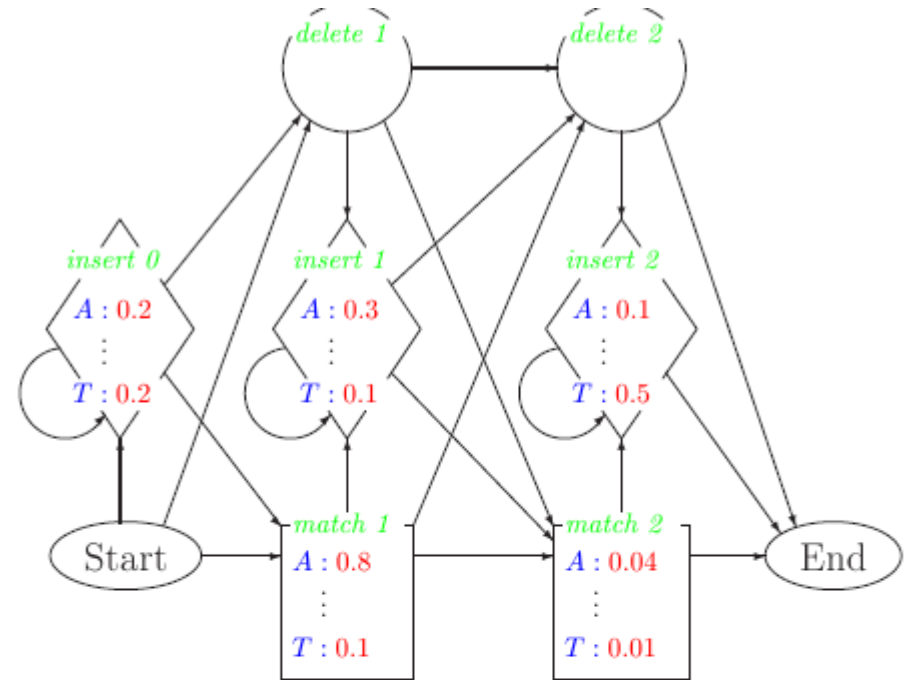
# HMMs in bioinformatics

Sequence family modeling ...



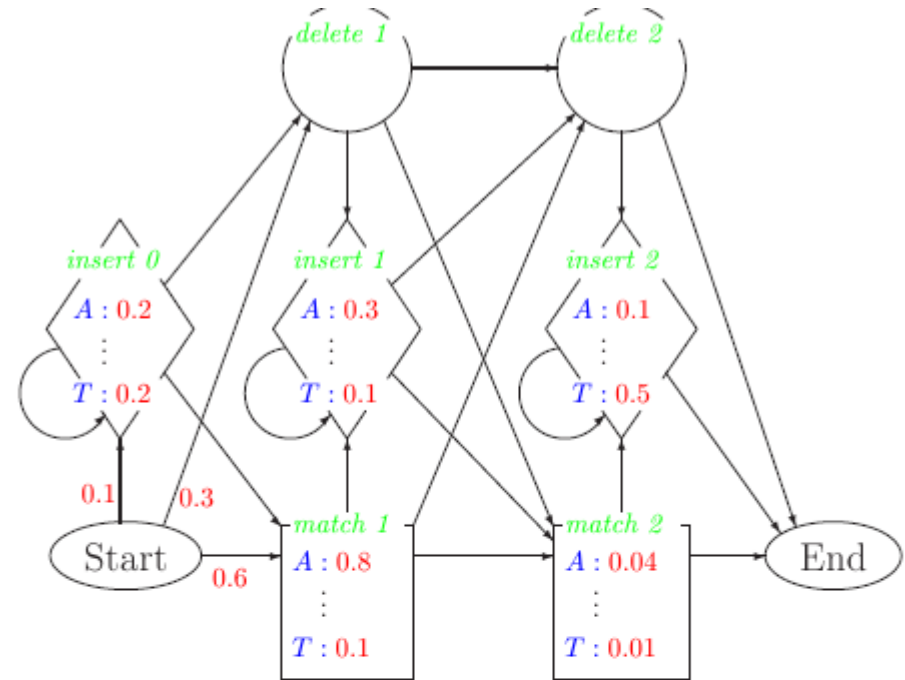
# HMMs in bioinformatics

Sequence family modeling ...

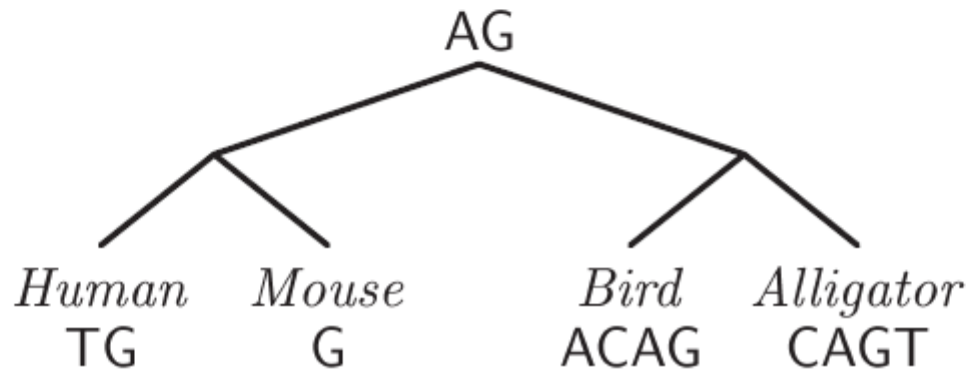


# HMMs in bioinformatics

Sequence family modeling  
Gene finding  
RNA 2<sup>nd</sup> structure prediction  
Transmembrane helix prediction  
...

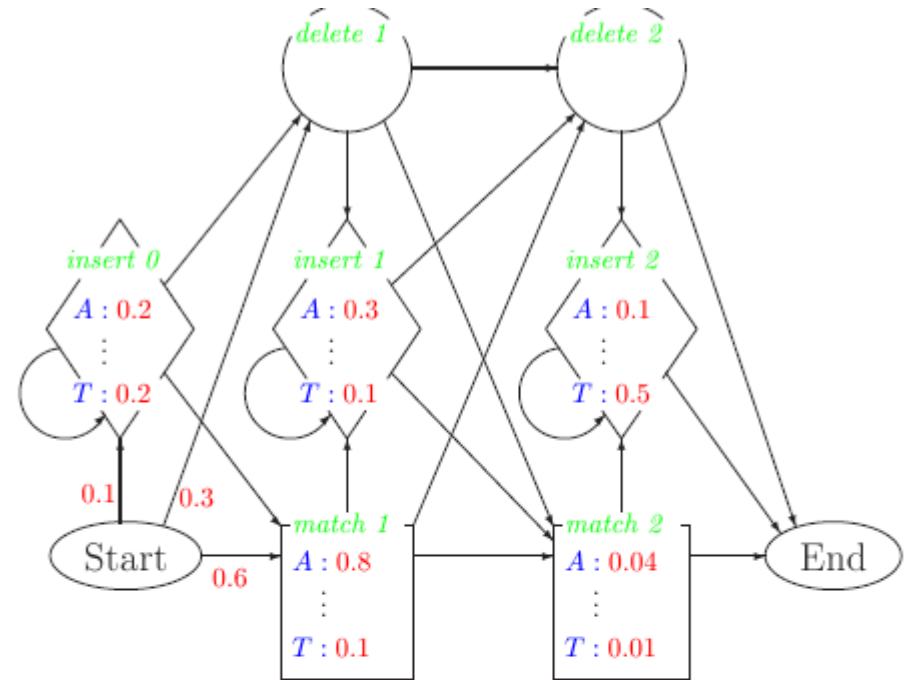


A profile HMM [Krogh et al, 1994]



# HMMs in bioinformatics

Sequence family modeling  
Gene finding  
RNA 2<sup>nd</sup> structure prediction  
Transmembrane helix prediction  
...



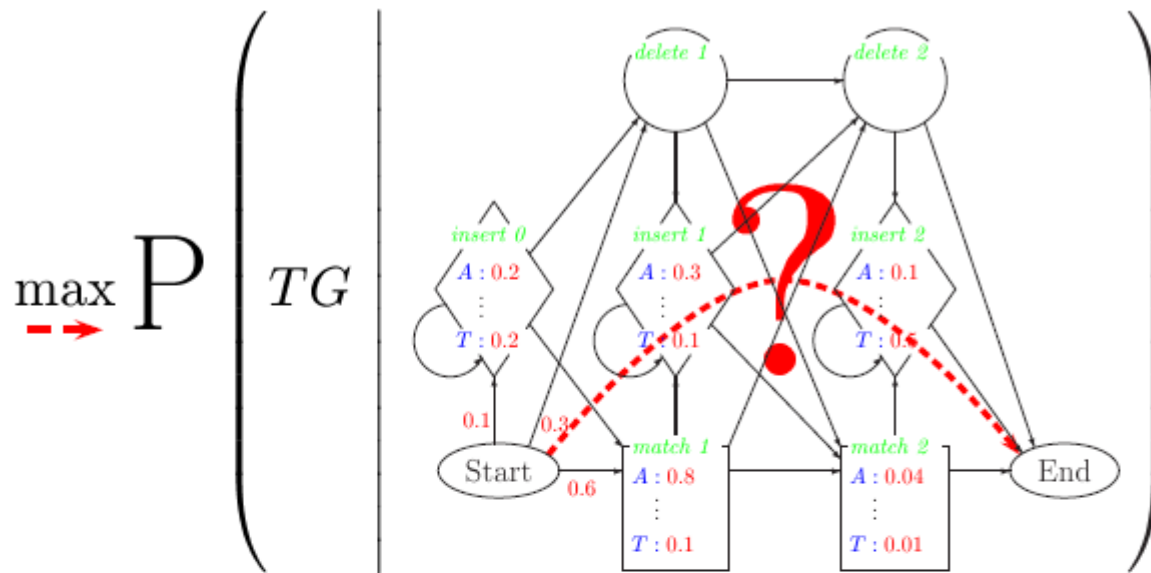
A profile HMM [Krogh et al, 1994]

**Classification:** Given a string  $S$ , find the probability that it is related to the modeled sequence family ...

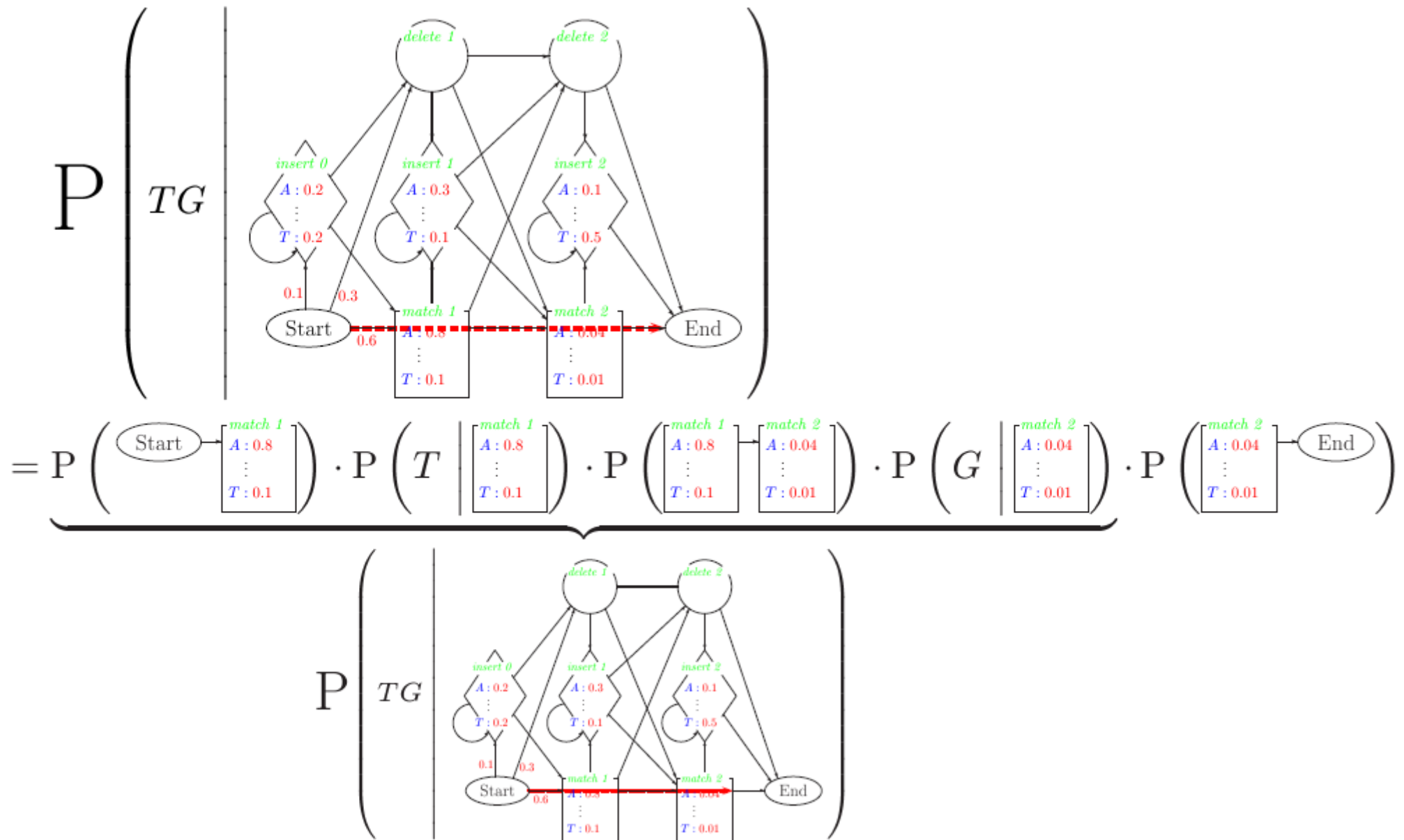
**Annotation:** Given a string  $S$ , find (the probability of) the most likely way it relates to the other sequence family members ...



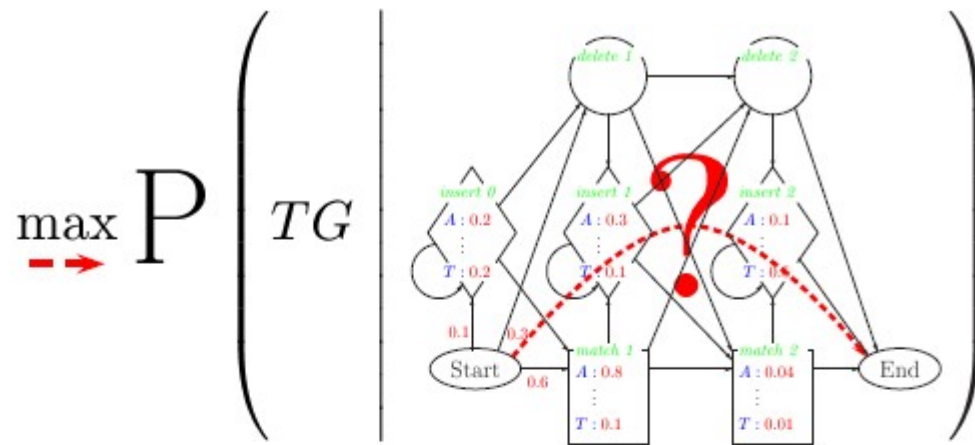
# Annotation – Viterbi algorithm



# Annotation – Viterbi algorithm

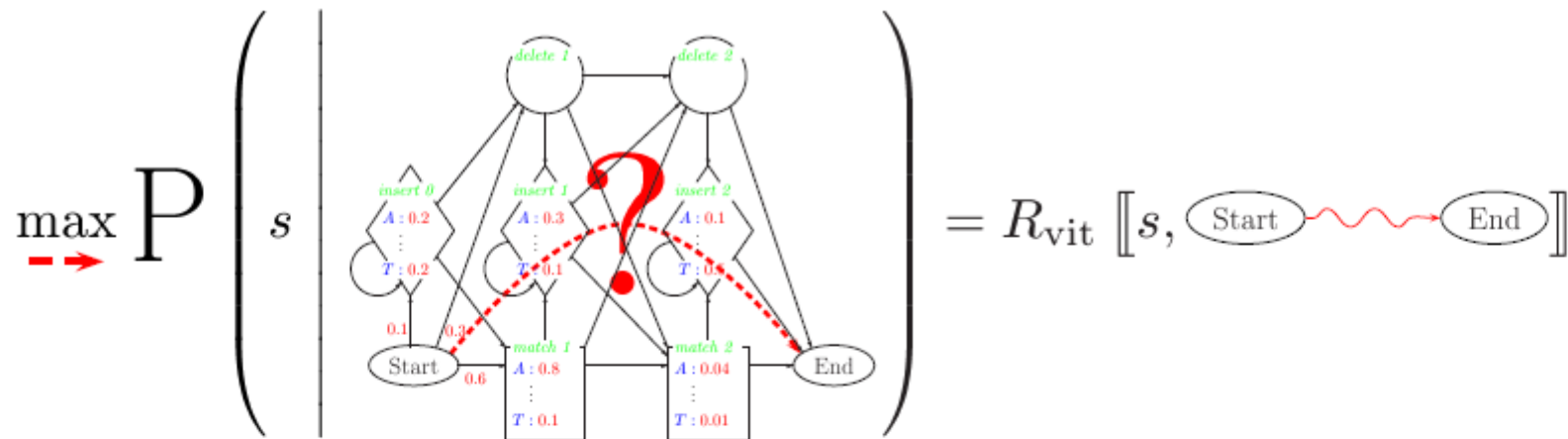


# Annotation – Viterbi algorithm



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

# Annotation – Viterbi algorithm



where

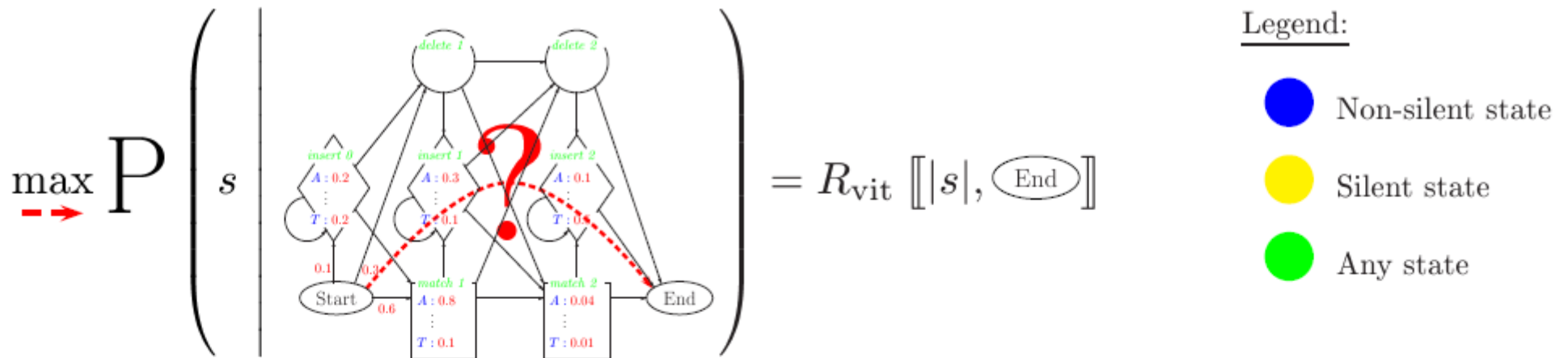
$$R_{\text{vit}} \left[ s[1..i], \text{Start} \rightsquigarrow \text{P} \right] = \max_{\text{q}} \left\{ R_{\text{vit}} \left[ s[1..i], \text{Start} \rightsquigarrow \text{q} \right] \cdot P(\text{q} \rightarrow \text{P}) \right\}$$

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

Legend:

- Non-silent state
- Silent state
- Any state

# Annotation – Viterbi algorithm



where

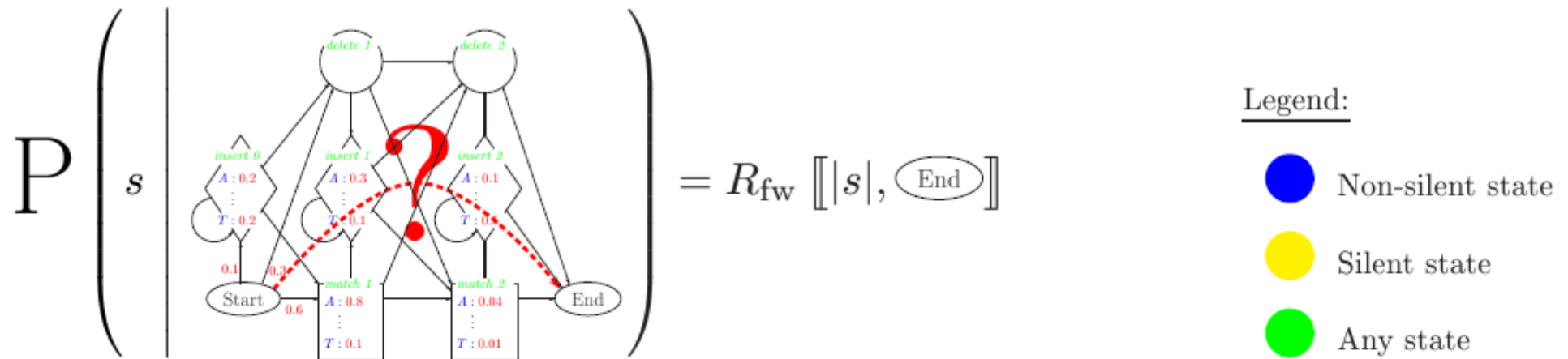
$$R_{\text{vit}} \left[ i, \text{P} \right] = \max_{\text{q}} \left\{ R_{\text{vit}} \left[ i, \text{q} \right] \cdot P \left( \text{q} \rightarrow \text{P} \right) \right\}$$

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

Yields the **Viterbi algorithm** that computes (the probability of) the most likely path that generates a string  $S$  in

time  $O(|S| \cdot \text{\#transitions})$  and space  $O(|S| \cdot \text{\#states})$

# Classification – Forward algorithm



where

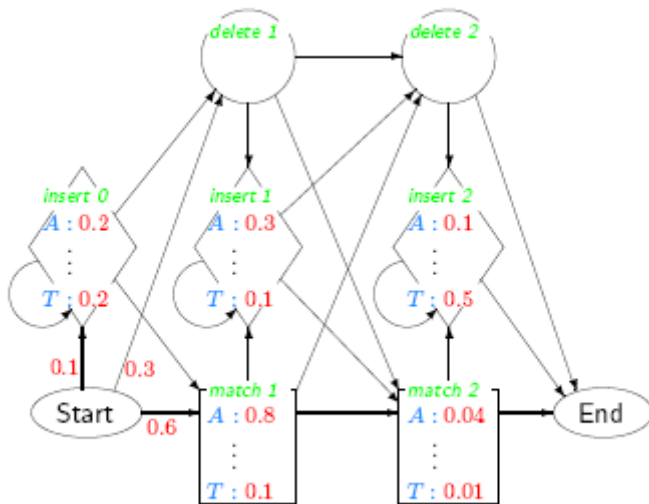
$$R_{\text{fw}} \left[ i, \text{P} \right] = \sum_{\text{Q}} \left\{ R_{\text{fw}} \left[ i, \text{Q} \right] \cdot P \left( \text{Q} \rightarrow \text{P} \right) \right\}$$

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

Yield the **forward algorithm** that computes the total probability of generating string  $S$  in

time  $O(|S| \cdot \text{\#transitions})$  and space  $O(|S| \cdot \text{\#states})$

# Other HMM problems



Pfam is a large collection of sequence alignments and profile HMMs for many common protein sequence families (Dec 2005, 8183 families)

<http://www.sanger.ac.uk/Software/Pfam>

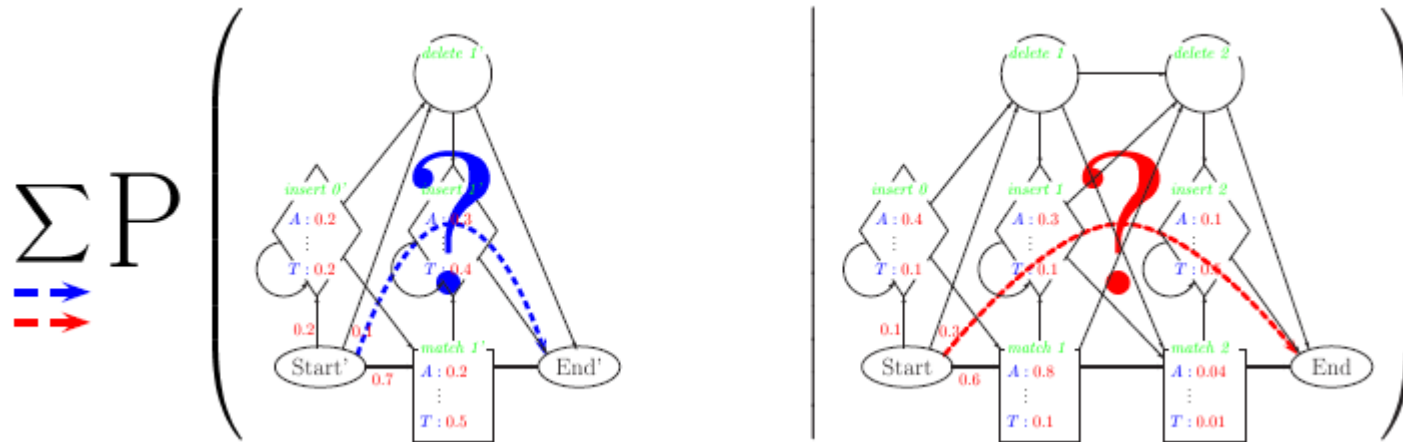
**Comparison:** Given two models, what is a measure of their likeness

*Compare entire sequence families*

**Consensus:** Given a model  $M$ , find the string  $S$  that have the highest probability under the model

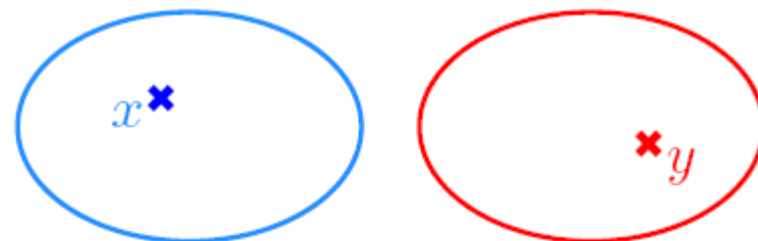
*Extract a short description of a sequence family*

# Comparing two HMMs



# Comparing two HMMs

A basic measure of similarity between two probability distributions is the *collision probability* ...

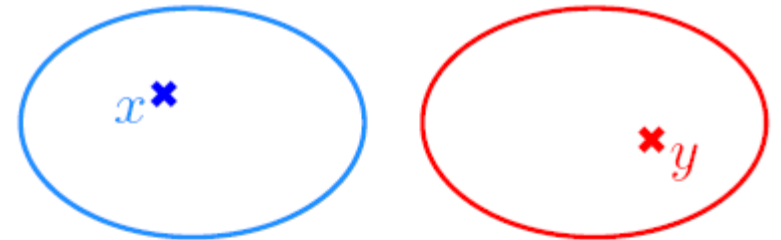


Collision probability:  $P(x = y)$

For two HMMs,  $M_1$  and  $M_2$ , this is:  $C(M_1, M_2) = \sum_{S \in \Sigma^*} P_{M_1}(S) \cdot P_{M_2}(S)$

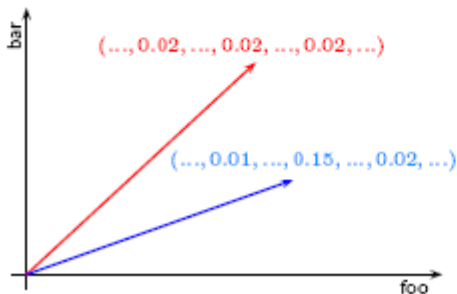
# Comparing two HMMs

A basic measure of similarity between two probability distributions is the *collision probability* ...



Collision probability:  $P(x = y)$

For two HMMs,  $M_1$  and  $M_2$ , this is:  $C(M_1, M_2) = \sum_{S \in \Sigma^*} P_{M_1}(S) \cdot P_{M_2}(S)$



If we view models  $M_1$  and  $M_2$  as vectors in the infinite dimensional vector space spanned by all finite strings, then the collision probability is the standard inner product ...

$$\langle M_1, M_2 \rangle = \sum_{S \in \Sigma^*} P_{M_1}(S) \cdot P_{M_2}(S) = \sum_{S \in \Sigma^*} (M_1)_S \cdot (M_2)_S$$

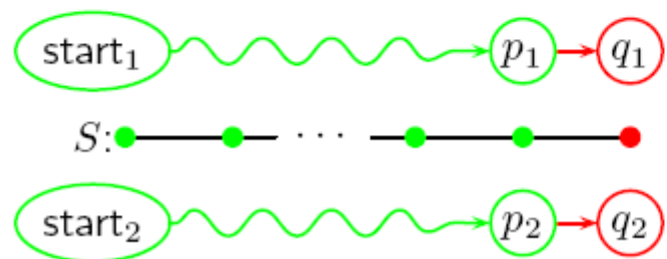
# Computing the collision probability

Let  $C(q_1, q_2)$  be the total probability of all runs  $\pi_1 : \text{start}_1 \rightsquigarrow q_1$  in  $M_1$  and  $\pi_2 : \text{start}_2 \rightsquigarrow q_2$  in  $M_2$  that generate equal strings; then  $C(\text{end}_1, \text{end}_2)$  is the collision probability of  $M_1$  and  $M_2$ , and ...

**Basis:** Before leaving the start-states nothing has been generated ...

$$\begin{array}{c}
 \text{start}_1 \\
 S : \epsilon \\
 \text{start}_2
 \end{array}
 \quad C(\text{start}_1, \text{start}_2) = 1$$

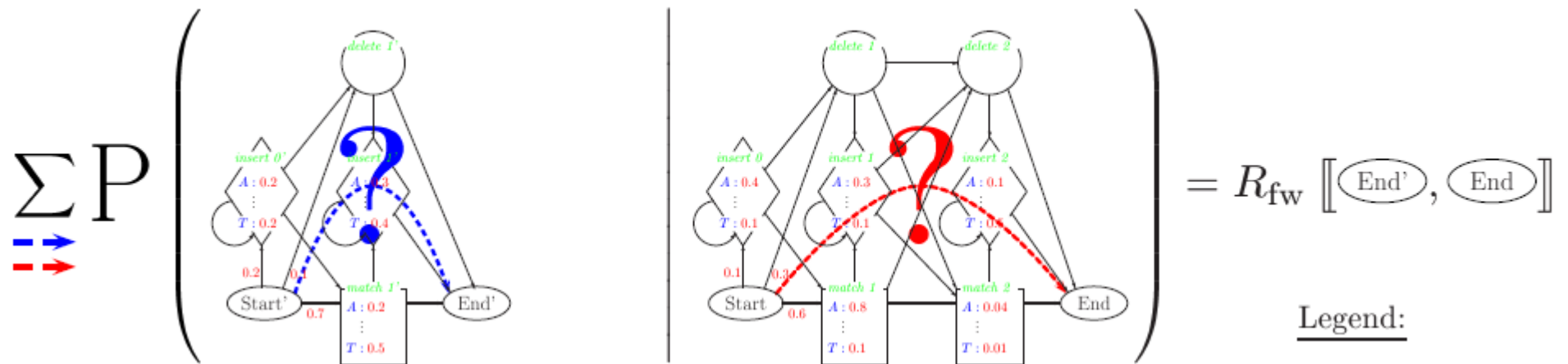
**Recursion:** Add one more transition to the runs ...



$$\begin{aligned}
 C(q_1, q_2) &= \sum_{p_1, p_2} C(p_1, p_2) \\
 &\quad \cdot P_{M_1}(p_1 \rightarrow q_1) P_{M_2}(p_2 \rightarrow q_2) \\
 &\quad \cdot \sum_{\sigma} P_{M_1}(\sigma | q_1) P_{M_2}(\sigma | q_2)
 \end{aligned}$$

# Computing the collision probability

Silent states make it somewhat more complicated ...



Legend:

- Non-silent state
- Silent state
- Any state

$$R_{fw} \llbracket p', p \rrbracket = \sum_{q, q'} \{ R_{fw} \llbracket q', q \rrbracket \cdot P(q' \rightarrow p') \cdot P(q \rightarrow p) \}$$

$$R_{fw} \llbracket p', p \rrbracket = \sum_q \{ R_{fw} \llbracket p', q \rrbracket \cdot P(q \rightarrow p) \}$$

$$R_{fw} \llbracket p', p \rrbracket = \sum_{q, q'} \{ R_{fw} \llbracket q', q \rrbracket \cdot P(q' \rightarrow p') \cdot P(q \rightarrow p) \} \cdot \sum_{\sigma} \{ P(\sigma | p') \cdot P(\sigma | p) \}$$

# Computing the collision probability

Silent states make the recursion more complicated, and the implementation depends on the model structure ...

$$\begin{aligned} C(q_1, q_2) &= \sum_{p_1 \rightarrow q_1, p_2 \rightarrow q_2} C(p_1, p_2) \cdot P(p_1 \rightarrow q_1) \cdot P(p_2 \rightarrow q_2) \cdot E(q_1, q_2) && q_1, q_2 \text{ non-silent} \\ C(q_1, q_2) &= \sum_{p_1 \rightarrow q_1, p_2 \rightarrow q_2} C(p_1, p_2) \cdot P(p_1 \rightarrow q_1) \cdot P(p_2 \rightarrow q_2) && q_1, q_2 \text{ silent} \\ C(q_1, q_2) &= \sum_{p_1 \rightarrow q_1} C(p_1, q_2) \cdot P(p_1 \rightarrow q_1) && q_1 \text{ silent, } q_2 \text{ non-silent} \end{aligned}$$

# Computing the collision probability

Silent states make the recursion more complicated, and the implementation depends on the model structure ...

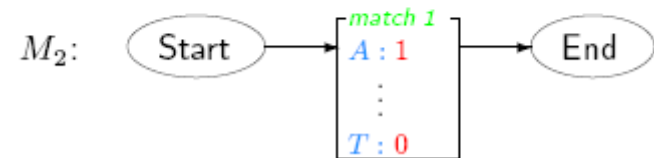
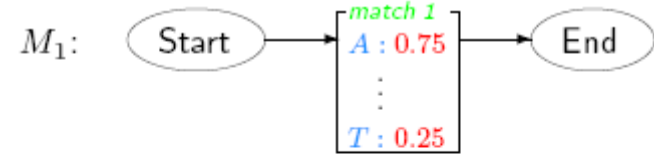
$$\begin{aligned} C(q_1, q_2) &= \sum_{p_1 \rightarrow q_1, p_2 \rightarrow q_2} C(p_1, p_2) \cdot P(p_1 \rightarrow q_1) \cdot P(p_2 \rightarrow q_2) \cdot E(q_1, q_2) && q_1, q_2 \text{ non-silent} \\ C(q_1, q_2) &= \sum_{p_1 \rightarrow q_1, p_2 \rightarrow q_2} C(p_1, p_2) \cdot P(p_1 \rightarrow q_1) \cdot P(p_2 \rightarrow q_2) && q_1, q_2 \text{ silent} \\ C(q_1, q_2) &= \sum_{p_1 \rightarrow q_1} C(p_1, q_2) \cdot P(p_1 \rightarrow q_1) && q_1 \text{ silent, } q_2 \text{ non-silent} \end{aligned}$$

Model Structure	Method	Time Complexity
Left-right, i.e. acyclic	Very simple dynamic programming	$O(m_1 m_2)$ where $m_i$ is number of transitions
Every state is part of at most one cycle	Slightly more complex dynamic programming	$O(m_1 m_2)$ where $m_i$ is number of transitions
Unrestricted	Solving a set of $n_1 n_2$ linear equations	$O((n_1 n_2)^3)$ where $n_i$ is number of states

# A reasonable measure?

The collision probability is not by itself a good similarity measure ...

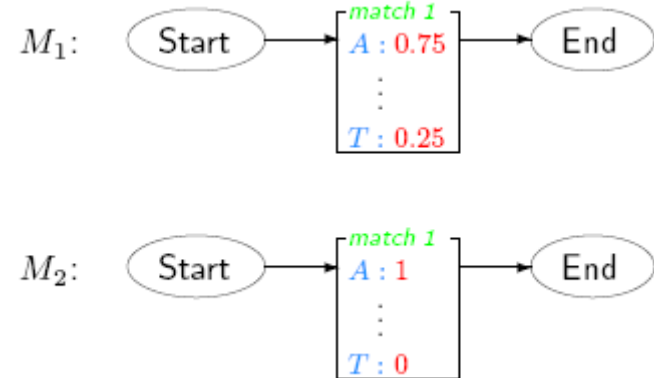
$$C(M_1, M_1) = 0.625 < C(M_1, M_2) = 0.75$$



# A reasonable measure?

The collision probability is not by itself a good similarity measure ...

$$C(M_1, M_1) = 0.625 < C(M_1, M_2) = 0.75$$



Recall that the collision probability  $C(M_1, M_2)$  is the standard inner product of  $M_1$  and  $M_2$  in the vector space spanned by all finite strings.

This can be used to compute other measures:

**$L_2$ -norm:**  $\|M_1 - M_2\|_2 = \sqrt{\langle M_1, M_1 \rangle + \langle M_2, M_2 \rangle - 2\langle M_1, M_2 \rangle}$

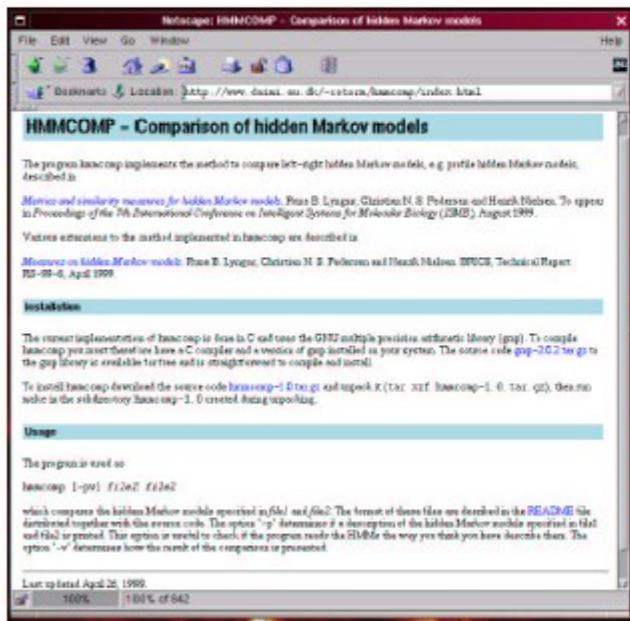
**Angle:**  $M_1 \angle M_2 = \arccos \left( \frac{\langle M_1, M_2 \rangle}{\sqrt{\langle M_1, M_1 \rangle \langle M_2, M_2 \rangle}} \right)$

**Hellinger:**  $D_H(M_1, M_2) = \sin(M_1 \angle M_2) / \sin \left( \frac{\pi - (M_1 \angle M_2)}{2} \right)$

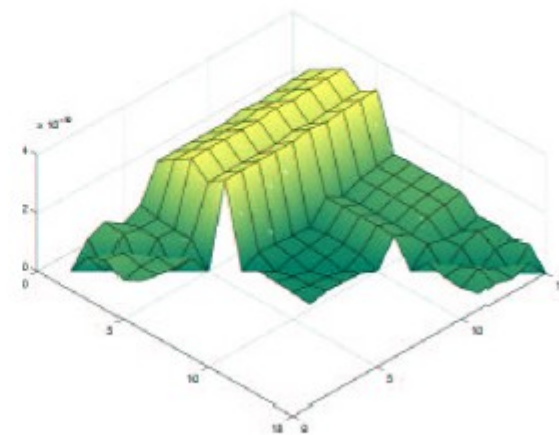
# An experiment

The method for computing the collision probability of left-right HMMs (including profile HMMs) has been implemented in a tool HMMcomp . . .

**An experiment:** Performing all pair-wise comparison of 15 models for “signal peptides” that is expected to divide into 3 groups of 5 models . . .



[www.daimi.au.dk/~cstorm/hmmcomp/](http://www.daimi.au.dk/~cstorm/hmmcomp/)



comparing using the  $L_2$ -distance

# Other measures ...

What about other distance measures? We can show that the  $L_1$  and  $L_\infty$  norms are **NP**-hard to compute ...

- $\|M_1 - M_2\|_1 = \sum_{S \in \Sigma^*} |P_{M_1}(S) - P_{M_2}(S)|$
- $\|M_1 - M_2\|_\infty = \max_{S \in \Sigma^*} |P_{M_1}(S) - P_{M_2}(S)|$

More generally ...

- For fixed even integer  $k$ , we can compute the  $L_k$  norm in time  $O(n^{3k})$
- For any other  $k$  computing the  $L_k$  norm seems to be **NP**-hard

Follows from the **NP-hardness** of the **Consensus String Problem**, that is, to compute the string having highest probability under a HMM ...

# Other measures ...

**Collision Probability:**  $\langle P_M, P_{M'} \rangle$

**Vector  $k$ -Norm:**  $\|P_M - P_{M'}\|_k = \sqrt[k]{\sum_s |P_M(s) - P_{M'}(s)|^k}$

*Special cases:* Computable in time  $O(n^{3k})$  for all even integers  $k$

$\|P_M - P_{M'}\|_1$  and  $\|P_M - P_{M'}\|_\infty = \max_s \{|P_M(s) - P_{M'}(s)|\}$

are **NP**-hard by reductions from  $\max_s \{P_M(s)\}$

**Angle:**  $P_M \angle P_{M'} = \arccos \left( \frac{\langle P_M, P_{M'} \rangle}{\sqrt{\langle P_M, P_M \rangle \cdot \langle P_{M'}, P_{M'} \rangle}} \right)$

**Variation Distance:**  $\max_S \left\{ \left| \sum_{s \in S} P_M(s) - P_{M'}(s) \right| \right\} = \frac{1}{2} \|P_M - P_{M'}\|_1$

**Kullback–Leibler Divergence:**  $H(P_M, P_{M'}) = \sum_s P_M(s) \log \frac{P_M(s)}{P_{M'}(s)}$

# Other measures ...

**Collision Probability:**  $\langle P_M, P_{M'} \rangle$

**Vector  $k$ -Norm:**  $\|P_M - P_{M'}\|_k = \sqrt[k]{\sum_s |P_M(s) - P_{M'}(s)|^k}$

*Special cases:* Computable in time  $O(n^{3k})$  for all even integers  $k$

$\|P_M - P_{M'}\|_1$  and  $\|P_M - P_{M'}\|_\infty = \max_s \{|P_M(s) - P_{M'}(s)|\}$

are **NP**-hard by reductions from  $\max_s \{P_M(s)\}$

**Angle:**  $P_M \angle P_{M'} = \arccos \left( \frac{\langle P_M, P_{M'} \rangle}{\sqrt{\langle P_M, P_M \rangle \cdot \langle P_{M'}, P_{M'} \rangle}} \right)$

**Variation Distance:**  $\max_S \left\{ \left| \sum_{s \in S} P_M(s) - P_{M'}(s) \right| \right\} = \frac{1}{2} \|P_M - P_{M'}\|_1$

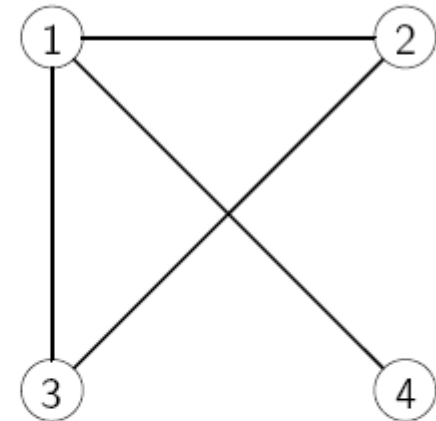
**Kullback–Leibler Divergence:**  $H(P_M, P_{M'}) = \sum_s P_M(s) \log \frac{P_M(s)}{P_{M'}(s)}$

# The consensus string problem

The Consensus String Problem is as hard as the Clique Problem ...

A *graph*  $G = (V, E)$  is a set  $V$  of nodes and a set  $E$  of edges connecting pairs of nodes

**The Clique Problem:** Find large(st) subset of nodes that are all pairwise connected



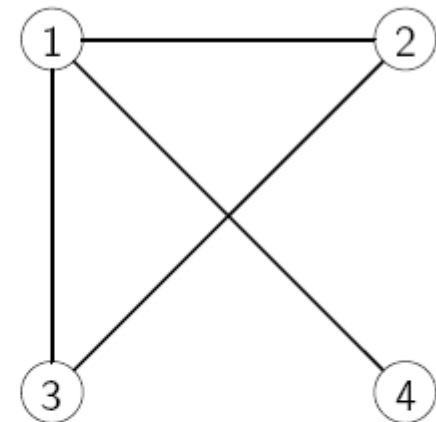
The Clique Problem is **NP**-complete, and if  $\mathbf{P} \neq \mathbf{NP}$ , not approximable within  $\mathcal{O}(|V|^{\frac{1}{2}-\epsilon})$ , for any  $\epsilon > 0$ , in polynomial time ...

# The consensus string problem

The Consensus String Problem is as hard as the Clique Problem ...

A graph  $G = (V, E)$  is a set  $V$  of nodes and a set  $E$  of edges connecting pairs of nodes

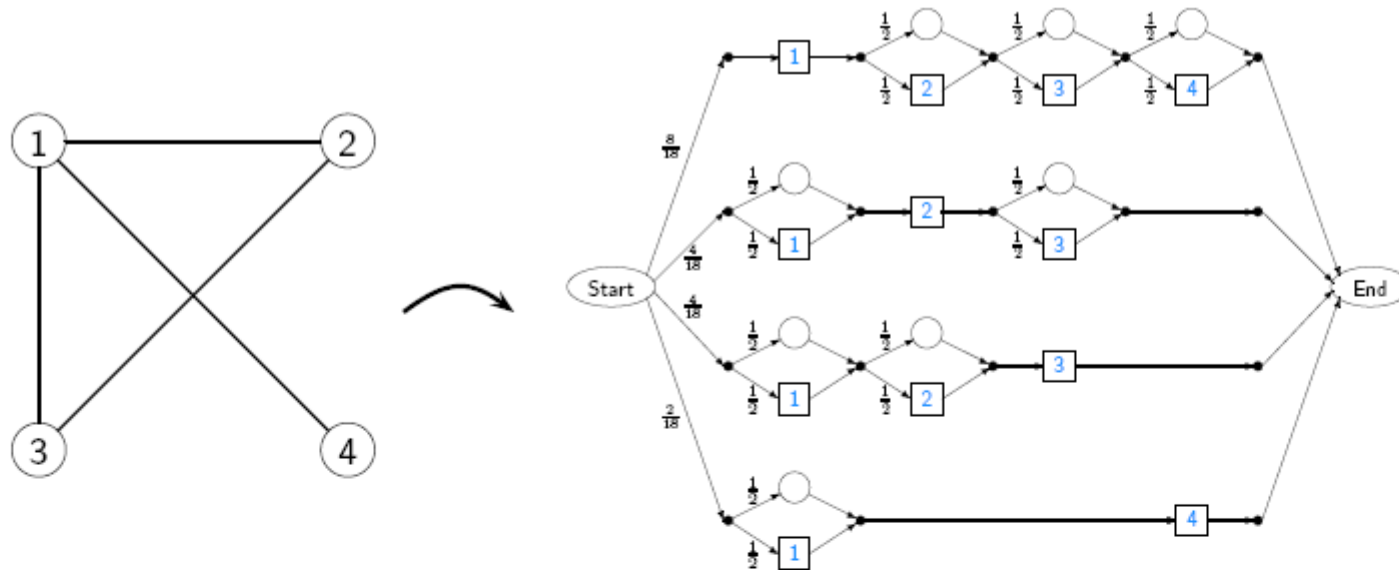
**The Clique Problem:** Find large(st) subset of nodes that are all pairwise connected



The Clique Problem is **NP**-complete, and if  $\mathbf{P} \neq \mathbf{NP}$ , not approximable within  $\mathcal{O}(|V|^{\frac{1}{2}-\epsilon})$ , for any  $\epsilon > 0$ , in polynomial time ...

**Proof Technique:** Given a graph  $G$ , we construct a HMM  $M$ , and argue that finding the string having the highest probability under  $M$  is equivalent to finding the largest clique in  $G$  ...

# The reduction



Given an undirected **graph**, we construct a **HMM** where ...

- Each node in the graph corresponds to a layer in the HMM
- A run through the layer for node  $i$  has to emit  $i$  and can emit  $j$  if node  $i$  has an edge connecting it to node  $j$
- All runs through the HMM have equal probability  $1/\gamma$

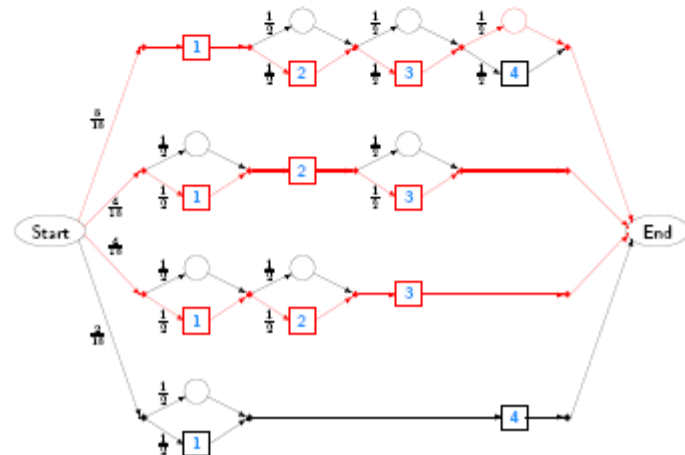
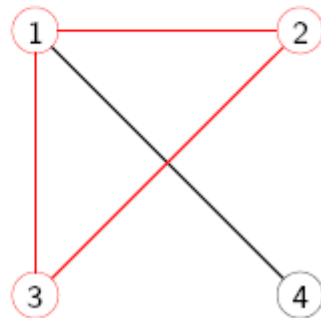
Also possible with an HMM with a **binary alphabet** and **no silent-states** ...

# The equivalence

**Claims:** The HMM is constructed such that ...

- an emitted string is a subsequence of  $1 \cdot 2 \cdot \dots \cdot |V|$
- a string is emitted by at most one run through each layer
- a string can only be emitted by a run through the layer for node  $i$ 
  - if it contains  $i$
  - if it contains  $j \neq i$ , node  $i$  has an edge connecting it to node  $j$

**Conclusion:** There is a string with probability  $k/\gamma$  under the HMM iff there is a clique of size  $k$  in the graph ...



# How hard are the problems?

It is unlikely that we can guarantee to approximate the probability of the consensus string within any constant factor in polynomial time ...

	Clique NP-complete	Consensus NP-hard
$P \neq NP$	Not approximable within $O( V ^{\frac{1}{2}-\epsilon})$	Not approximable within $O(n^{\frac{1}{4}-\epsilon})$
$ZPP \neq NP$	Not approximable within $O( V ^{1-\epsilon})$	Not approximable within $O(n^{\frac{1}{2}-\epsilon})$
$ZPTIME(f(n)) \neq NP,$ $f(n) = 2^{O(\log n (\log \log n)^{\frac{3}{2}})}$	Not approximable within $O( V ^{1-o(1)})$	Not approximable within $O(n^{\frac{1}{2}-o(1)})$

$|V|$  is the size of the graph,  $n < |V|^2$  is the number of states in the HMM

# Derived hardness results

For two hidden Markov models it is ...

- **NP**-hard to approximate the  $L_\infty$  norm within any constant factor

$$\|M - M'\|_\infty = \max_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|$$

- **NP**-hard to compute the  $L_1$  norm and the variation distance

$$\|M - M'\|_1 = \sum_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|$$

$$\|M - M'\|_v = \max_{\Omega \subseteq \Sigma^*} \left| \sum_{S \in \Omega} (P_M(S) - P_{M'}(S)) \right| = \frac{1}{2} \cdot \|M - M'\|_1$$

Furthermore ...

- **NP**-hard to approximate the probability of the best annotation of a string by a class HMM within any constant factor

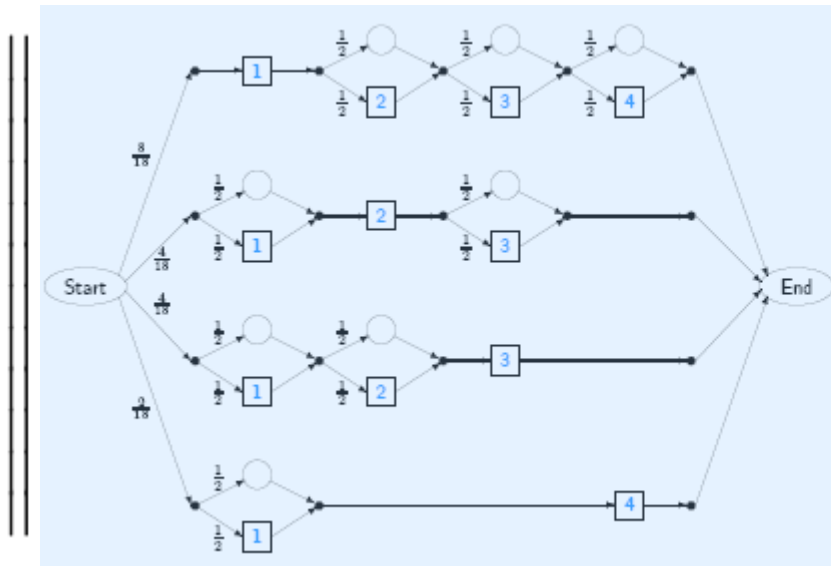
# Hardness of the $L_\infty$ norm

$$\|M - M'\|_\infty = \lim_{k \rightarrow \infty} \sqrt[k]{\sum_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|^k} = \max_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|$$

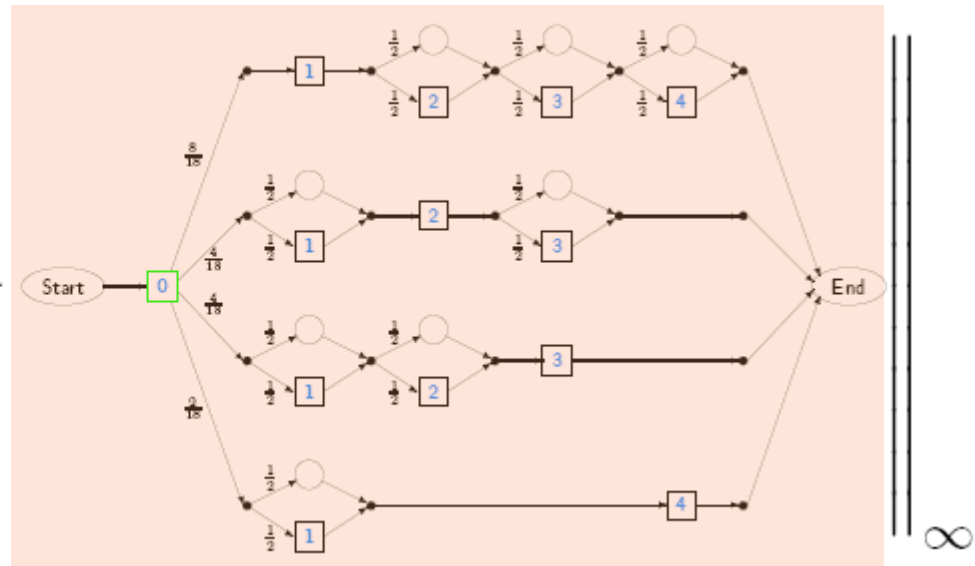
**Idea:** Reduction from clique (via consensus string). We construct two HMMs  $M$  and  $M'$  from a graph  $G$  cf. the construction from the hardness proof of consensus string, and see that computing the  $L_\infty$  norm between  $M$  and  $M'$  yields the probability of the consensus string of  $M$ , i.e. yields the maximum clique size of  $G$ ...

# Hardness of the $L_\infty$ norm

$$\|M - M'\|_\infty = \lim_{k \rightarrow \infty} \sqrt[k]{\sum_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|^k} = \max_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|$$



$M$

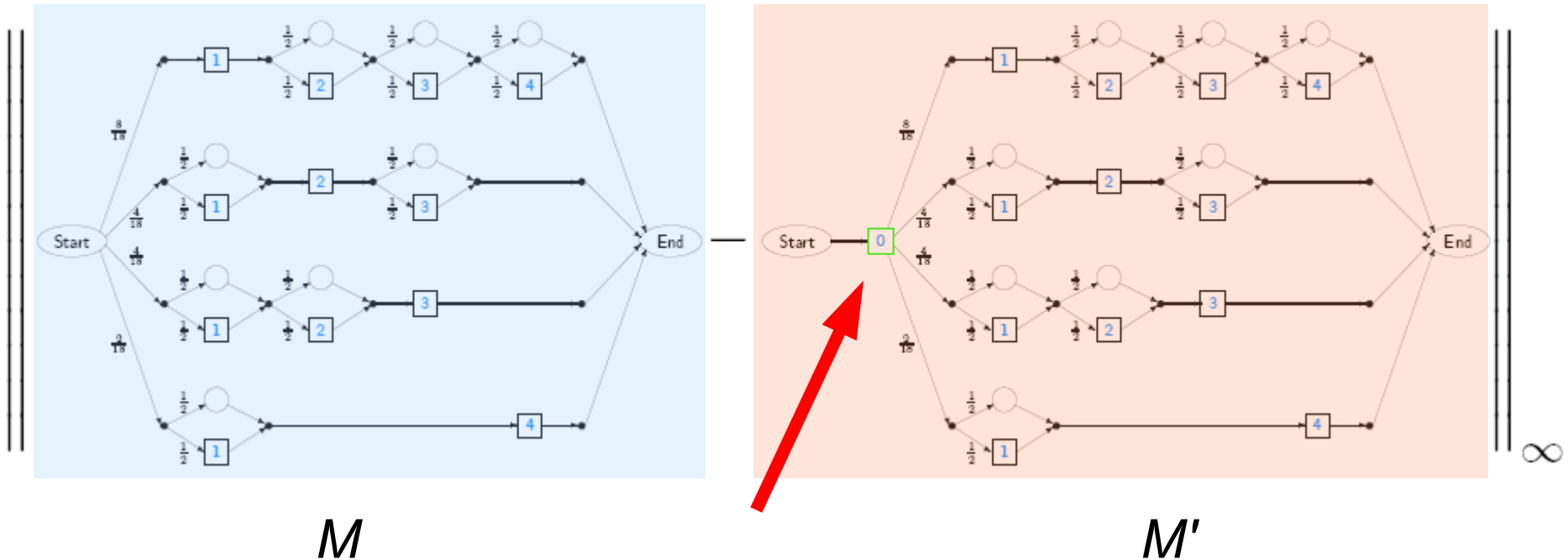


$M'$

$\infty$

# Hardness of the $L_\infty$ norm

$$\|M - M'\|_\infty = \lim_{k \rightarrow \infty} \sqrt[k]{\sum_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|^k} = \max_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|$$



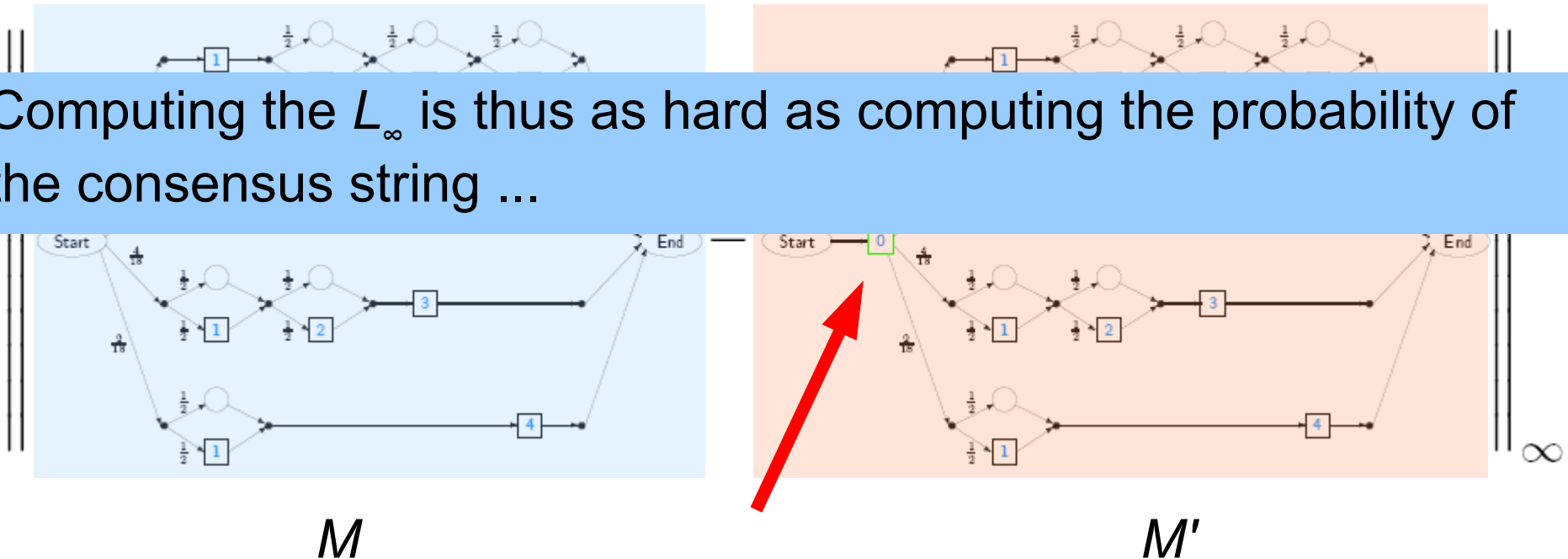
$M$  can generate  $S$  iff  $M'$  can generate  $0S$ , i.e.

$$\max |P_M(S) - P_{M'}(S)| = \max |P_M(S)|$$

# Hardness of the $L_\infty$ norm

$$\|M - M'\|_\infty = \lim_{k \rightarrow \infty} \sqrt[k]{\sum_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|^k} = \max_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|$$

Computing the  $L_\infty$  is thus as hard as computing the probability of the consensus string ...



$M$  can generate  $S$  iff  $M'$  can generate  $0S$ , i.e.

$$\max |P_M(S) - P_{M'}(S)| = \max |P_M(S)|$$

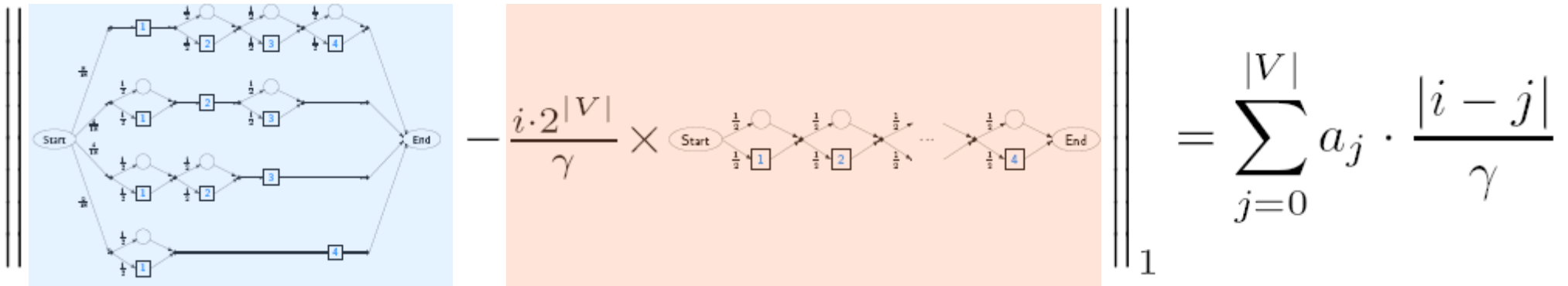
# Hardness of the $L_1$ norm

$$\|M - M'\|_1 = \sum_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|$$

**Idea:** Same as before, i.e. reduction from clique (via consensus string). We construct two HMMs  $M$  and  $M'$  and see that if we can compute the  $L_1$  norm between  $M$  and  $M'$  then we (in polynomial time) can derive the maximum clique size of  $G$ ...

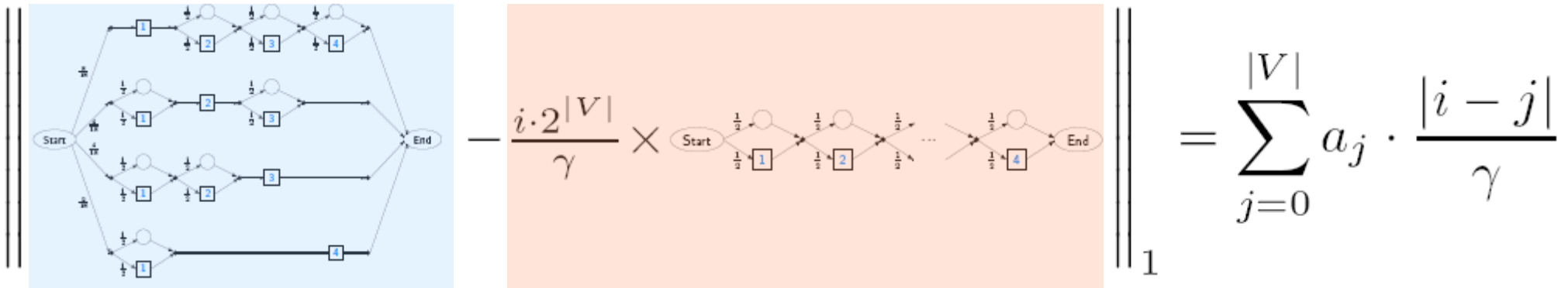
# Hardness of the $L_1$ norm

$$\|M - M'\|_1 = \sum_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|$$



# Hardness of the $L_1$ norm

$$\|M - M'\|_1 = \sum_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|$$

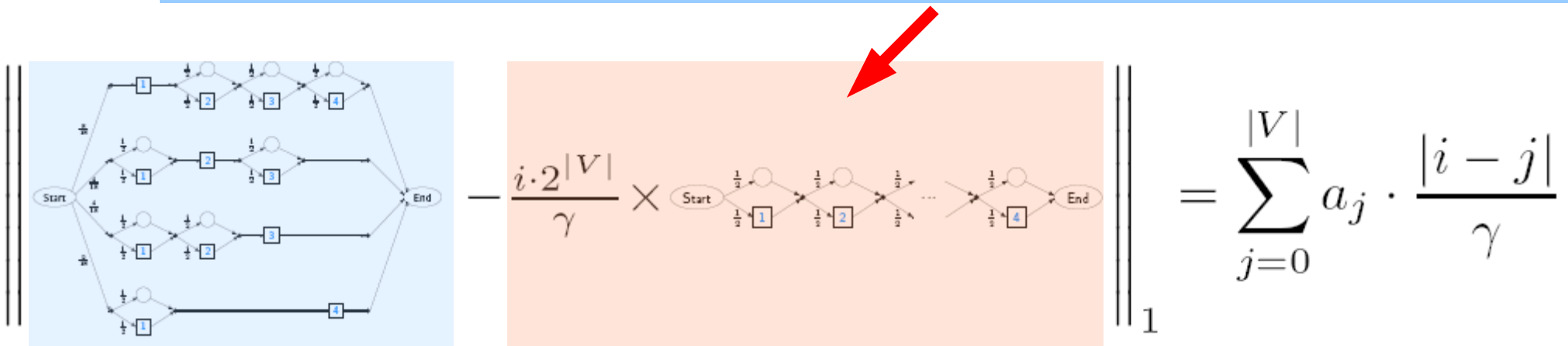


The model  $M_G$  which from the reduction from clique. Every string generated by  $M_G$  is a subsequence of  $1..|V|$  and has probability  $i/\gamma$  for some  $i$  in  $\{0, 1, \dots, |V|\}$ .

Let  $a_j$  be the number of strings generated with prob  $j/\gamma$

# Hardness of the $L_1$ norm

A model  $M_i$  which generates every subsequence of  $1..|V|$  with equal probability  $i/\gamma$  for any fixed  $i$  in  $\{0, 1, \dots, |V|\}$  and all other strings with “probability 0” (see technical trick in paper)

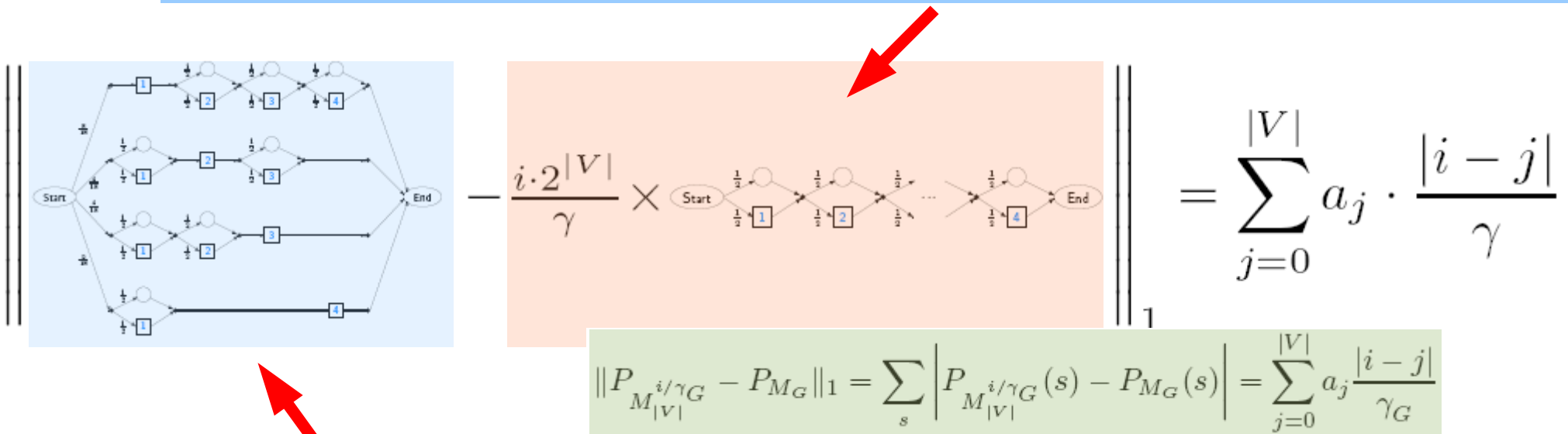


The model  $M_G$  which from the reduction from clique. Every string generated by  $M_G$  is a subsequence of  $1..|V|$  and has probability  $i/\gamma$  for some  $i$  in  $\{0, 1, \dots, |V|\}$ .

Let  $a_j$  be the number of strings generated with prob  $j/\gamma$

# Hardness of the $L_1$ norm

A model  $M_i$  which generates every subsequence of  $1..|V|$  with equal probability  $i/\gamma$  for any fixed  $i$  in  $\{0, 1, \dots, |V|\}$  and all other strings with “probability 0” (see technical trick in paper)

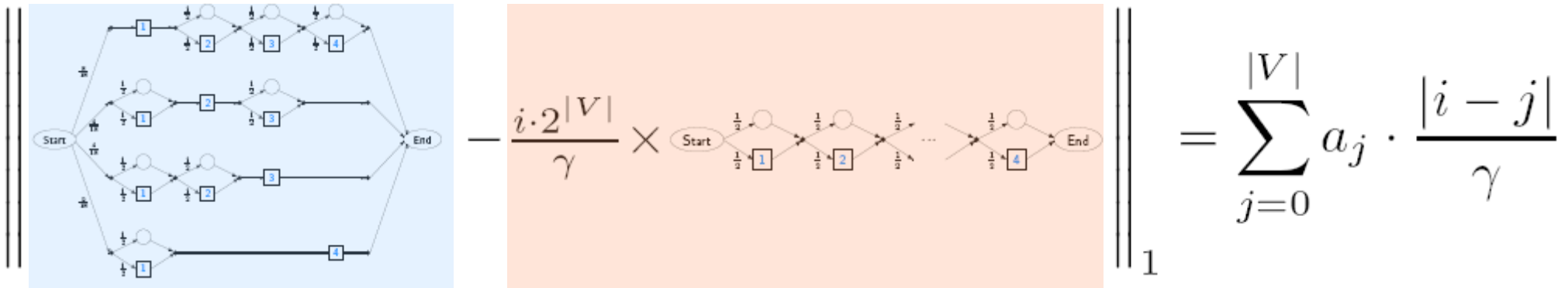


The model  $M_G$  which from the reduction from clique. Every string generated by  $M_G$  is a subsequence of  $1..|V|$  and has probability  $i/\gamma$  for some  $i$  in  $\{0, 1, \dots, |V|\}$ .

Let  $a_j$  be the number of strings generated with prob  $j/\gamma$

# Hardness of the $L_1$ norm

$$\|M - M'\|_1 = \sum_{S \in \Sigma^*} |P_M(S) - P_{M'}(S)|$$

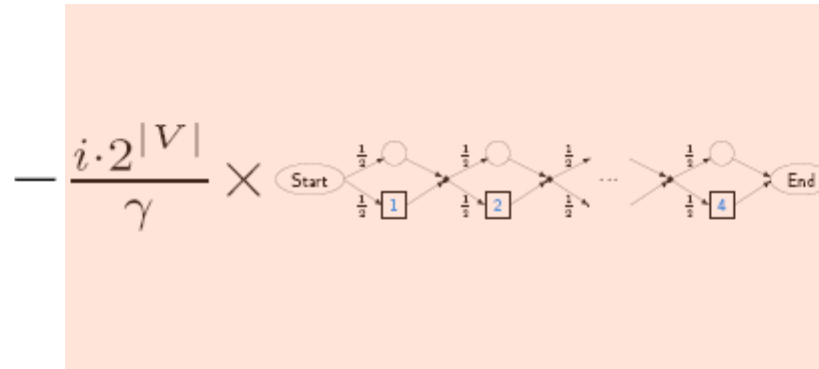
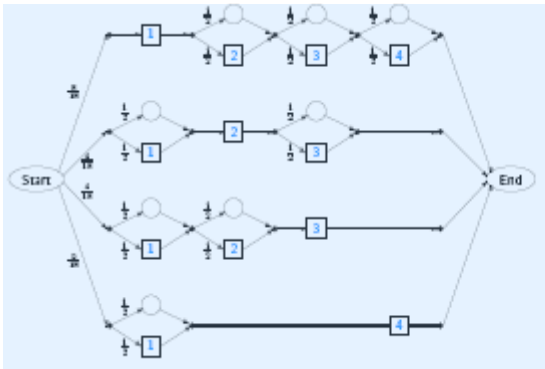


By repeating the comparison for all  $i \in \{0, 1, \dots, |V|\}$ , we get an efficiently solvable set of linear equations  $Ma = \gamma l$ , where  $M_{ij} = |i - j|$  and  $l_i$  is the result of the  $i$ 'th comparison ...

# Hardness of the $L_1$ norm

**Conclusion:** There is a string with probability  $k/\gamma$  under the HMM iff there is a clique of size  $k$  in the graph ...

Let  $a_j$  be the number of strings generated with prob  $j/\gamma$



$$\left\| \left\| \frac{i \cdot 2^{|V|}}{\gamma} \times \text{Diagram} \right\| \right\|_1 = \sum_{j=0}^{|V|} a_j \cdot \frac{|i - j|}{\gamma}$$

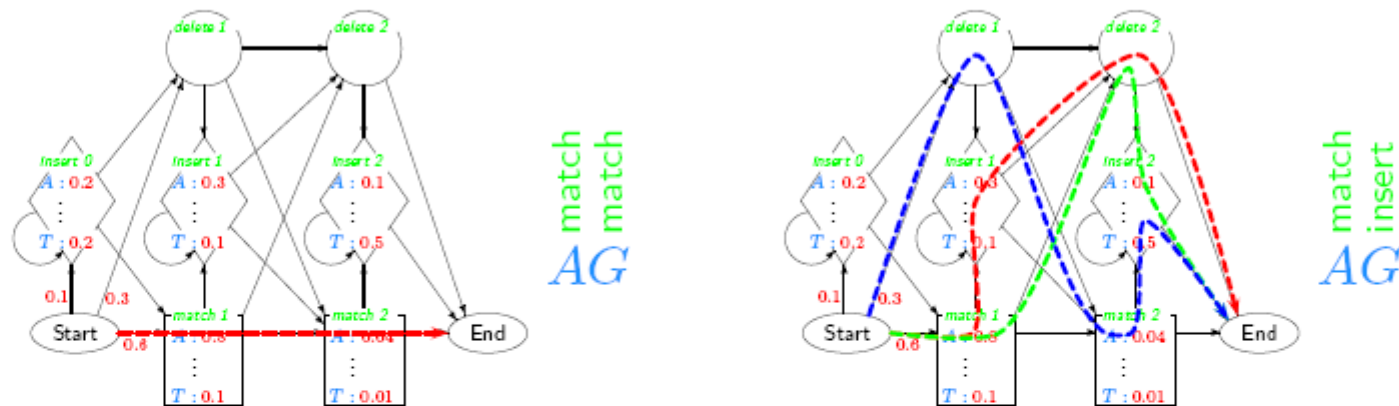
By repeating the comparison for all  $i \in \{0, 1, \dots, |V|\}$ , we get an efficiently solvable set of linear equations  $Ma = \gamma l$ , where  $M_{ij} = |i - j|$  and  $l_i$  is the result of the  $i$ 'th comparison ...

The maximum  $j$ , where  $a_j \neq 0$ , is the size of the maximum clique in  $G$ . Computing the  $L_1$  norm is thus as hard as the Clique Problem ...

# Other types of HMMs

In a class HMM  $M$  the non-silent states are divided into classes. It is used to annotate strings with classes, not states . . .

Example:



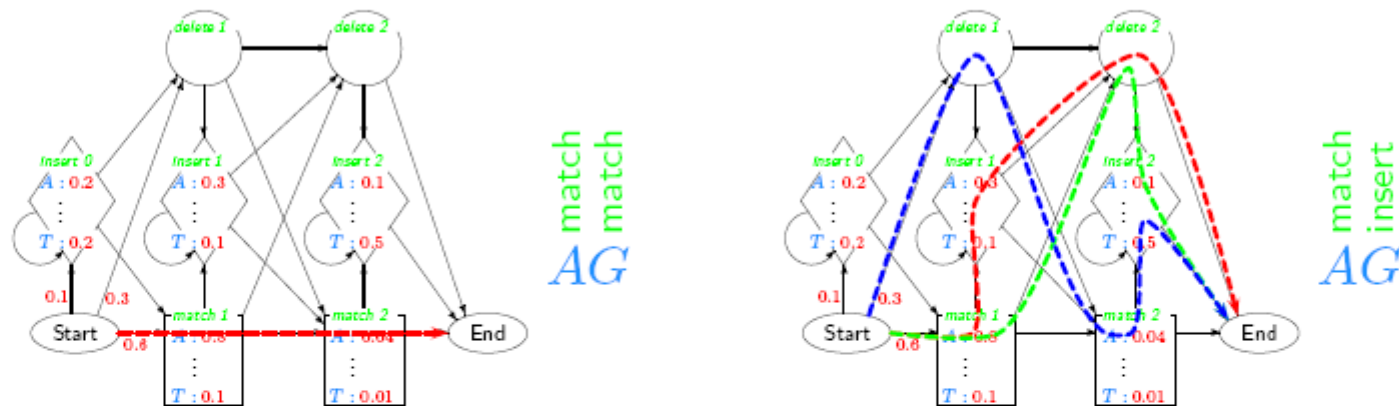
We will say that a path  $\pi$  in  $M$  is *consistent* with an annotation  $l$  if

- the number of non-silent states in  $\pi$  is equal to the length of  $l$
- the  $i$ 'th non-silent state in  $\pi$  is of class  $l_i$

# Other types of HMMs

In a class HMM  $M$  the non-silent states are divided into classes. It is used to annotate strings with classes, not states ...

Example:



We will say that a path  $\pi$  in  $M$  is *consistent* with an annotation  $l$  if

- the number of non-silent states in  $\pi$  is equal to the length of  $l$
- the  $i$ 'th non-silent state in  $\pi$  is of class  $l_i$

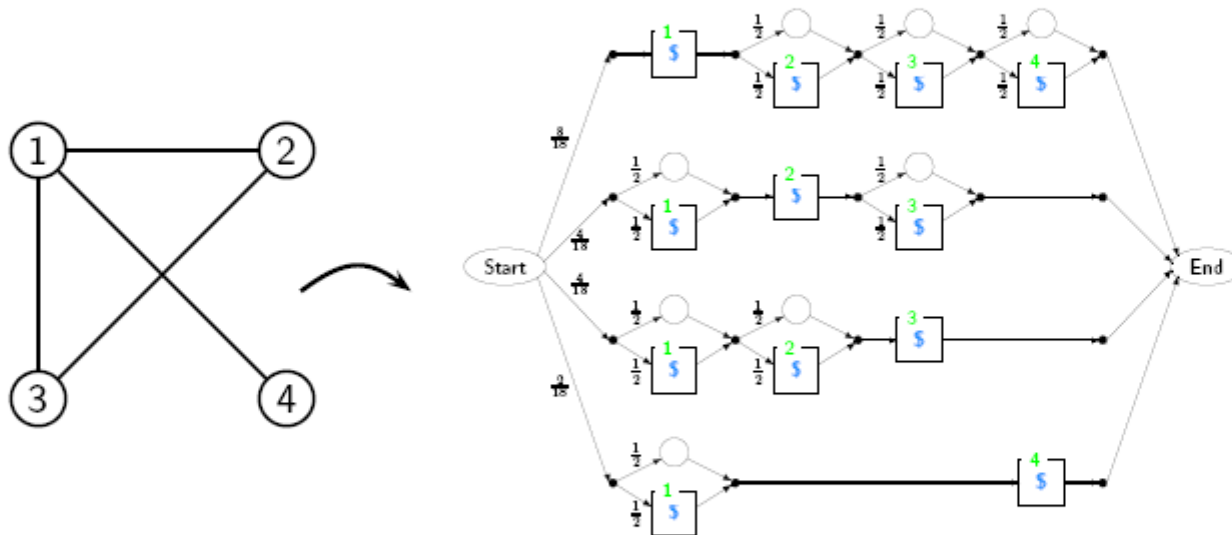
An optimal annotation  $l$  of a string  $S$  is one maximizing

$$\sum_{\pi \text{ consistent with } l} P_M(\pi, S)$$

# Hardness of class HMM annotation

Finding an optimal annotation of a string is essentially like finding the consensus string subject to certain restrictions . . .

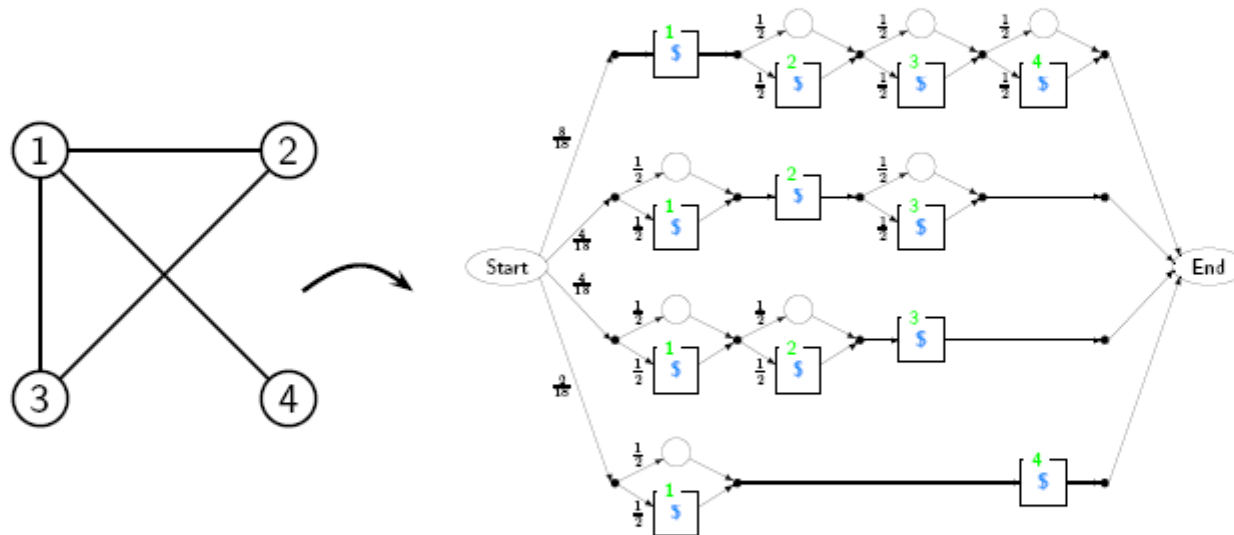
The **NP**-hardness of the class HMM annotation problem can be established in much the same way as for the consensus string problem



# Hardness of class HMM annotation

Finding an optimal annotation of a string is essentially like finding the consensus string subject to certain restrictions ...

The **NP**-hardness of the class HMM annotation problem can be established in much the same way as for the consensus string problem



**Obs:**  $P_M(S) = P_{M'}(\$^{|S|}, S)$ , i.e. the size of the largest clique in the graph is proportional to the prob. of the most likely labeling of any string ...

$$\max_{0 \leq k \leq |V|} \{ \text{Probability of optimal annotation of } \$^k \text{ in } M' \}$$

# Summary

## Basic result ...

- **NP**-hard to approximate the probability of the consensus string of an HMM within any constant factor

## This implies ...

- **NP**-hard to approximate the  $L_\infty$ -distance between two HMMs within any constant factor
- **NP**-hard to compute the  $L_1$ - and variation distance between two HMMs

## More generally ...

- For fixed even integer  $k$ , we can compute the  $L_k$ -distance between HMMs in time  $O(n^{3k})$ , where  $n$  is the number of states
- For any other  $k$  computing the  $L_k$ -distance between two HMMs seems to be **NP**-hard?