CSE 549: Hidden Markov Models (in prep for ab initio gene finding)

Stony Brook University

All slides marked * are courtesy of Carl Kingsford

A Basic Probability Refresher

Ω — The sample space (set of all possible outcomes)

e.g. all possible values for a roll of a die {1,2,3,4,5,6}, need not be finite

E — An event; a subset of the sample space

e.g. all even rolls of a die {2,4,6}

X — Random variable; measurable function from $\Omega \rightarrow E$

e.g. X = value on the upward face of the die

Pr(E) — Probability of the event E

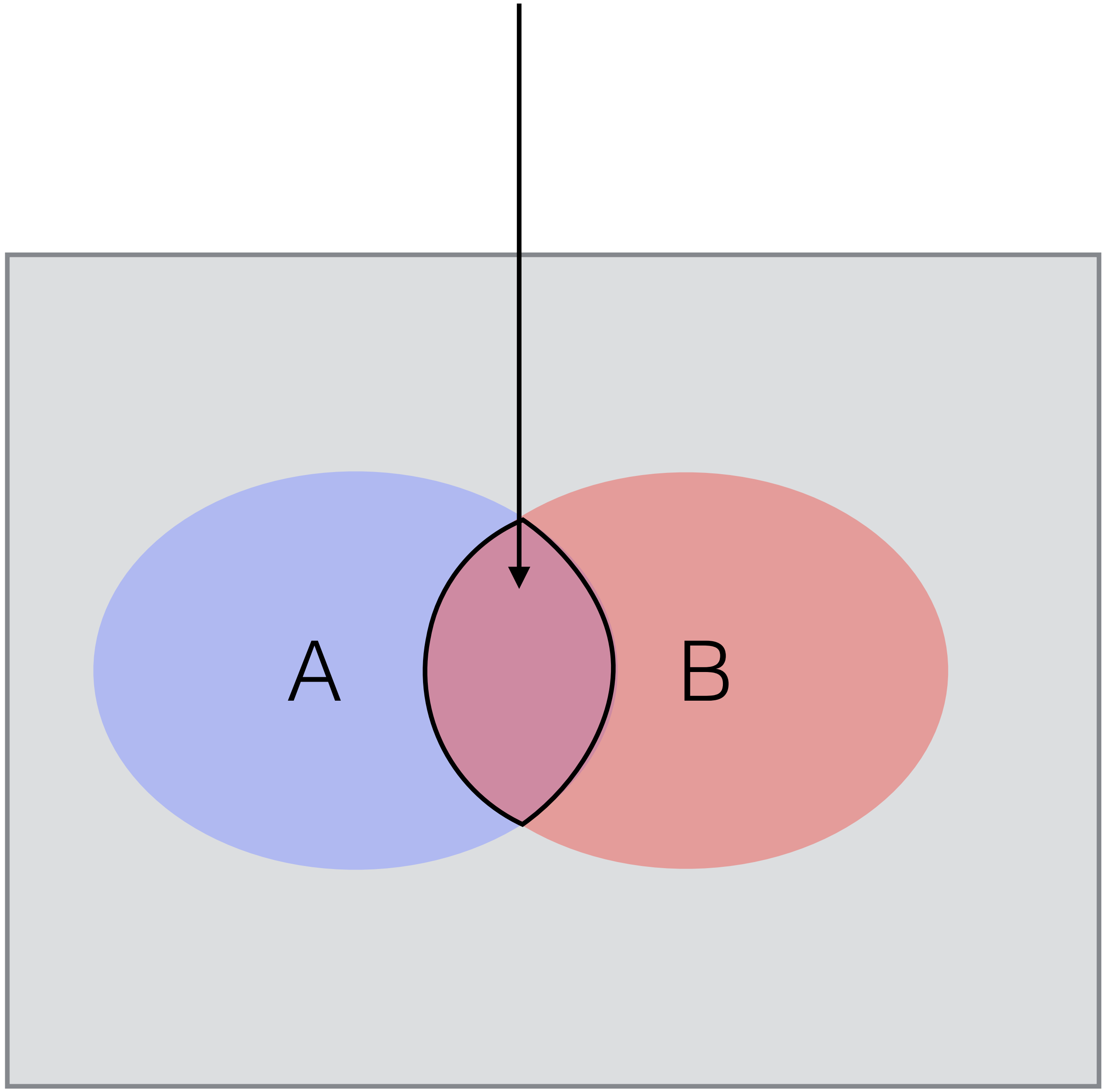
e.g. $\Pr(X \text{ is even}) = \Pr(X \in \{2,4,6\}) = |\{2,4,6\}| / |\{1,2,3,4,5,6\}| = 0.5$

In a discrete, finite, sample space, **if all outcomes are equally likely**, one can think of this as $|E| / |\Omega|$

Probabilities

<u>Joint Probability</u>

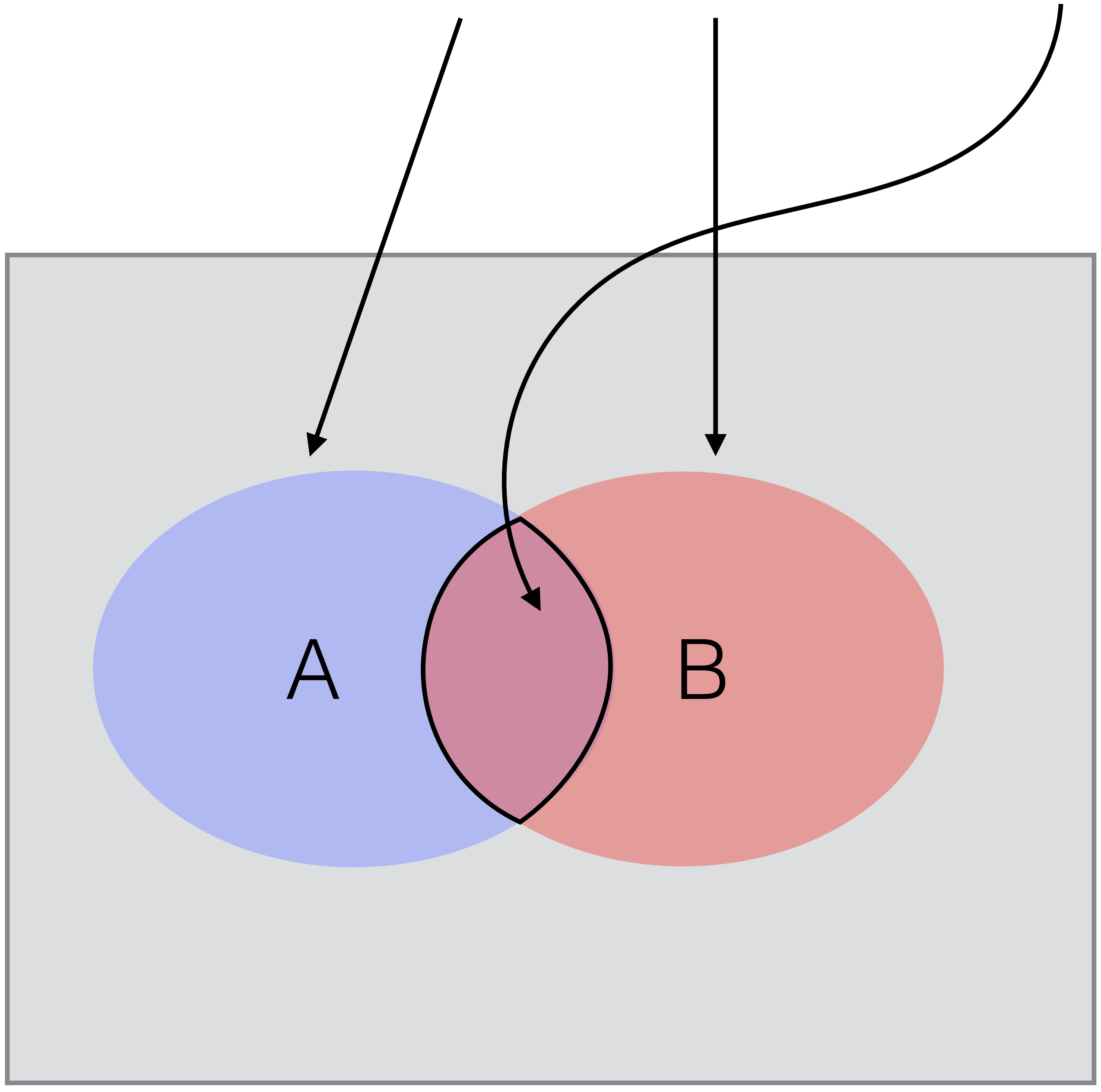
$$\Pr(A,B) = \Pr(A \cap B) = \Pr(A|B)\,\Pr(B)$$



Probabilities

Probability of Union of Two Events

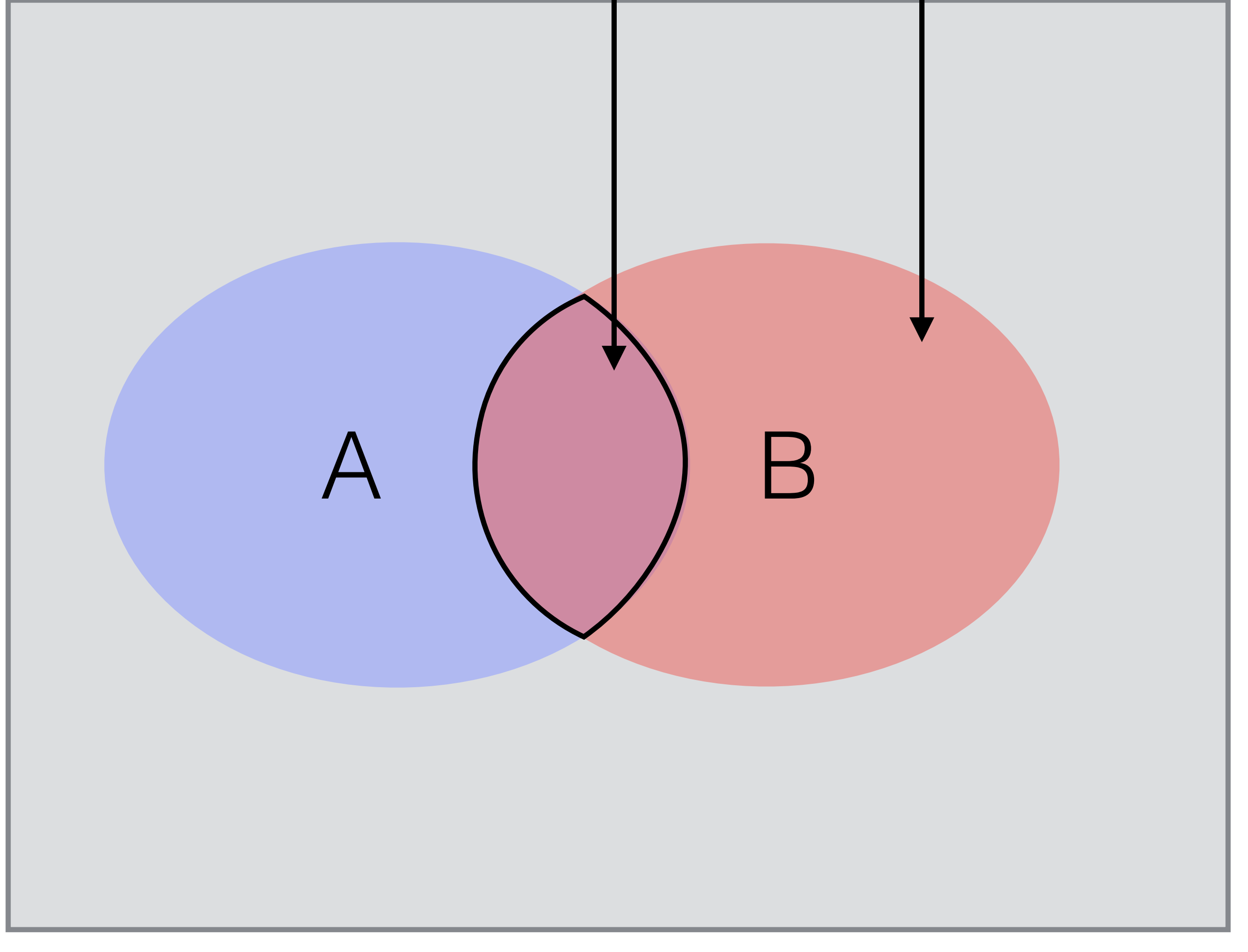
$$Pr(A \cup B) = Pr(A) + Pr(B) - Pr(A,B)$$



Probabilities

Conditional Probability

$$Pr(A|B) = Pr(A,B) \,/\, Pr(B)$$



Independence

Independent Events

$$X \perp Y \Leftrightarrow \Pr(X,Y) = \Pr(X)\Pr(Y)$$

Conditionally Independent Events

$$X \perp Y \mid Z \Leftrightarrow \Pr(X,Y \mid Z) = \Pr(X \mid Z)\Pr(Y \mid Z)$$

Checking a Casino

Fair coin:
Pr(Heads) = 0.5

Biased coin:
Pr(Heads) = 0.75

?

Suppose either a fair or biased coin was used to generate a sequence of heads & tails. But we don't know which type of coin was actual used.

Heads/Tails: ↑ ↑ ↓ ↓ ↓ ↓ ↑ ↑ ↑ ↑ ↓ ↑ ↓ ↑ ↓ ↑ ↓ ↑

How could we guess which coin was more likely?

*

Compute the Probability of the Observed Sequence

Fair coin: Pr(Heads) = 0.5
Biased coin: Pr(Heads) = 0.75

X = ↑ ↑ ↓ ↓ ↓ ↓ ↑

$$\Pr(x \mid \text{Fair}) = 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 = 0.5^7 = 0.0078125$$

$$\Pr(x \mid \text{Biased}) = 0.75 \times 0.75 \times 0.25 \times 0.25 \times 0.25 \times 0.25 \times 0.75 = 0.001647949$$

The *log-odds score:*

$$\log_2 \frac{\Pr(x \mid \text{Fair})}{\Pr(x \mid \text{Biased})} = \log_2 \frac{0.0078}{0.0016} = 2.245$$

> 0. Hence "Fair" is a better guess.

*

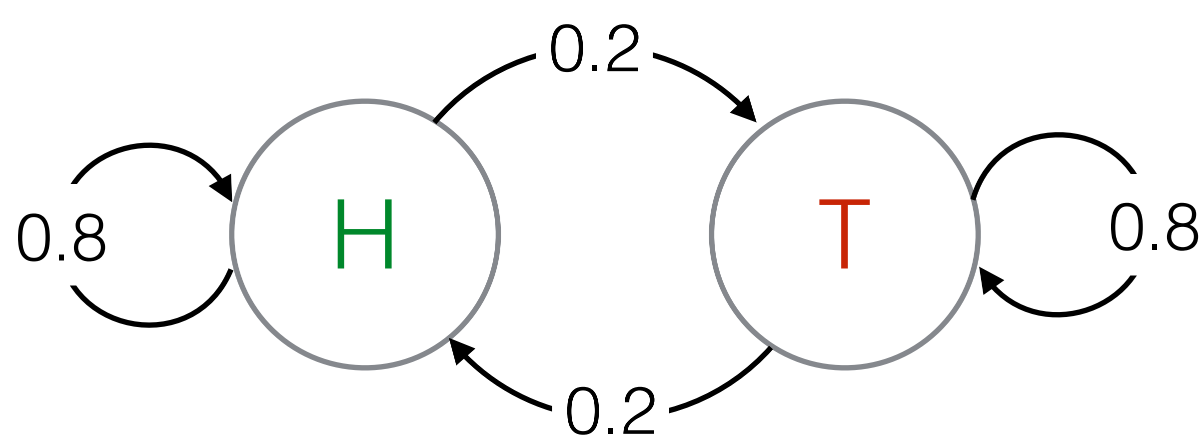
(1^{st} order) Markov Chain

What if the coin tosses weren't independent?

e.g. coin tended to have "runs" of the same value.

Discrete, random process where the ***next state depends only on the current state***.

Markov Chain describing a "hot" coin



Total transition probability at each state is unity

(1^{st} order) Markov Chain

What if the coin tosses weren't independent?

e.g. coin tended to have "runs" of the same value.

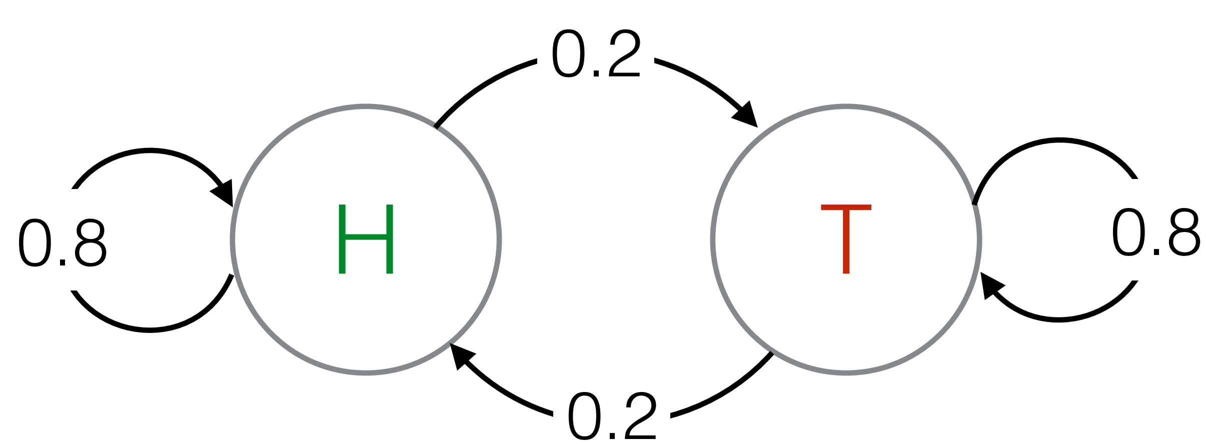
Discrete, random process where the ***next state depends only on the current state***.

Factorization Always True

$$\Pr(x_1, x_2, \ldots, x_n) = \Pr(x_n \mid x_1, x_2, \ldots, x_{n-1}) \Pr(x_{n-1} \mid x_1, x_2, \ldots, x_{n-2}) \ldots \Pr(x_2 \mid x_1) \Pr(x_1)$$

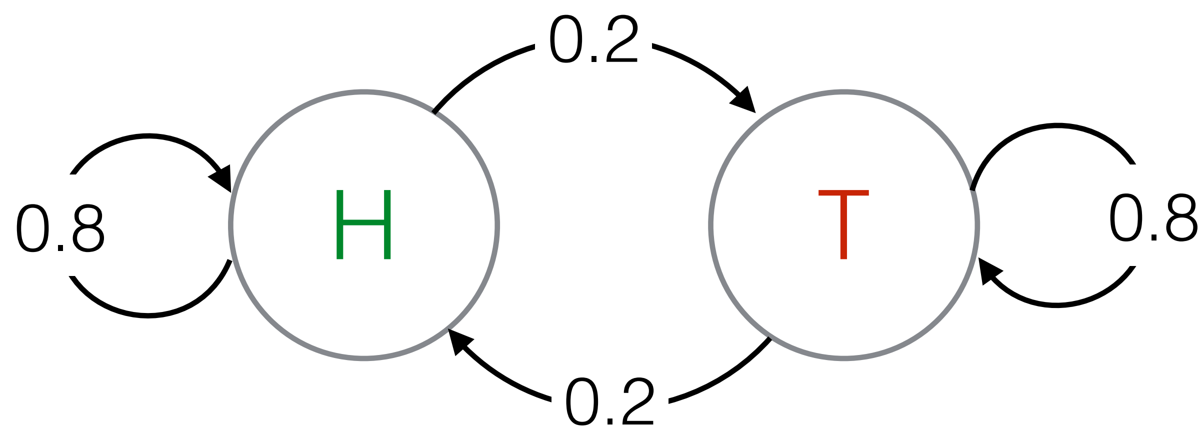
Factorization True if process is Markovian

$$\Pr(x_1, x_2, \ldots, x_n) = \Pr(x_n \mid x_{n-1}) \Pr(x_{n-1} \mid x_{n-2}) \ldots \Pr(x_2 \mid x_1) \Pr(x_1)$$



(1st order) Markov Chain

Markov Chain describing a "hot" coin



Can define the joint probability for a sequence of observations:

$$\mathbf{x} = \text{HTTTTTHHHHTTTHHHHH}$$

$$\Pr(\mathbf{x}) = \Pr(x_1, x_2, \ldots, x_n) = \Pr(x_1) \prod_{t=2}^{n} \Pr(x_t \mid x_{t-1})$$

e.g. above, if we were equally likely to start at H or T

$$\Pr(\mathbf{x}) = 0.5 \times 0.2 \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.2 \times 0.8 \times 0.8 \times 0.8 \times 0.2 \times 0.8 \times 0.8 \times 0.2 \times 0.8 \times 0.8 \times 0.8 \times 0.8$$

$$\approx .000043980465111040$$

Markov Chain Detour

Mark V Shaney Developed by Bruce Ellis & Rob Pike in the 1980s

>From mvs Fri Nov 16 17:11 EST 1984 remote from alice
It looks like Reagan is going to say? Ummm... Oh yes, I was looking for. I'm so glad I remembered it. Yeah, what I have wondered if I had committed a crime. Don't eat with your assessment of Reagon and Mondale. Up your nose with a guy from a firm that specifically researches the teen-age market. As a friend of mine would say, "It really doesn't matter"... It looks like Reagan is holding back the arms of the American eating public have changed dramatically, and it got pretty boring after about 300 games.

People, having a much larger number of varieties, and are very different from what one can find in Chinatowns across the country (things like pork buns, steamed dumplings, etc.) They can be cheap, being sold for around 30 to 75 cents apiece (depending on size), are generally not greasy, can be adequately explained by stupidity. Singles have felt insecure since we came down from the Conservative world at large. But Chuqui is the way it happened and the prices are VERY reasonable.

Can anyone think of myself as a third sex. Yes, I am expected to have. People often get used to me knowing these things and then a cover is placed over all of them. Along the side of the $$ are spent by (or at least for) the girls. You can't settle the issue. It seems I've forgotten what it is, but I don't. I know about violence against women, and I really doubt they will ever join together into a large number of jokes. It showed Adam, just after being created. He has a modem and an autodial routine. He calls my number 1440 times a day. So I will conclude by saying that I can well understand that she might soon have the time, it makes sense, again, to get the gist of my argument, I was in that (though it's a Republican administration).
_-_-_-_-Mark

- "I spent an interesting evening recently with a grain of salt."
- "I hope that there are sour apples in every bushel."

One appealing property of Markov Chains is that they are ***generative*** models

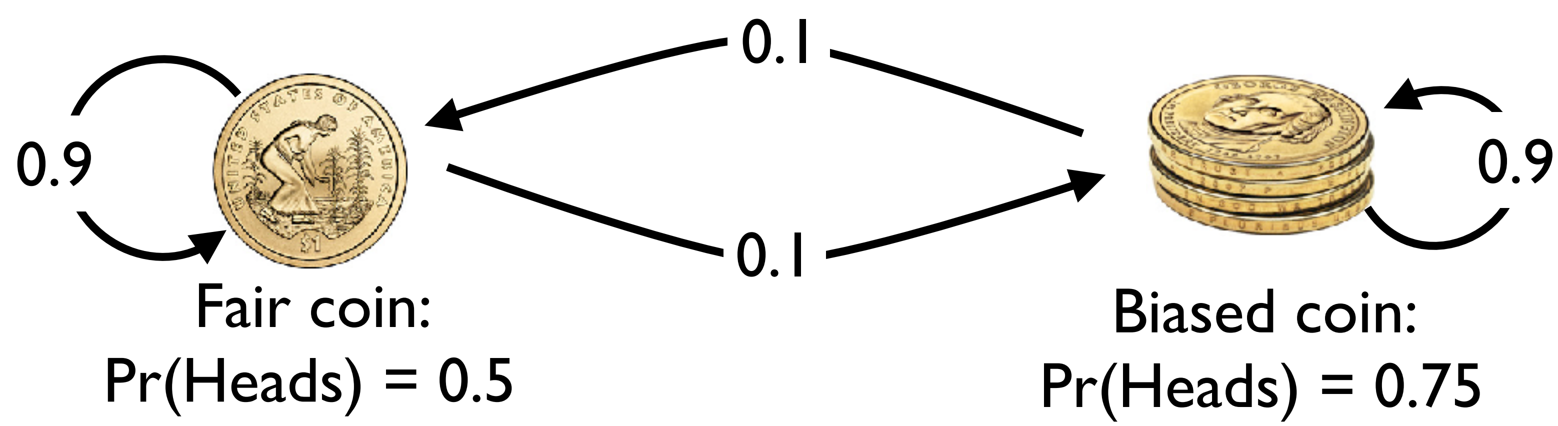
Walk the model and take transitions proportional to their probabilities — you get a stochastic output that is consistent with your model!

*

(Wikipedia, 2011)

Back to the casino: What if the casino can switch coins?

Fair coin: Pr(Heads) = 0.5
Biased coin: Pr(Heads) = 0.75
Probability of switching coins = 0.1

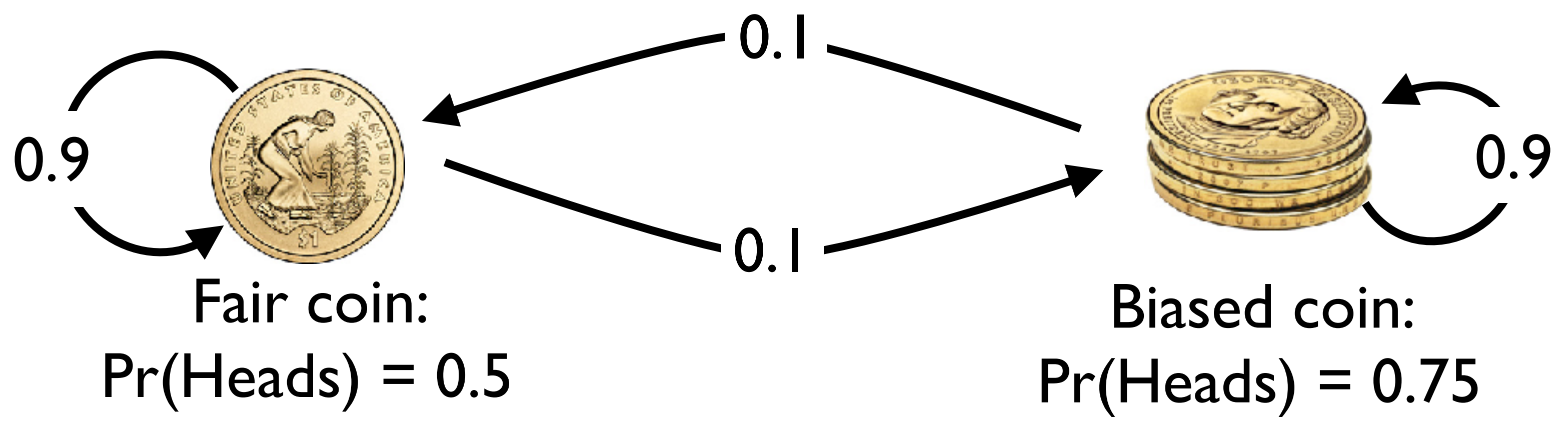


Looks like a Markov chain, but different. The "state" (i.e. "fair" or "biased") is not observed. **However**, we do observe output that depends, probabilistically, on the states (e.g. heads or tails). This is a **Hidden Markov Model** (HMM).

*

Typically, we're interested in questions involving these hidden states

Fair coin: Pr(Heads) = 0.5
Biased coin: Pr(Heads) = 0.75
Probability of switching coins = 0.1



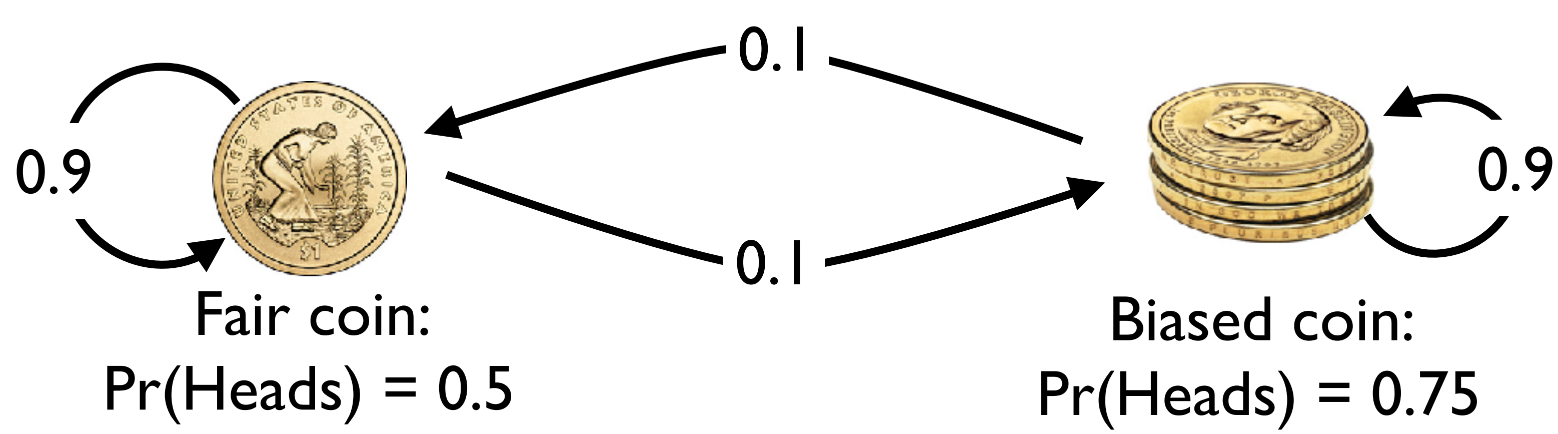
How can we compute the probability of the entire sequence?

How could we guess which coin was more likely **at each position**?

*

How can we compute the probability of the entire sequence?

Fair coin: Pr(Heads) = 0.5
Biased coin: Pr(Heads) = 0.75
Probability of switching coins = 0.1

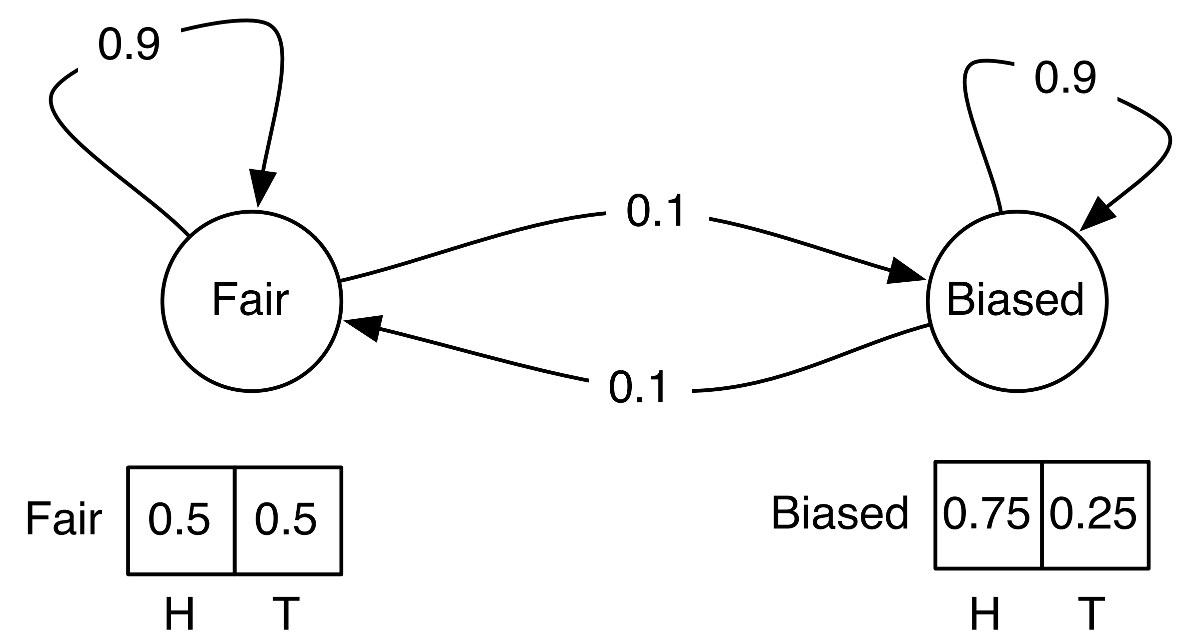


If we knew the set of hidden states, computing this would be easy!

*

What if the casino switches coins?

How can we compute the probability of the entire sequence?



$\Pr(\text{state}_x \text{ at } t \mid \text{state}_y \text{ at } t\text{-}1)$

$\Pr(\text{result} \mid \text{state})$

type of coin

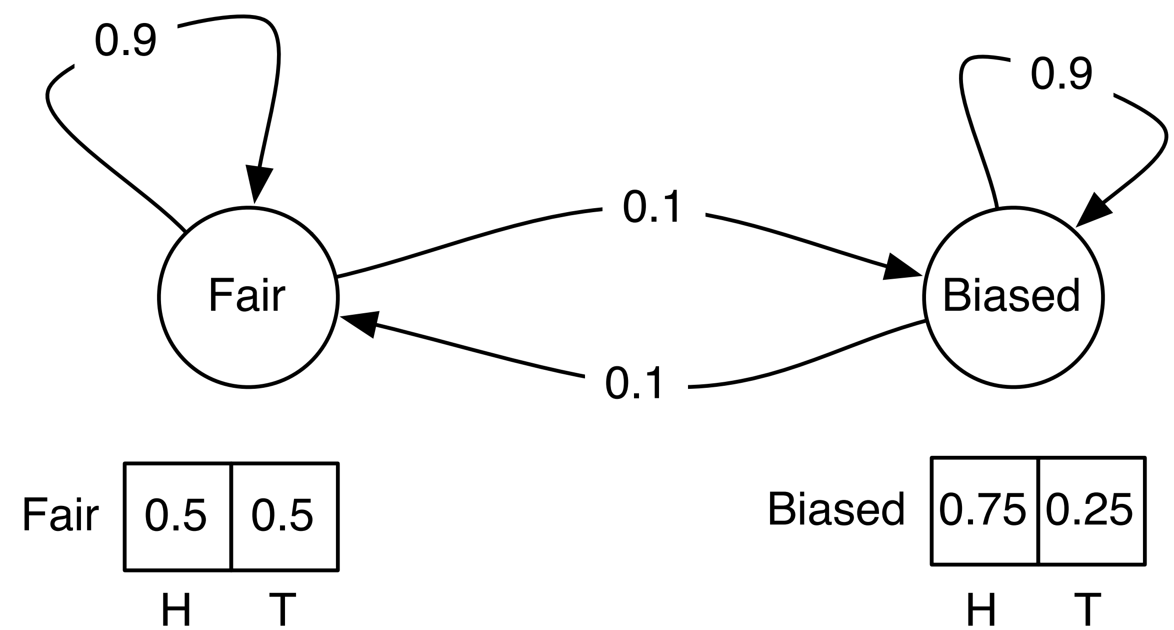
result of flip

$x =$	↓	↑	↓	↑	↑	↑	↓	↑	↑	↓
$\pi =$	F	F	F	B	B	B	B	F	F	F
$\Pr(x_i \mid \pi_i) =$	0.5	0.5	0.5	0.75	0.75	0.75	0.25	0.5	0.5	0.5
$\Pr(\pi_i \rightarrow \pi_{i+1}) =$	0.5	0.9	0.9	0.1	0.9	0.9	0.9	0.1	0.1	0.1

*

What if the casino switches coins?

How can we compute the probability of the entire sequence?



Pr(state$_x$ at t | state$_y$ at t-1)

Pr(result | state)

type of coin

result of flip

x =	$\downarrow$	$\uparrow$	$\downarrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\downarrow$	$\uparrow$	$\uparrow$	$\downarrow$
π =	F	F	F	B	B	B	B	F	F	F
$\Pr(x_i \mid \pi_i)$ =	0.5	0.5	0.5	0.75	0.75	0.75	0.25	0.5	0.5	0.5
$\Pr(\pi_i \rightarrow \pi_{i+1})$ =	0.1	0.9	0.9	0.1	0.9	0.9	0.9	0.1	0.1	0.1

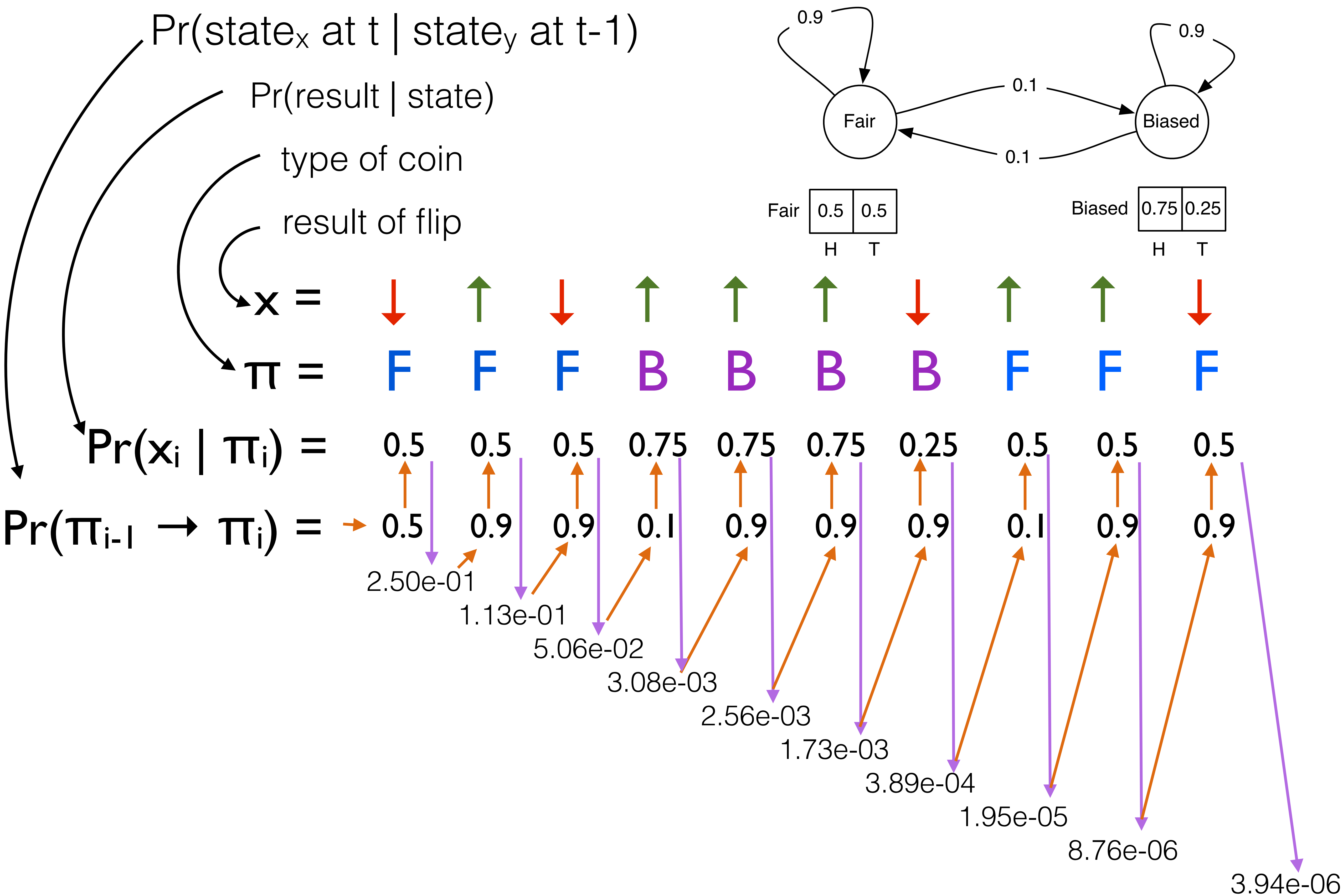
Pr(start with fair coin)

*

What if the casino switches coins?

*

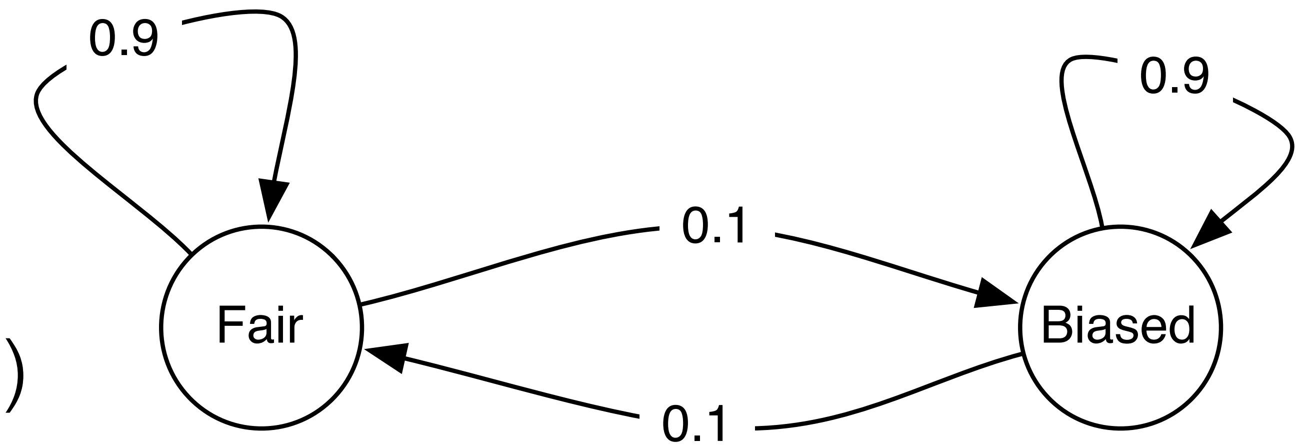
How can we compute the probability of the entire sequence?



What if the casino switches coins?

*

How can we compute the probability of the entire sequence?



Fair	0.5	0.5
	H	T

Biased	0.75	0.25
	H	T

Pr(state$_x$ at t | state$_y$ at t-1)

Pr(result | state)

type of coin

result of flip

x = ↓ ↑ ↓ ↑ ↑ ↑ ↓ ↑ ↑ ↓

π =

Pr(x$_i$ |) =

Pr(→) =

But, remember, we don't observe π in practice

What if the casino switches coins?

How can we compute the probability of the entire sequence?

We’ll return to how to tackle this later.

First, how is this related to Biology?

What does this have to do with biology?

Before:

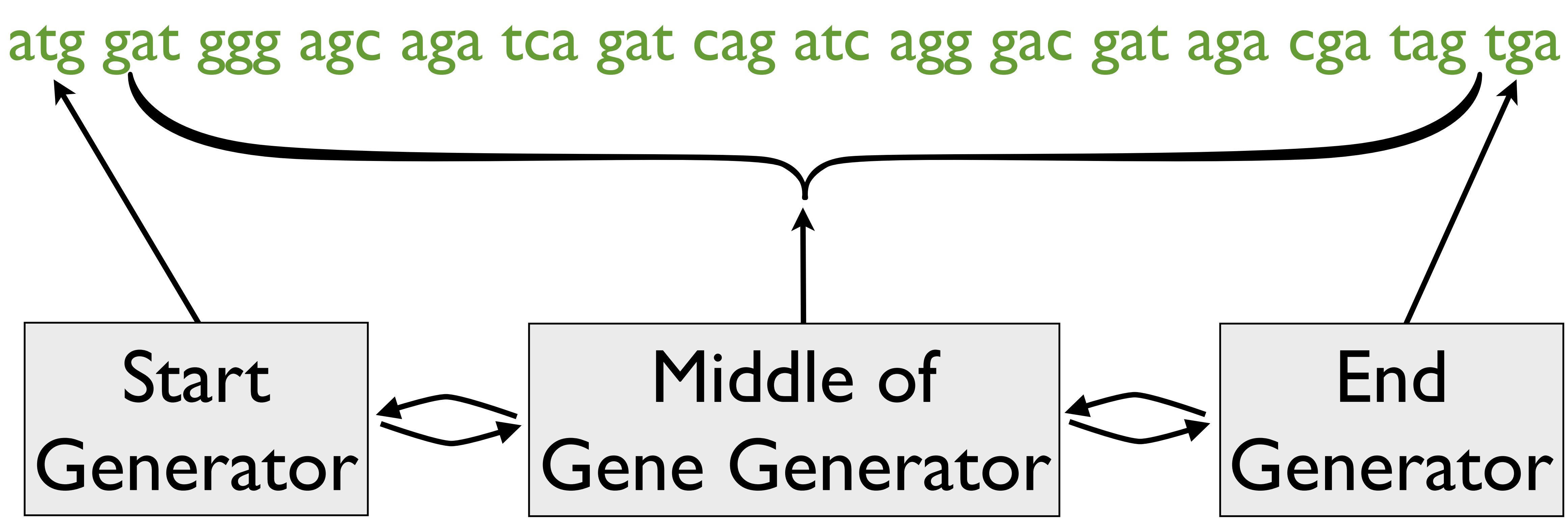
How likely is it that this sequence was generated by a fair coin?

Which parts were generated by a biased coin?

Now:

How likely is it that this is a gene?

Which parts are the start, middle and end?

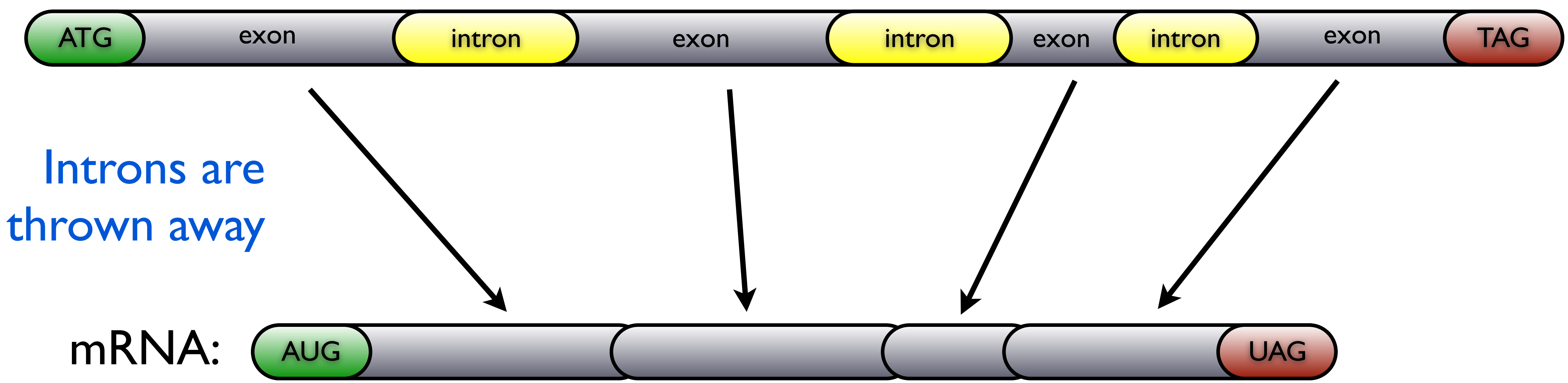


*

Eukaryotic Genes & Exon Splicing

Prokaryotic (bacterial) genes look like this:

Eukaryotic genes usually look like this:



Exons are concatenated together

This spliced RNA is what is translated into a protein.

*

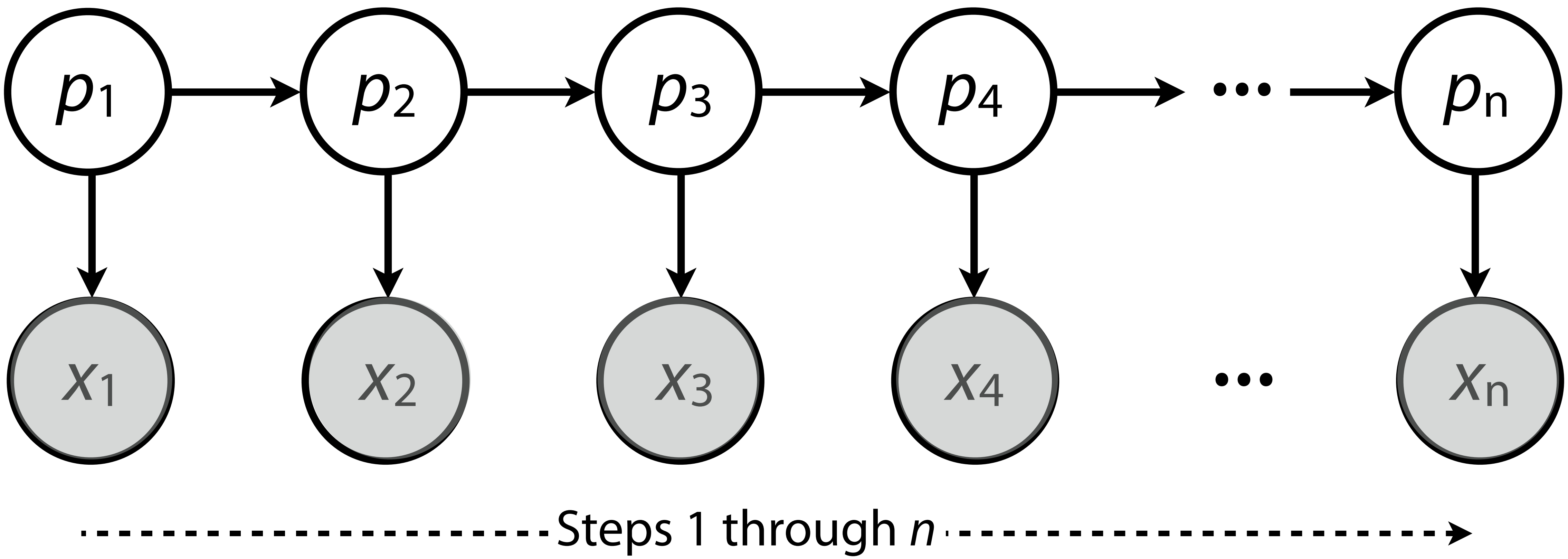
Eukaryotic Genes & Exon Splicing

Given this

Recover this

Under what sequence of “states” (exon, intron, start codon etc.) is the observed sequence of nucleotides maximized?

Hidden Markov Model (Think of this picture)

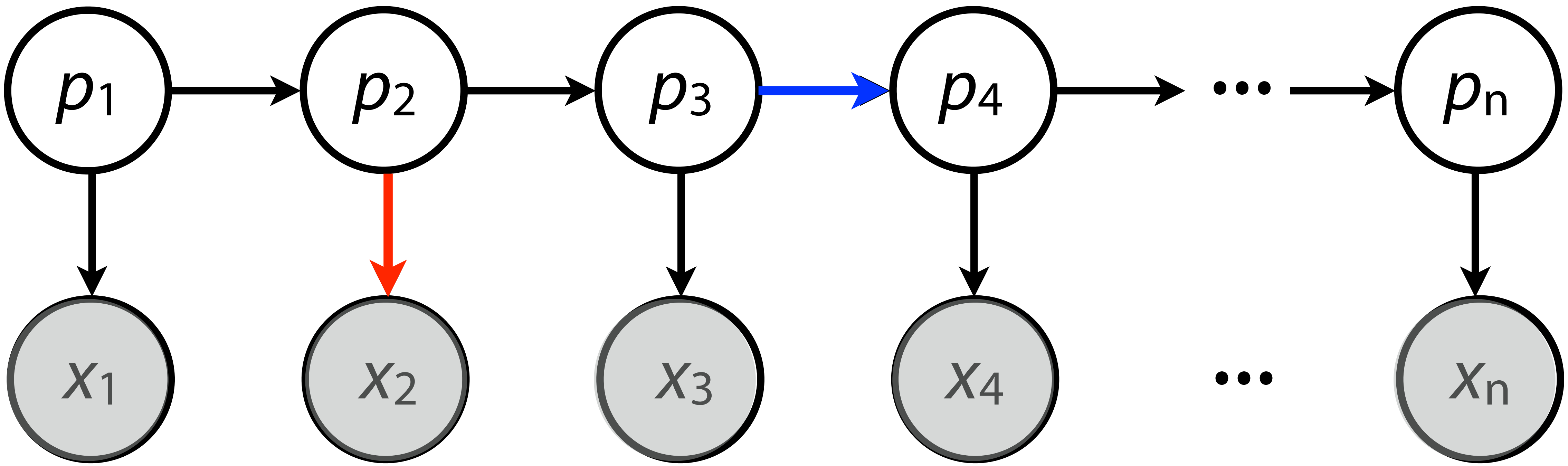


$p = \{ p_1, p_2, ..., p_n \}$ is a sequence of *states* (AKA a *path*). Each p_i takes a value from set Q. We ***do not*** observe p.

$x = \{ x_1, x_2, ..., x_n \}$ is a sequence of *emissions*. Each x_i takes a value from set Σ. We ***do*** observe x.

+ slide courtesy of Ben Langmead

Hidden Markov Model



Like for Markov chains, edges capture conditional independence:

x_2 is conditionally independent of everything else given p_2

p_4 is conditionally independent of everything else given p_3

Probability of being in a particular state at step *i* is known once we know what state we were in at step *i*-1. Probability of seeing a particular emission at step *i* is known once we know what state we were in at step *i*.

+ slide courtesy of Ben Langmead

Formal Definition of a HMM

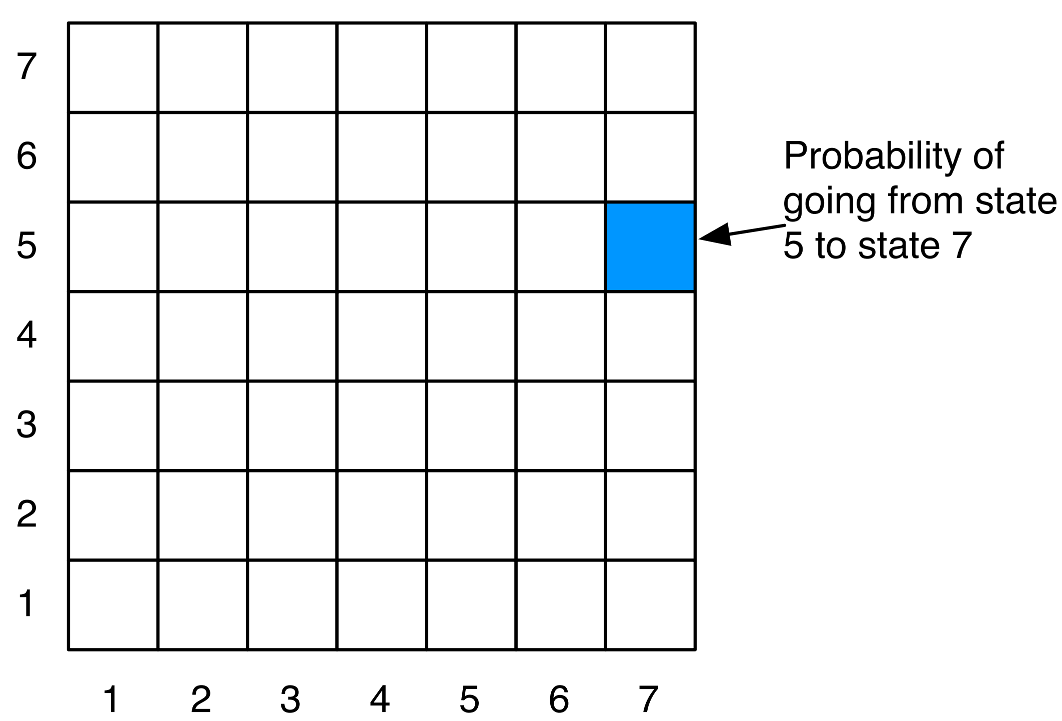
∑ = alphabet of symbols.

Q = set of states.

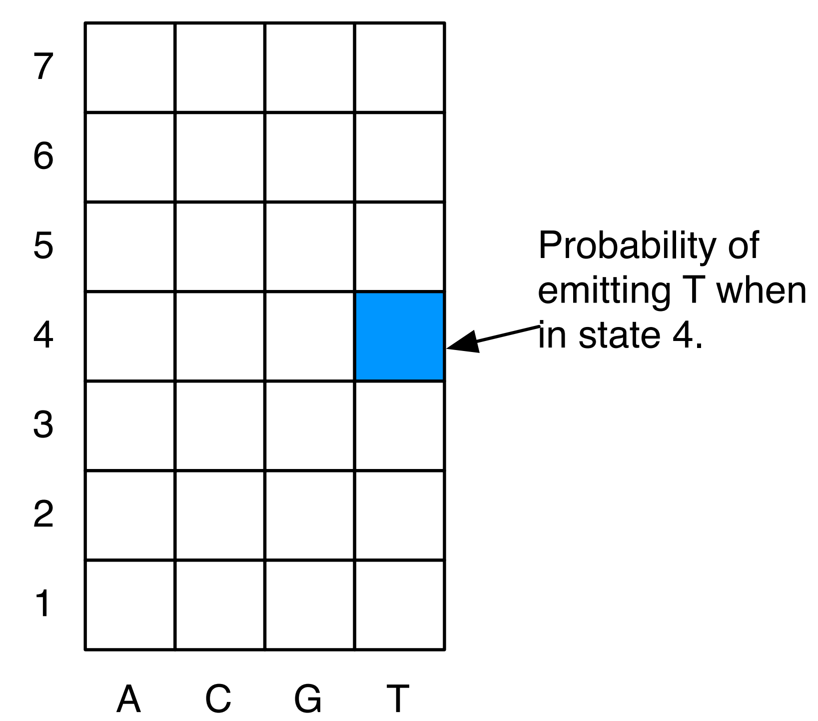
A = an |Q| x |Q| matrix where entry (k,l) is the probability of moving from state k to state l.

E = a |Q| x |∑| matrix, where entry (k,b) is the probability of emitting b when in state k.

A =

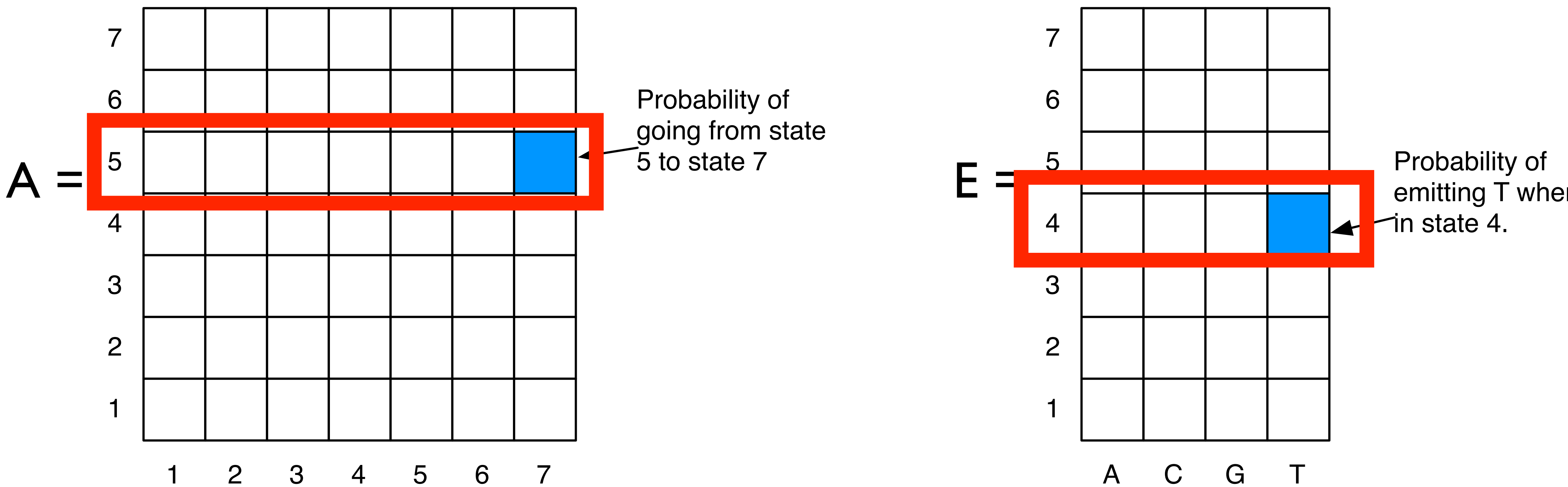


E =



*

Constraints on A and E



Sum of the # in each row must be 1.

*

The Decoding Problem

Given x and π, we can compute:

- $\Pr(x \mid \pi)$: product of $\Pr(x_i \mid \pi_i)$
- $\Pr(\pi)$: product of $\Pr(\pi_i \to \pi_{i+1})$
- $\Pr(x, \pi)$: product of all the $\Pr(x_i \mid \pi_i)$ and $\Pr(\pi_i \to \pi_{i+1})$

$$\Pr(x, \pi) = \Pr(\pi_0 \to \pi_1) \prod_{i=1}^{n} \Pr(x_i \mid \pi_i) \Pr(\pi_i \to \pi_{i+1})$$

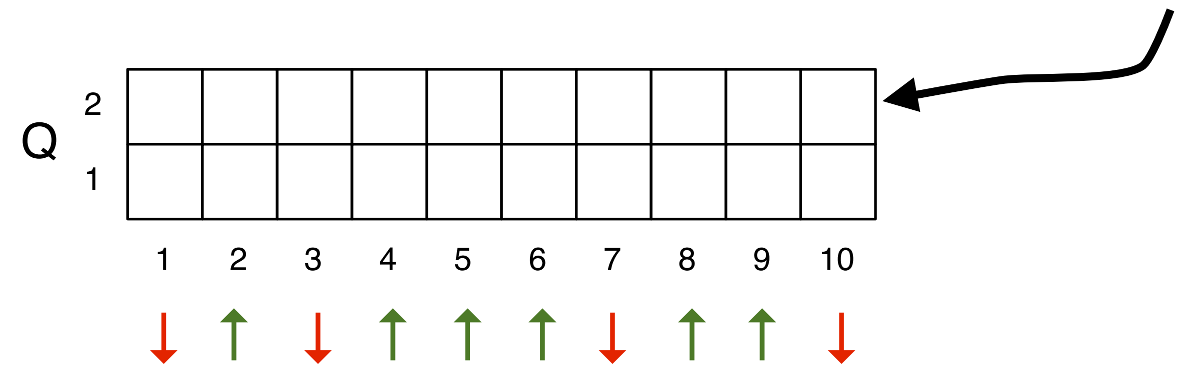
But they are "hidden" Markov models because π is unknown.

Decoding Problem: Given a sequence $x_1, x_2, x_3, \ldots x_n$ generated by an HMM (∑, Q, A, E), find a path π that maximizes $\Pr(x, \pi)$.

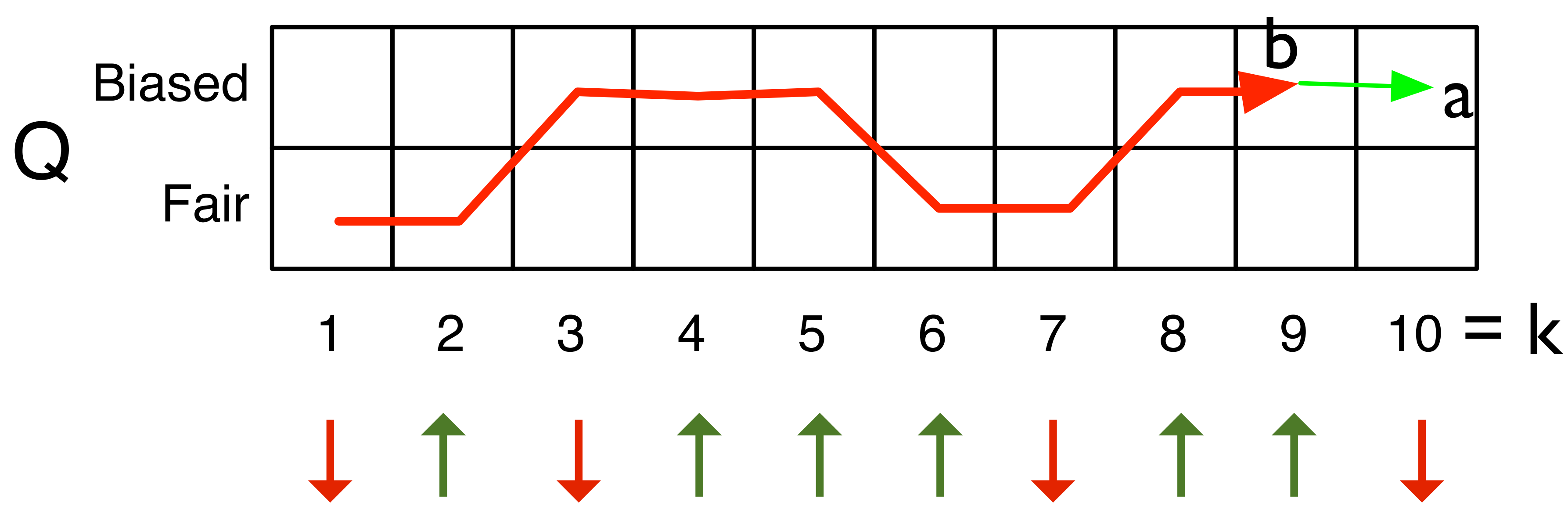
*

The Viterbi Algorithm to Find Best Path

A[a, k] := the probability of the **best** path for $x_1 \ldots x_k$ that ends at state a.

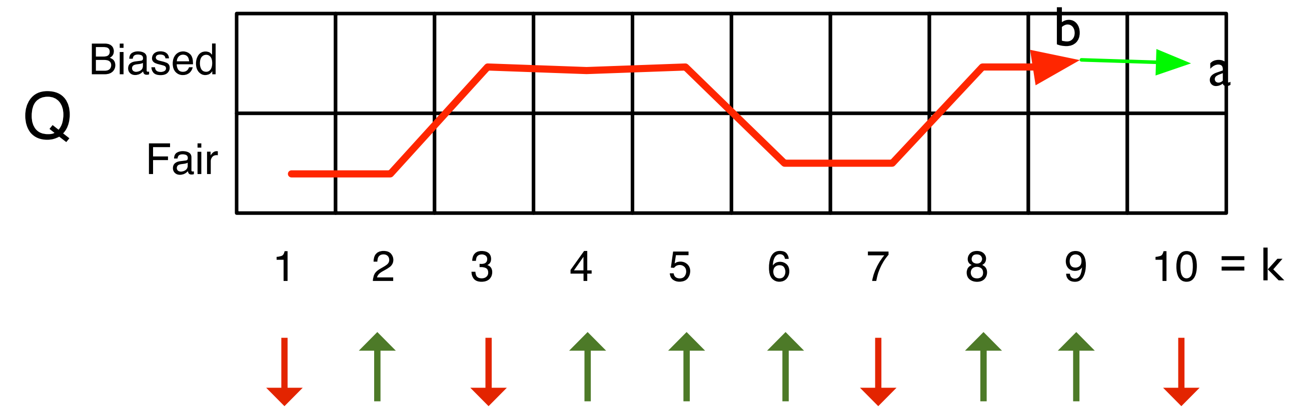


A[a, k] = the probability of the best path for $x_1 \ldots x_{k-1}$ that goes to some state *b* times probability of a transition from b to a, and then the probability to output x_k from state a.



*

Viterbi DP Recurrence



$$A[a,k] = \max_{b \in Q} \{ \underbrace{A[b,k-1]}_{\text{Best path for } x_1..x_k \text{ ending in state } b} \times \underbrace{\Pr(b \rightarrow a)}_{\text{Probability of transitioning from state } b \text{ to state } a} \times \underbrace{\Pr(x_k \mid \pi_k = a)}_{\text{Probability of outputting } x_k \text{ given that the } k^{th} \text{ state is } a.} \}$$

Over all possible previous states.

Best path for $x_1..x_k$ ending in state *b*

Probability of transitioning from state *b* to state *a*

Probability of outputting x_k given that the k^{th} state is *a*.

Base case:

$$A[a,1] = \underbrace{\Pr(\pi_1 = a)}_{\text{Probability that the first state is } a} \times \underbrace{\Pr(x_1 \mid \pi_1 = a)}_{\text{Probability of emitting } x_1 \text{ given the first state is } a.}$$

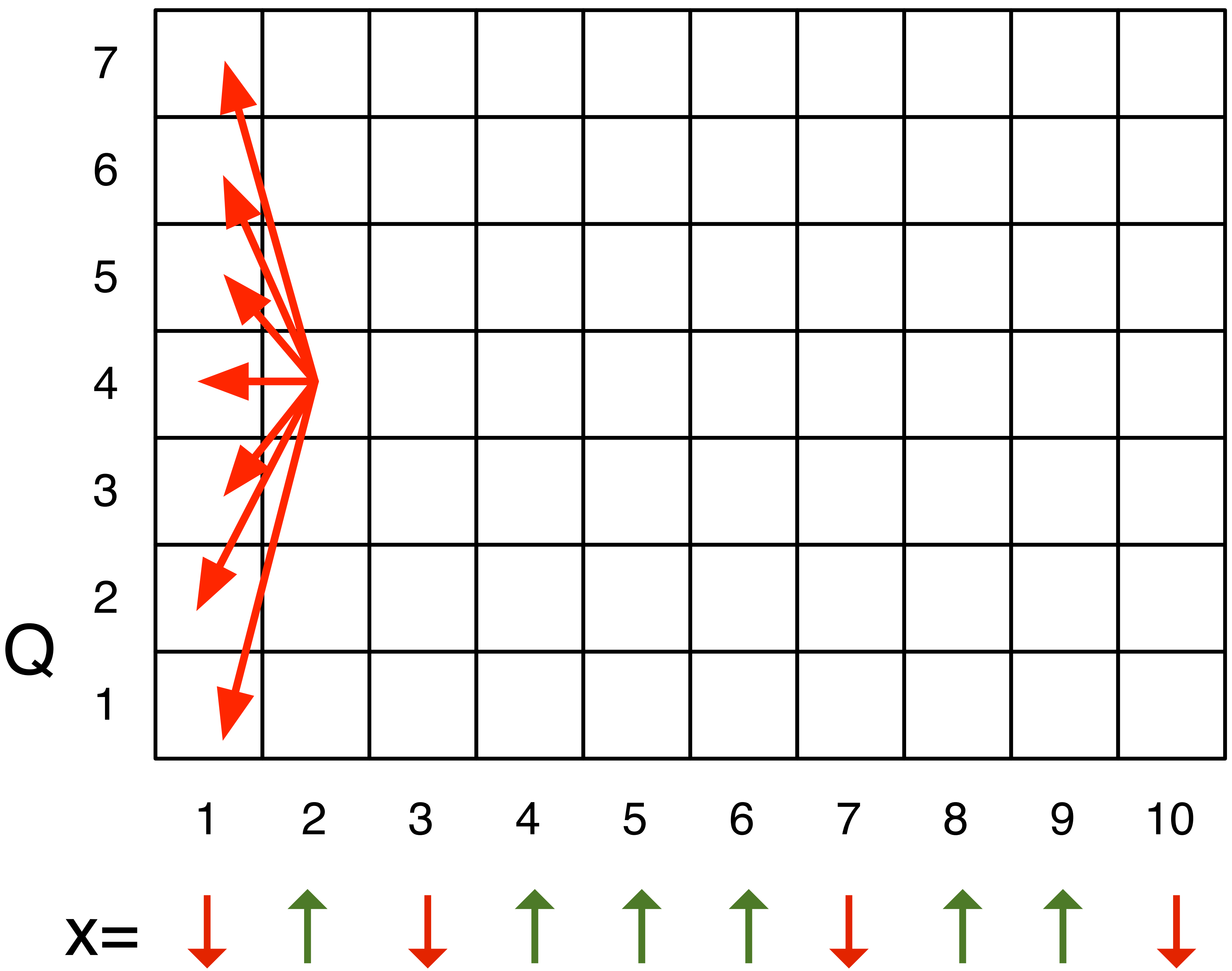
Probability that the first state is *a*

Probability of emitting x_1 given the first state is *a*.

Generally, our model can include an "initial"/ starting distribution

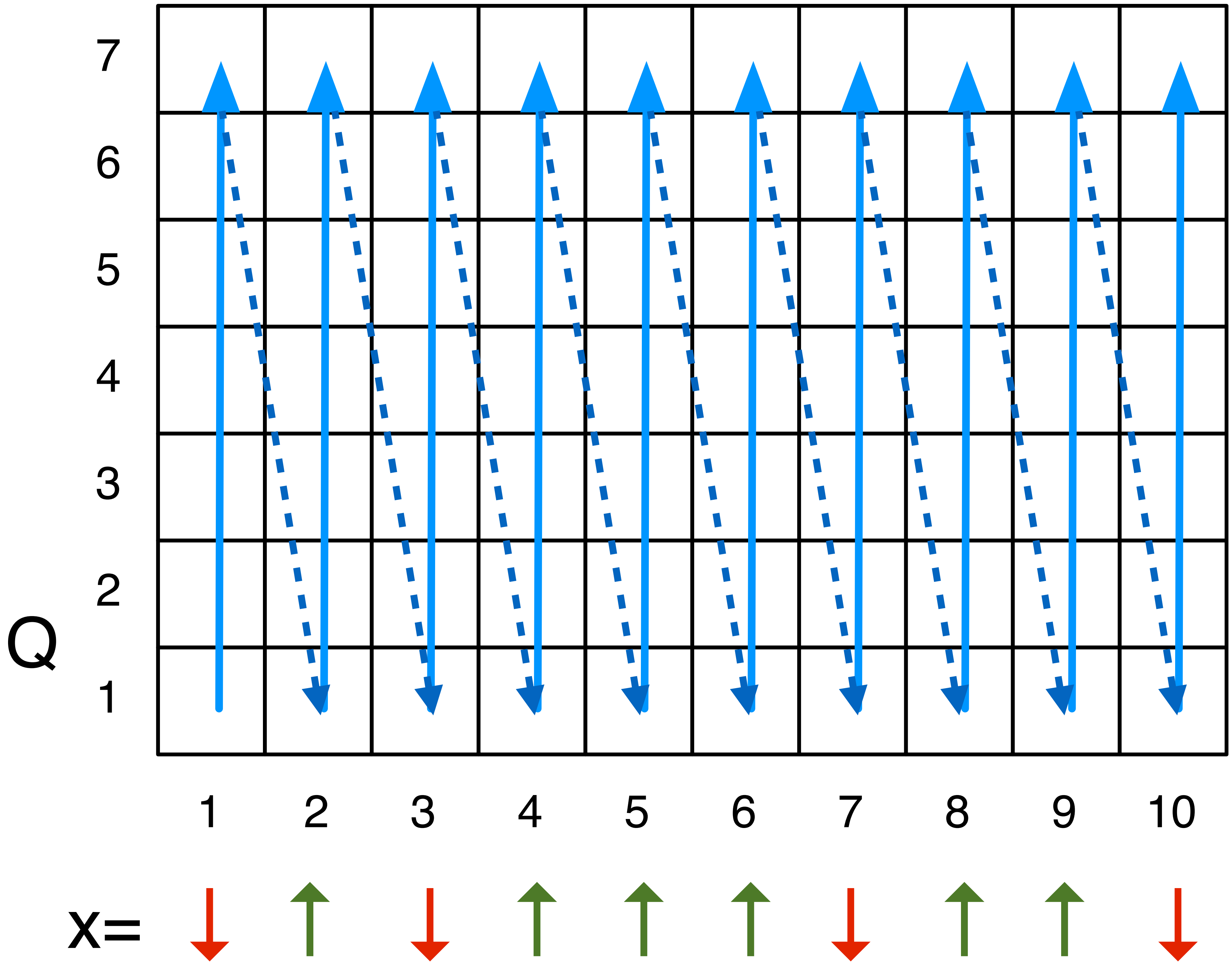
*

Which Cells Do We Depend On?



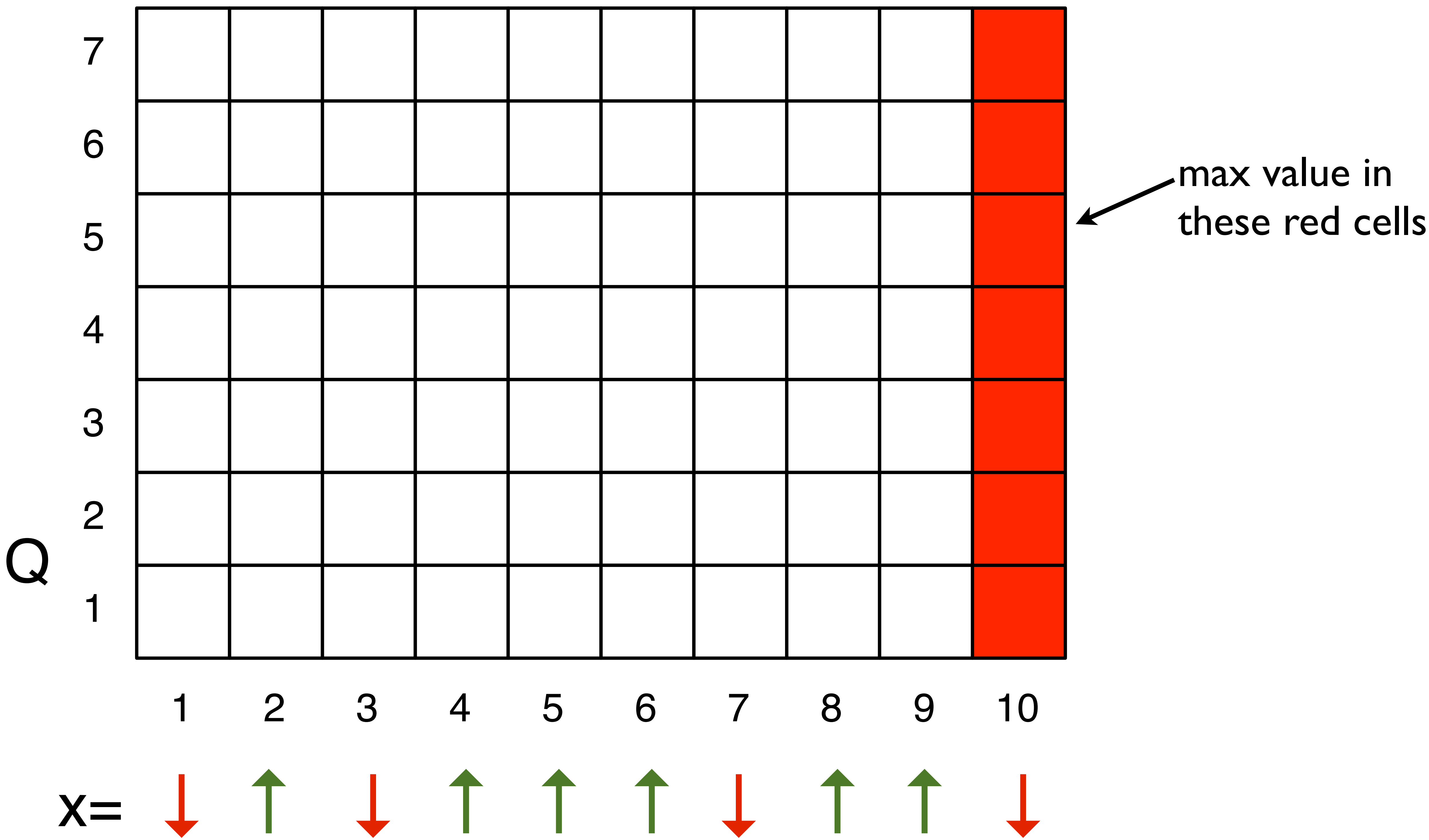
*

Order to Fill in the Matrix:



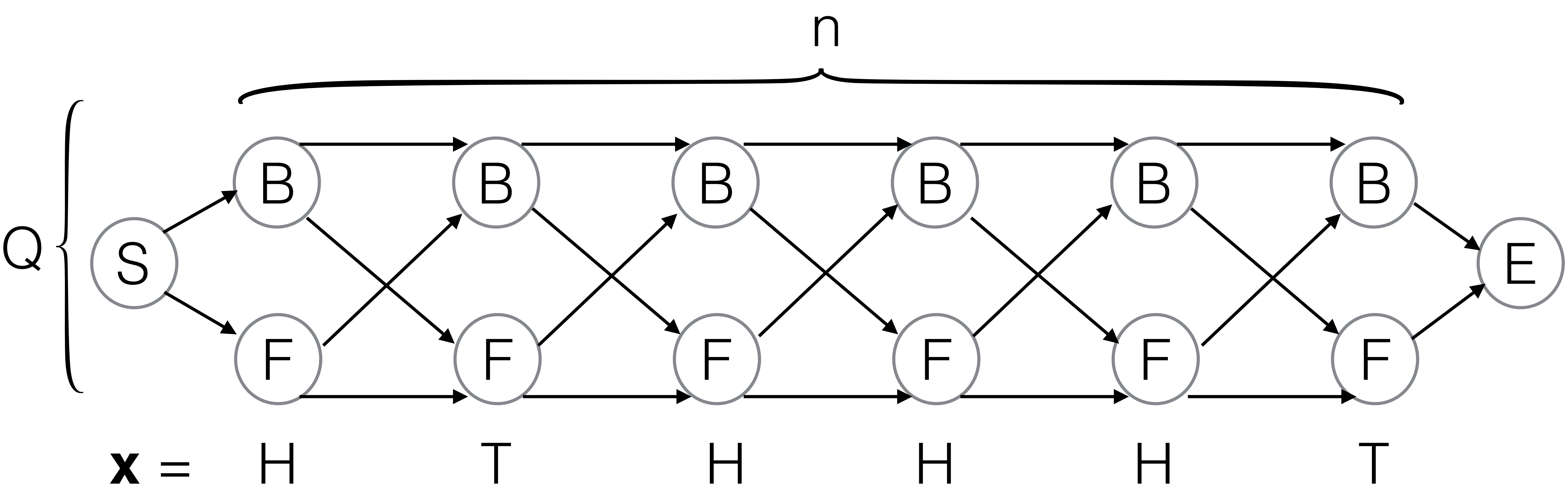
*

Where's the answer?



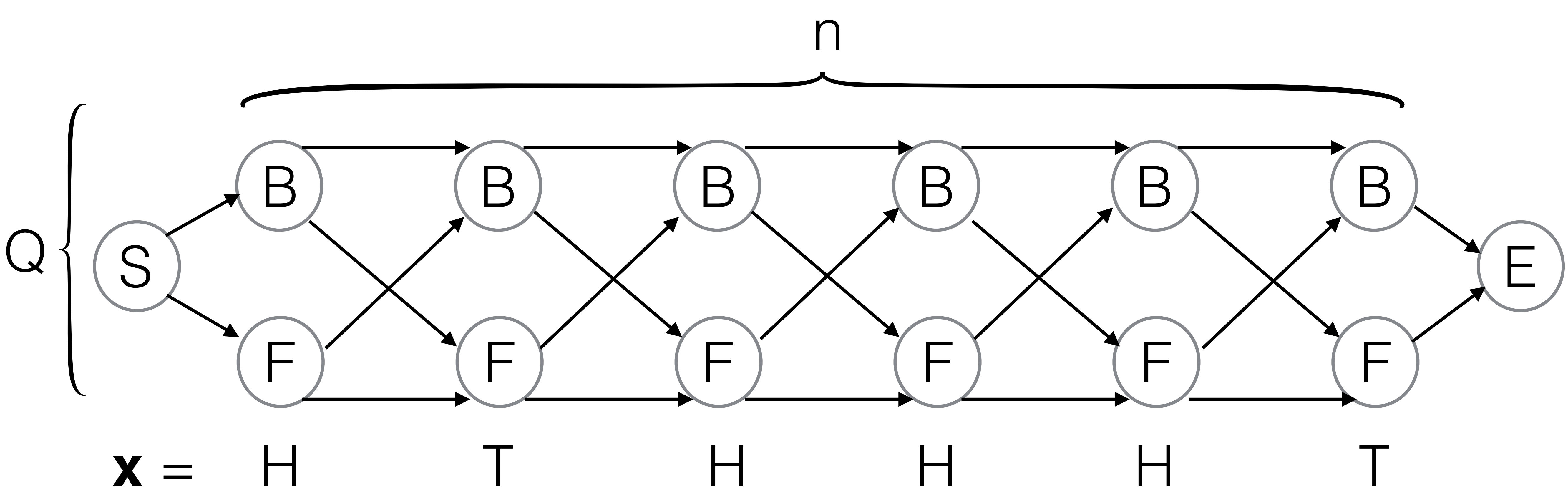
*

Trellis Graph



The trellis graph “unfolds” the states of the HMM over (discrete) time.

Trellis Graph



Finding the maximum probability path through the trellis graph can be accomplished efficiently with the Viterbi algorithm — think back to lecture 3

DAG View of Dynamic Programming

The formulation of a DP as traversal of a DAG is a very powerful framework for thinking about and implementing different DPs.

1. topological sort
2. visit each vertex in the topological ordering and do updates

The pseudo-code of the Viterbi algorithm is presented in Algorithm 1.

Algorithm 1 Viterbi Algorithm.

```
procedure VITERBI(G, w, s)
    topologically sort the vertices of G
    INITIALIZE(G, s)
    for each vertex v in topological order do
        for each edge e = (u, v) in BS(v) do
            d(v) ⊕= d(u) ⊗ w(e)
```

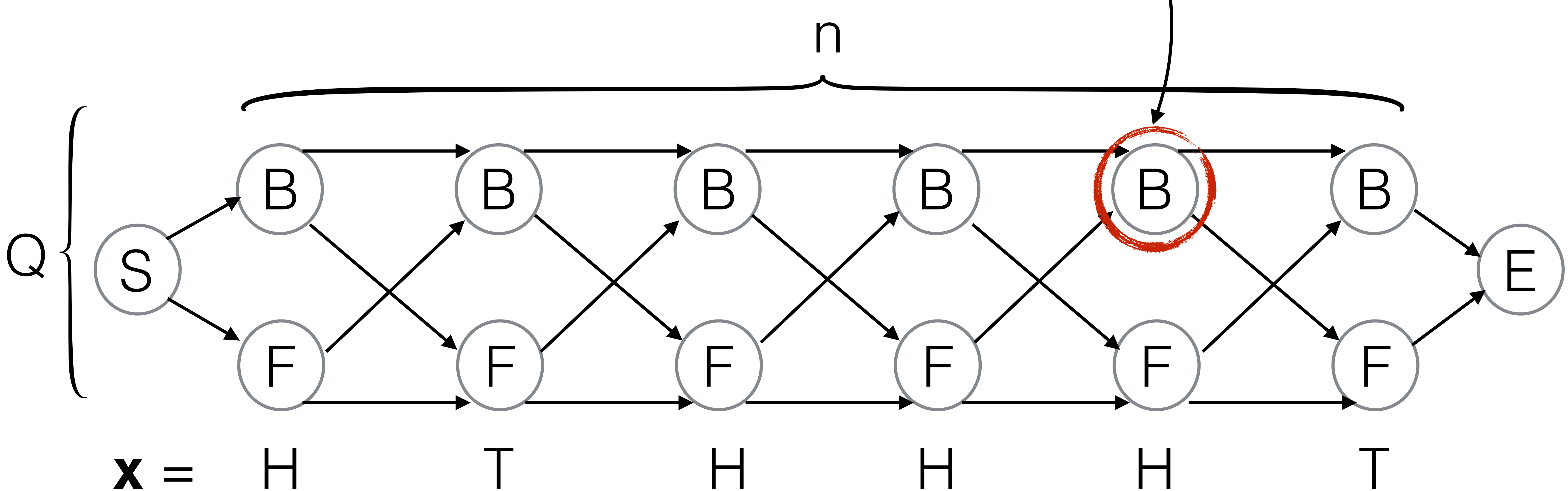
Semiring	Set	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$	intuition/application
Boolean	$\{0, 1\}$	$\vee$	$\wedge$	0	1	logical deduction, recognition
Viterbi	$[0, 1]$	max	$\times$	0	1	prob. of the best derivation
Inside	$\mathbb{R}^+ \cup \{+\infty\}$	$+$	$\times$	0	1	prob. of a string
Real	$\mathbb{R} \cup \{+\infty\}$	min	$+$	$+\infty$	0	shortest-distance
Tropical	$\mathbb{R}^+ \cup \{+\infty\}$	min	$+$	$+\infty$	0	with non-negative weights
Counting	$\mathbb{N}$	$+$	$\times$	0	1	number of paths

Table 1: Examples of semirings

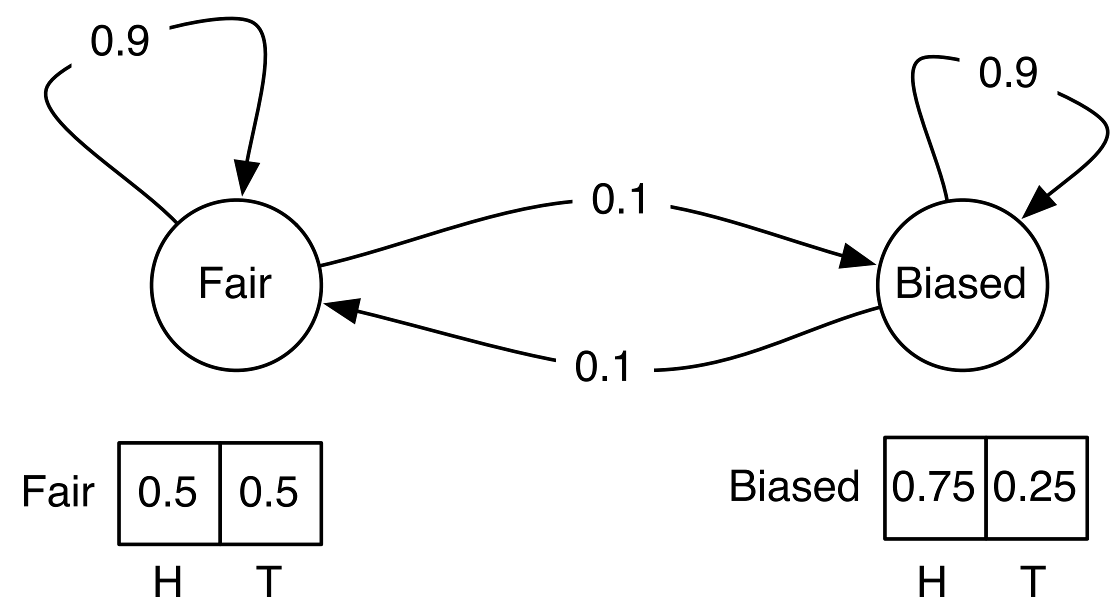
Huang, Liang. "Dynamic programming algorithms in semiring and hypergraph frameworks." Qualification Exam Report (2006): 1-19.

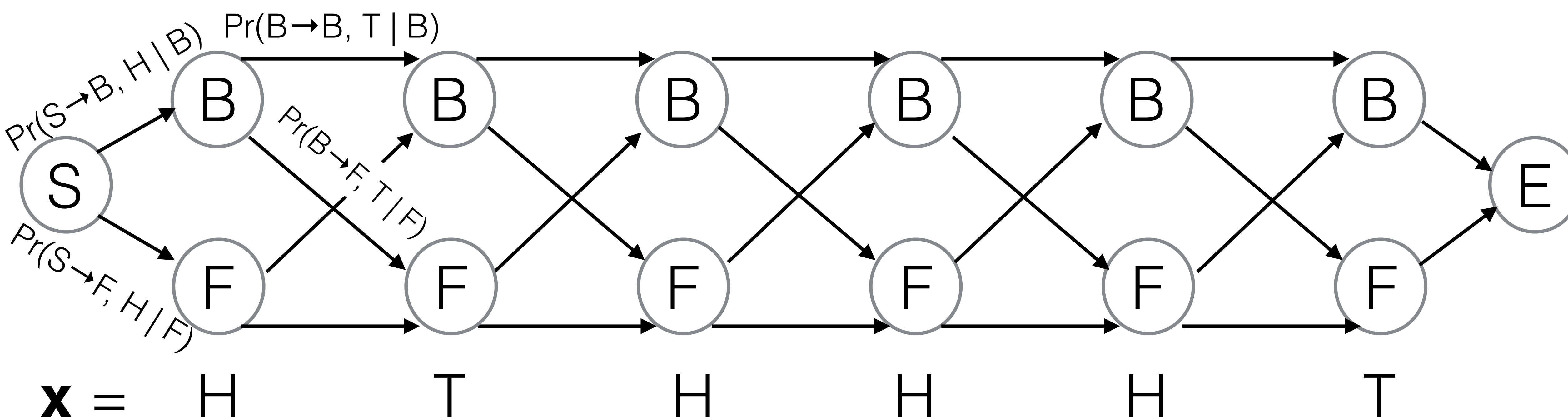
Trellis Graph

When we want to compute the prob. of the best path ending here, we already have the prob. of the best path at all predecessors, as well as the conditional prob. of each incoming edge

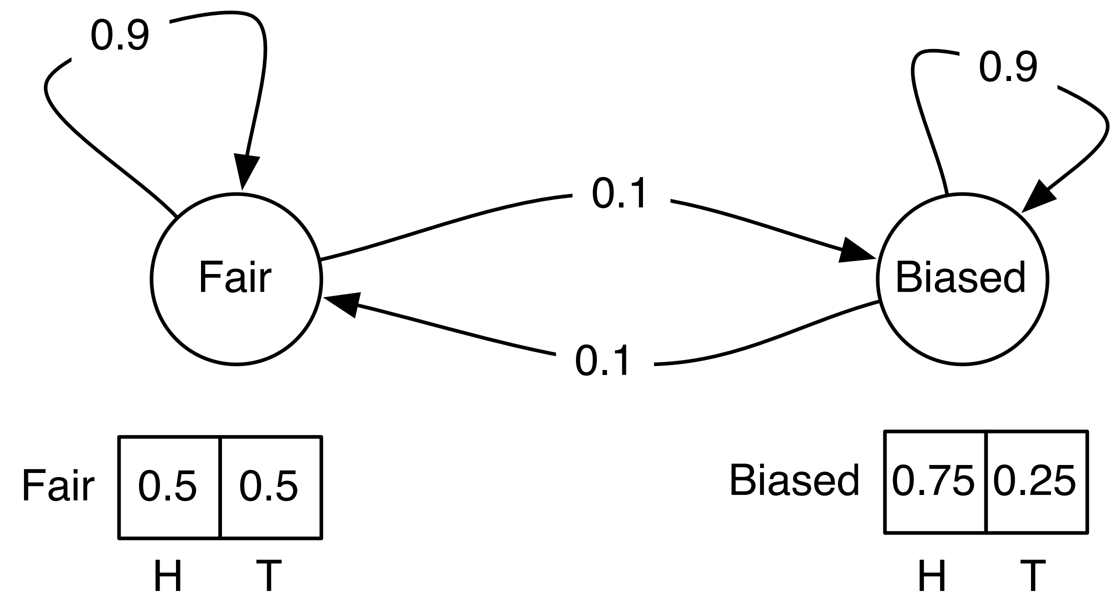


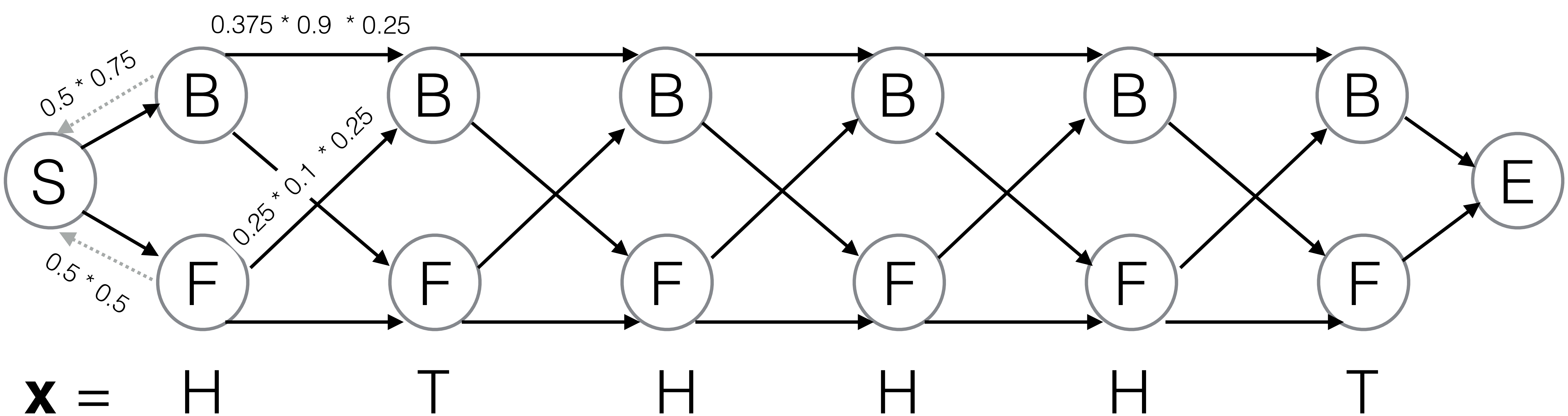
Graph View of Viterbi



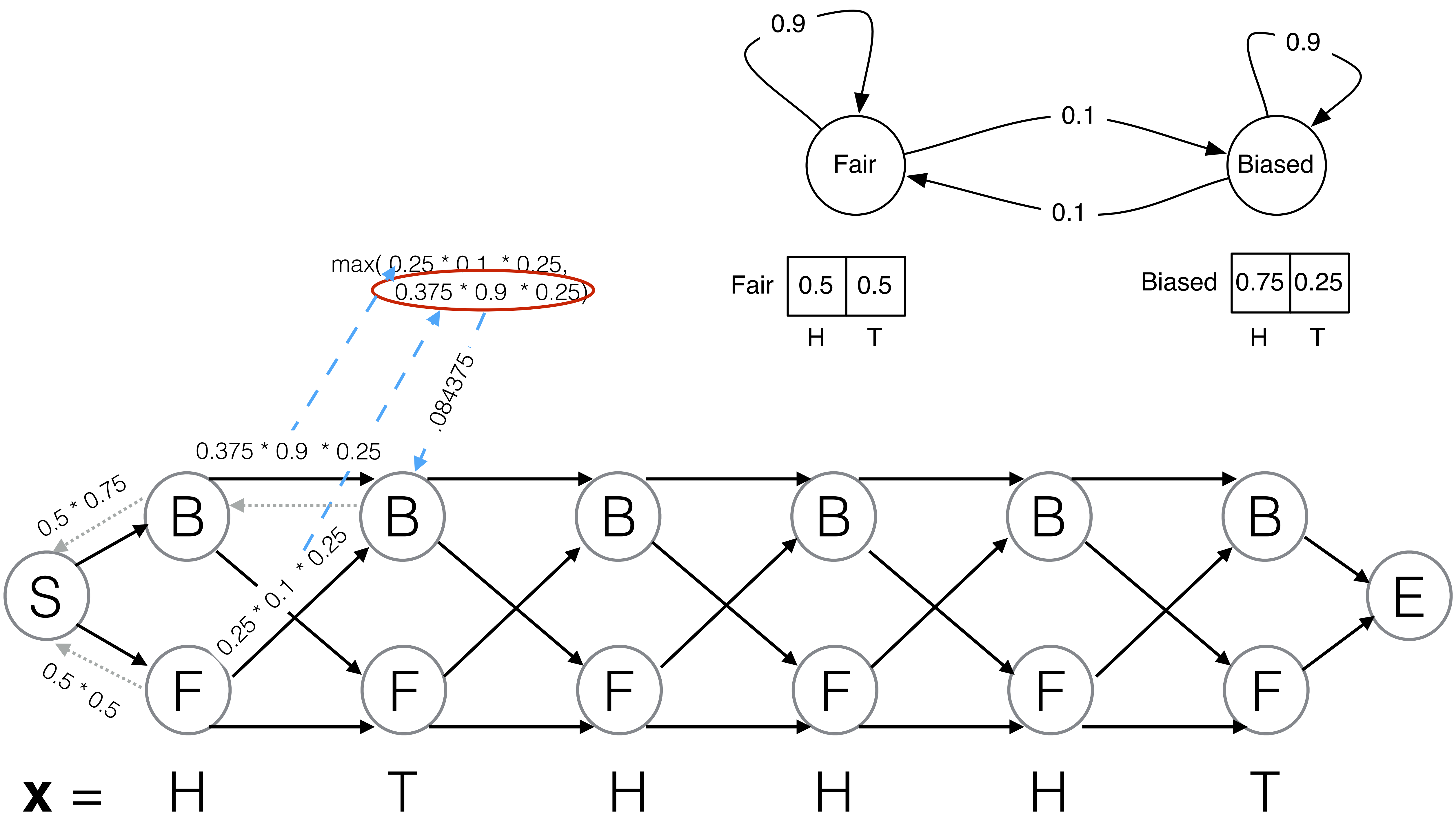


Graph View of Viterbi

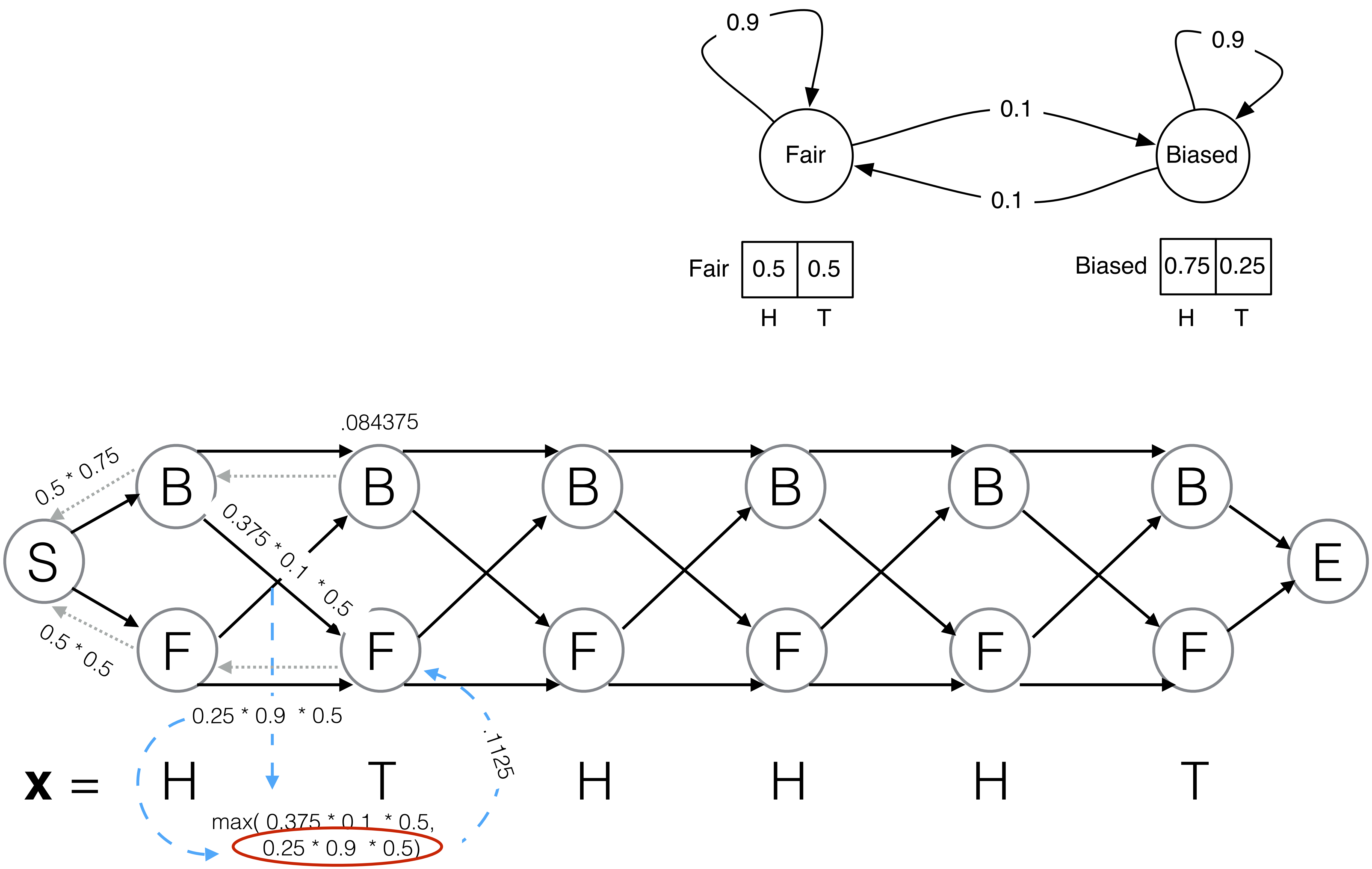




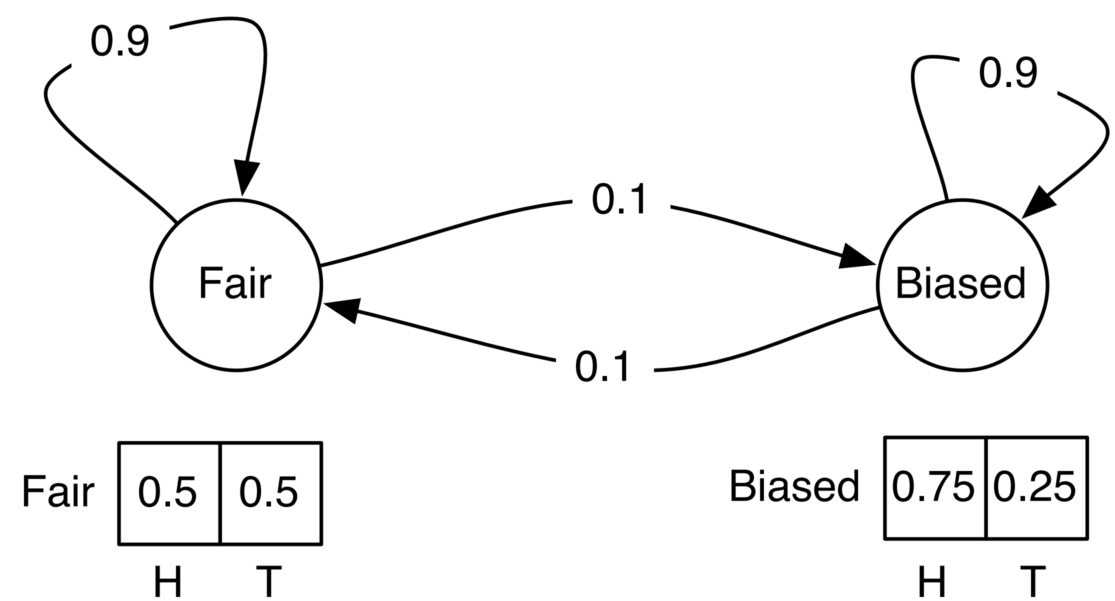
Graph View of Viterbi

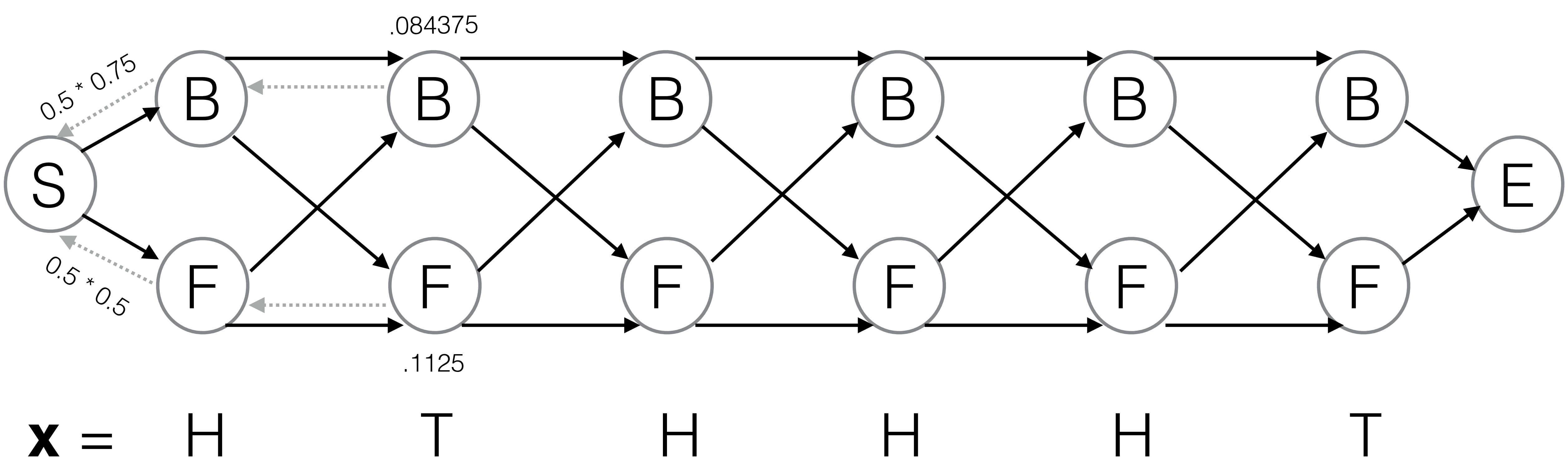


Graph View of Viterbi

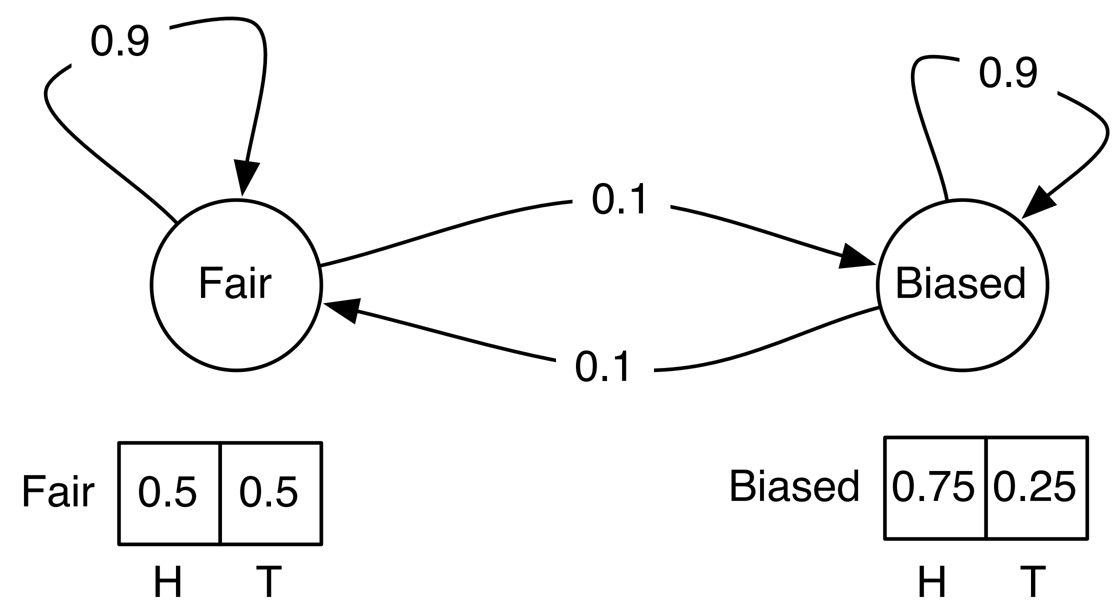


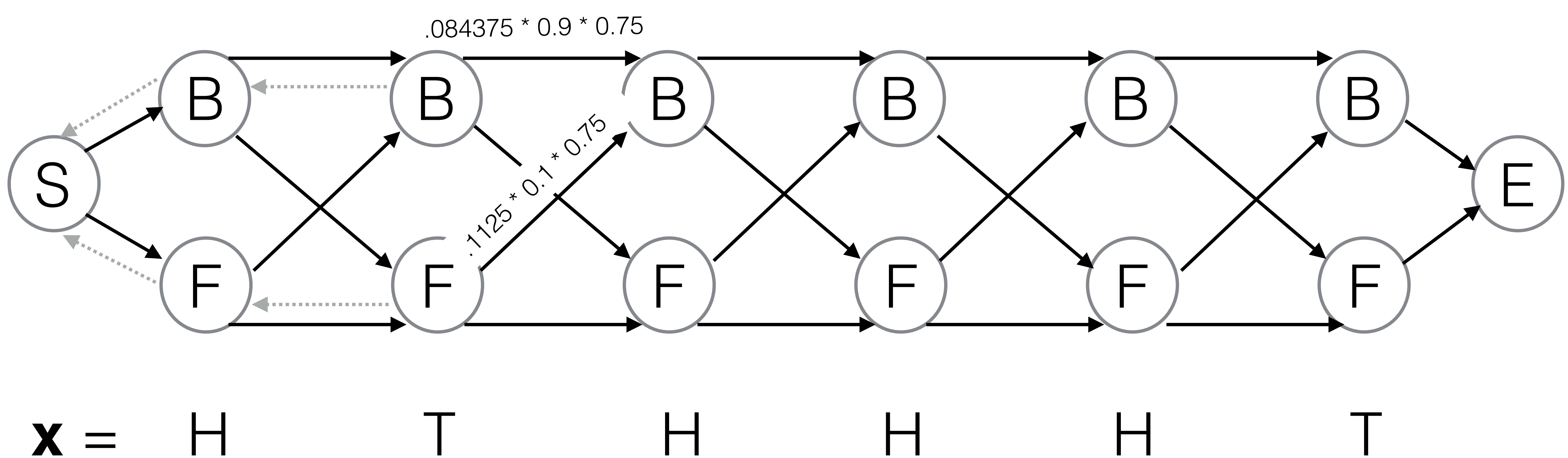
Graph View of Viterbi



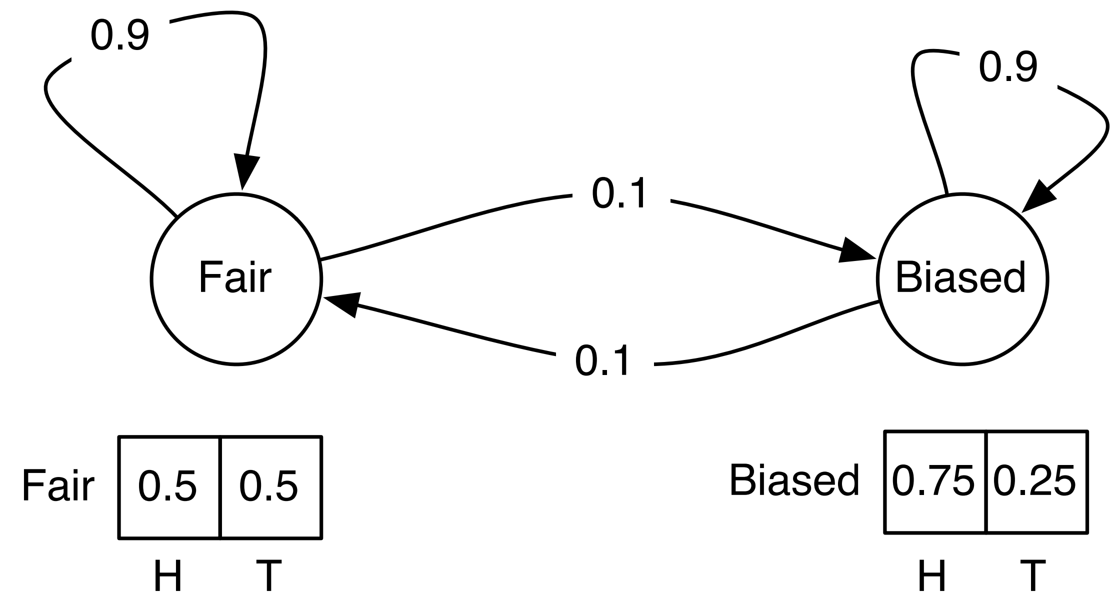


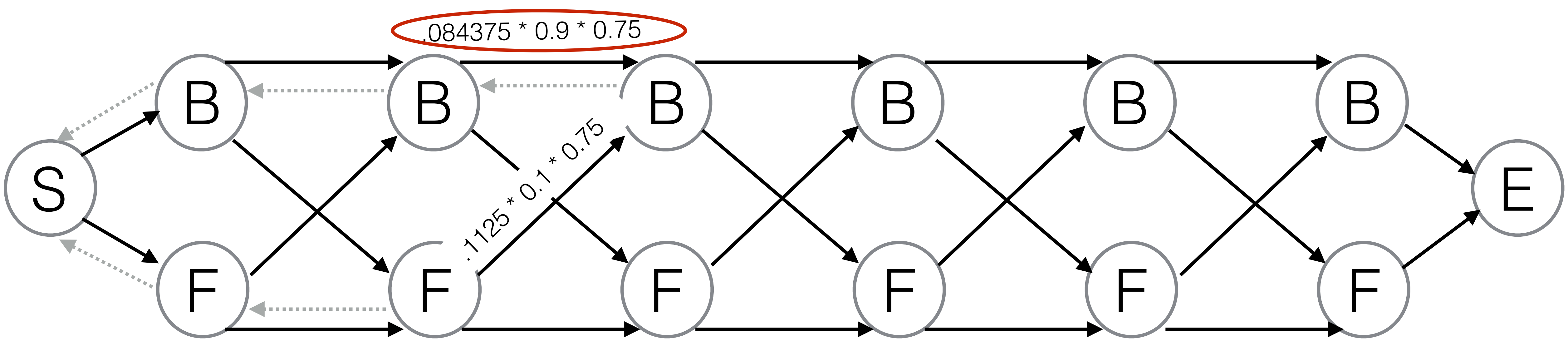
Graph View of Viterbi





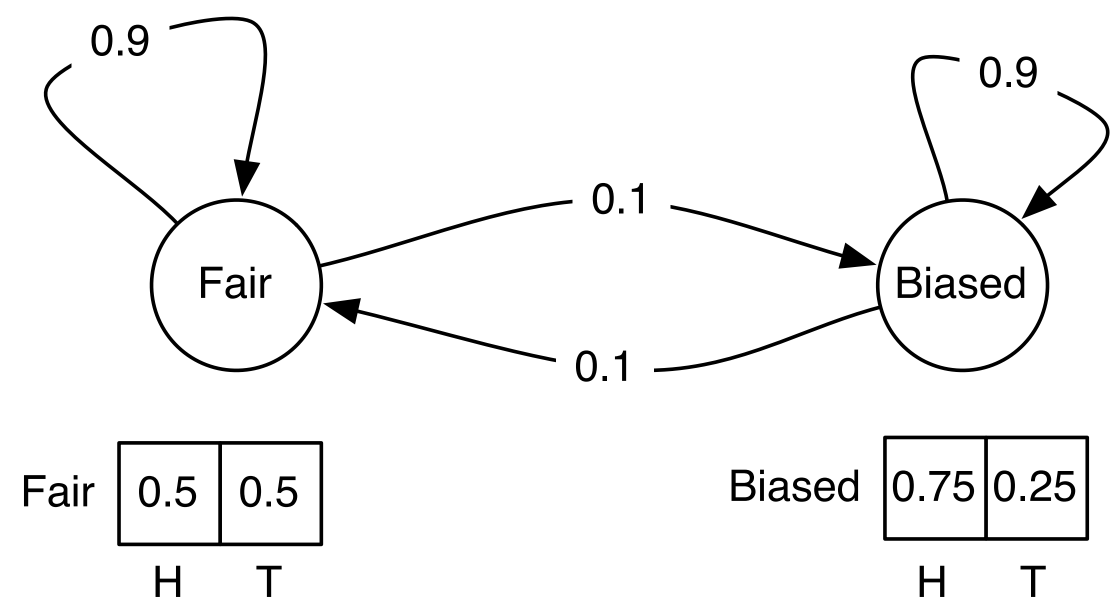
Graph View of Viterbi

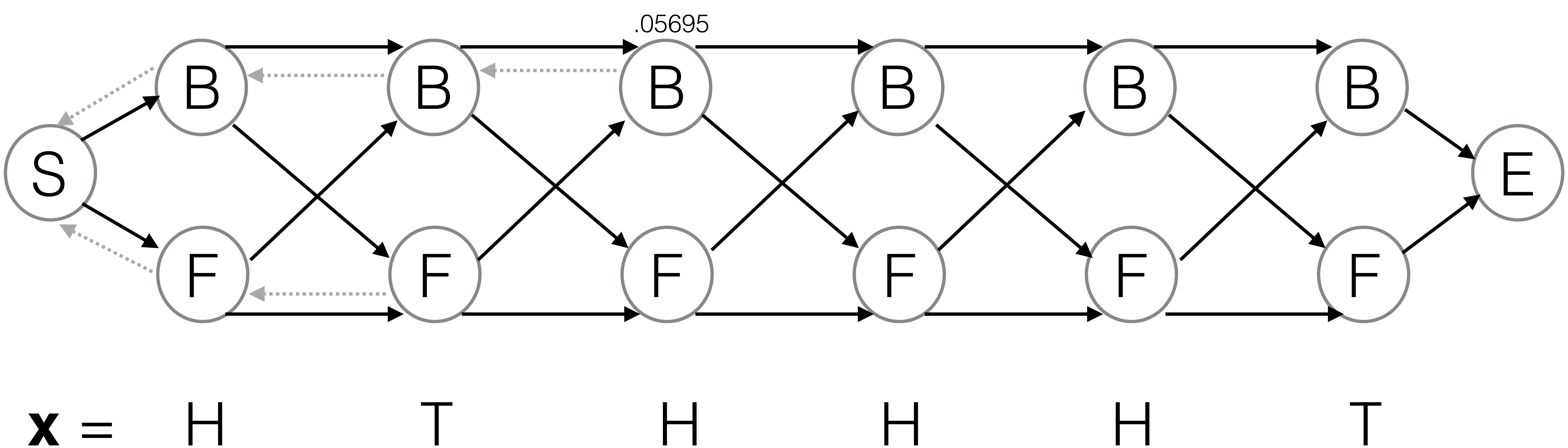




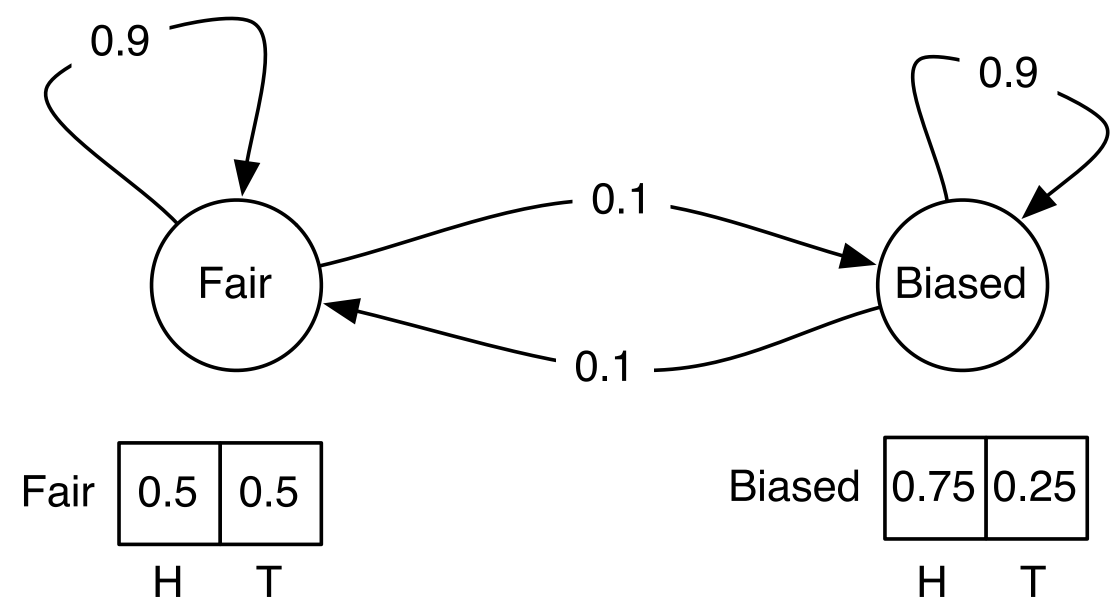
x = H T H H H T

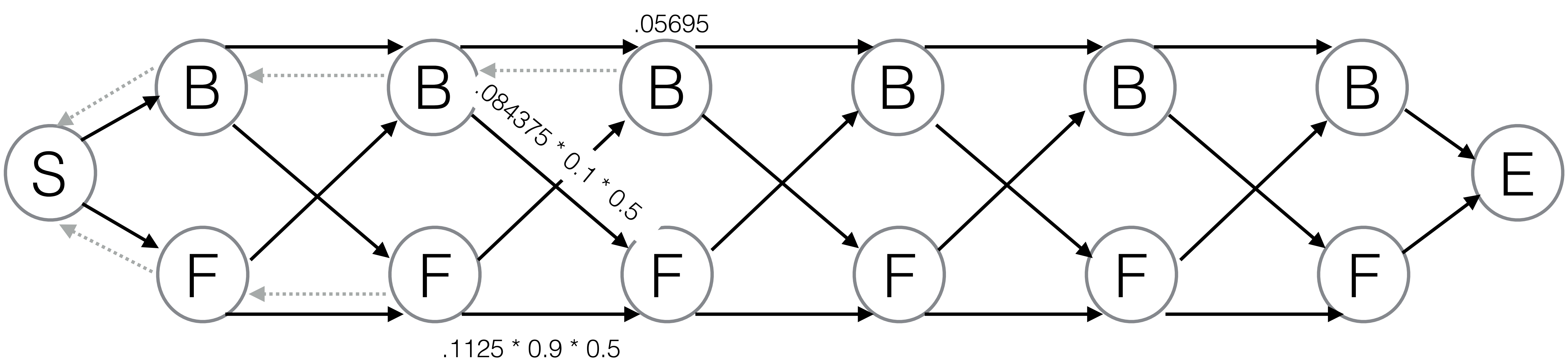
Graph View of Viterbi





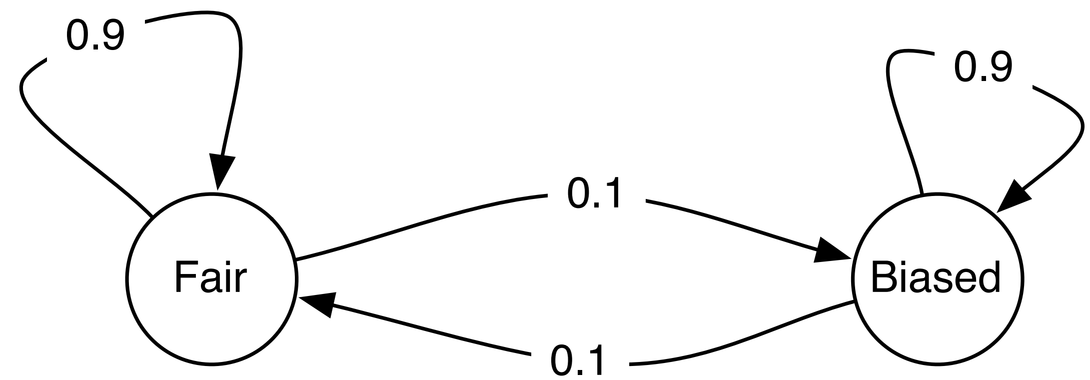
Graph View of Viterbi

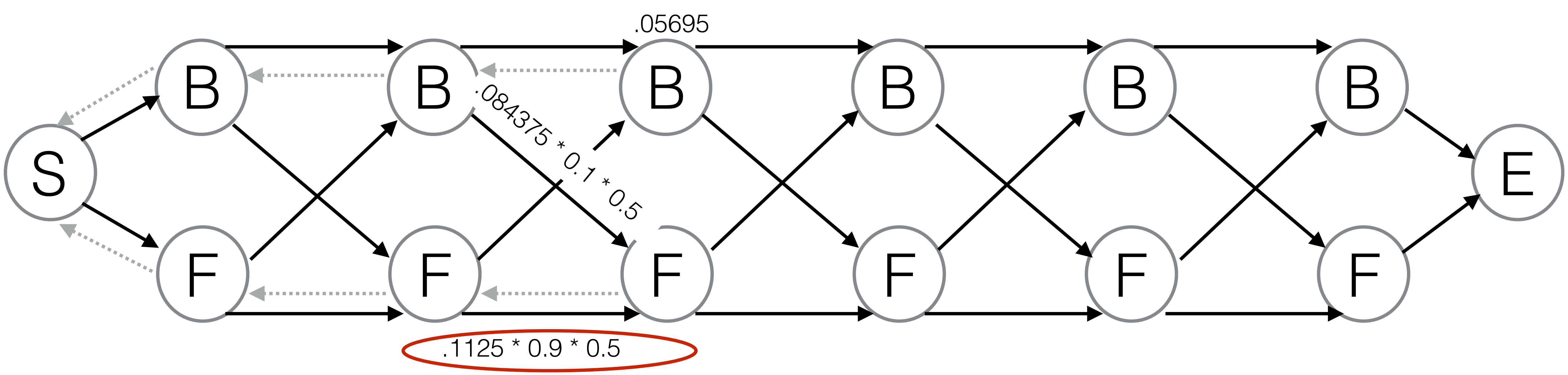




x = H T H H H T

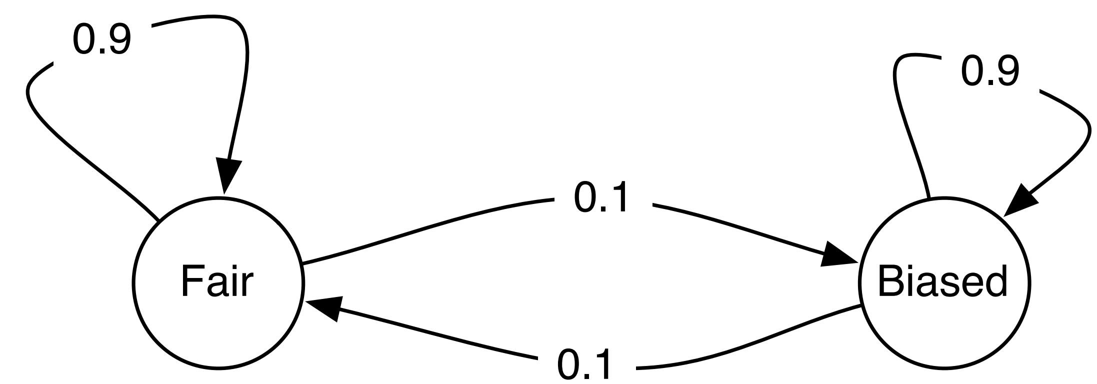
Graph View of Viterbi



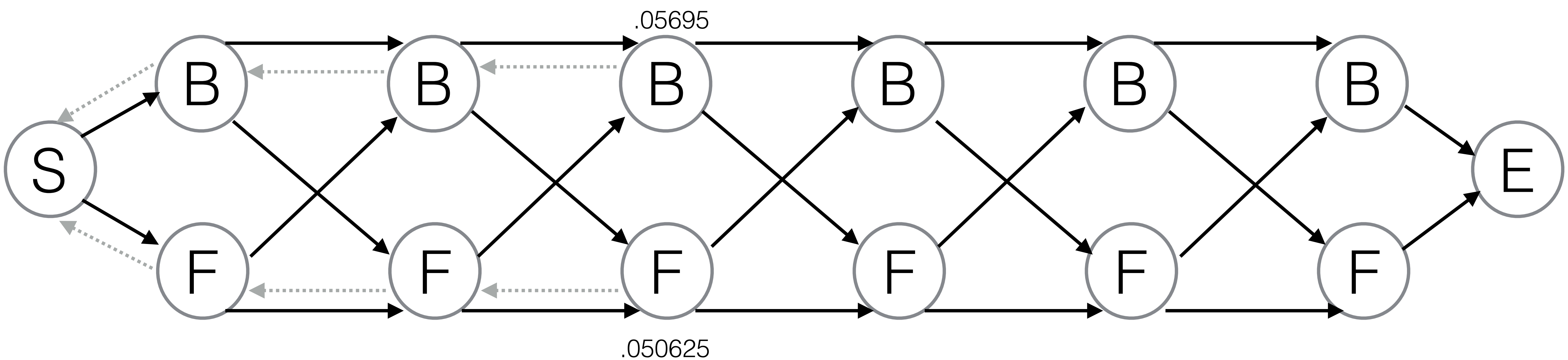


x = H T H H H T

Graph View of Viterbi

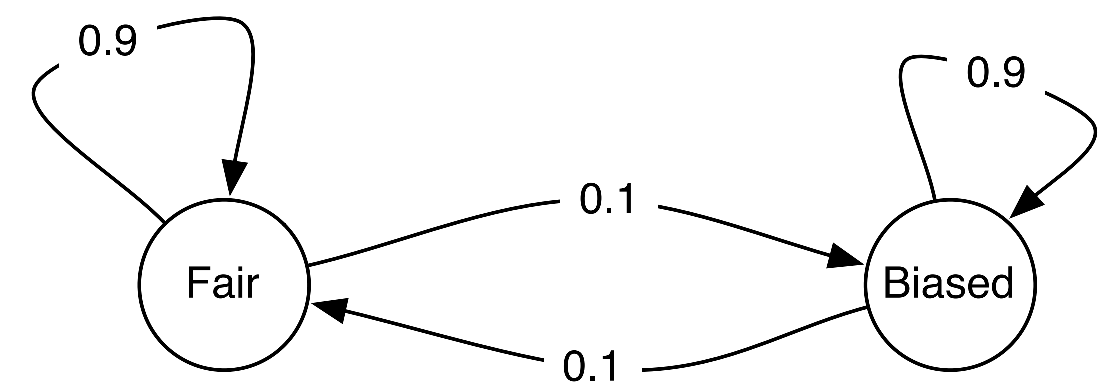


	H	T
Fair	0.5	0.5
Biased	0.75	0.25



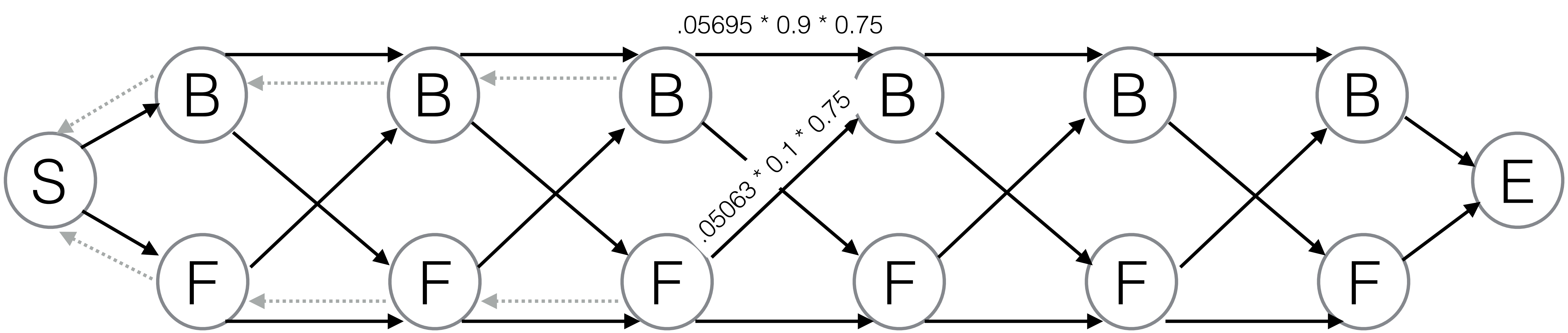
$\mathbf{x}$ = H T H H H T

Graph View of Viterbi



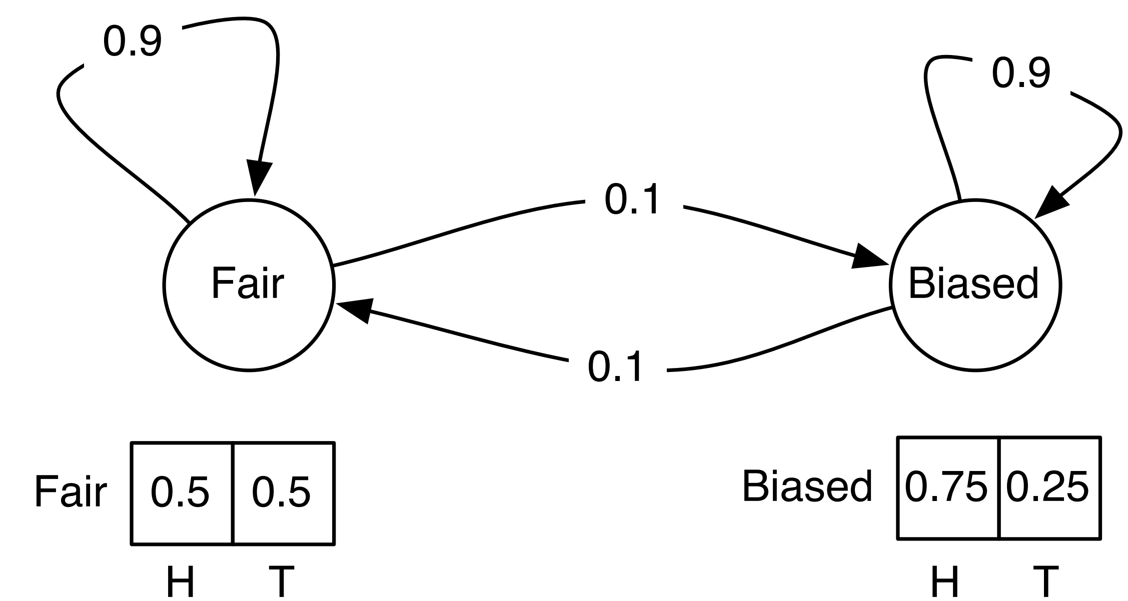
Fair	0.5	0.5
	H	T

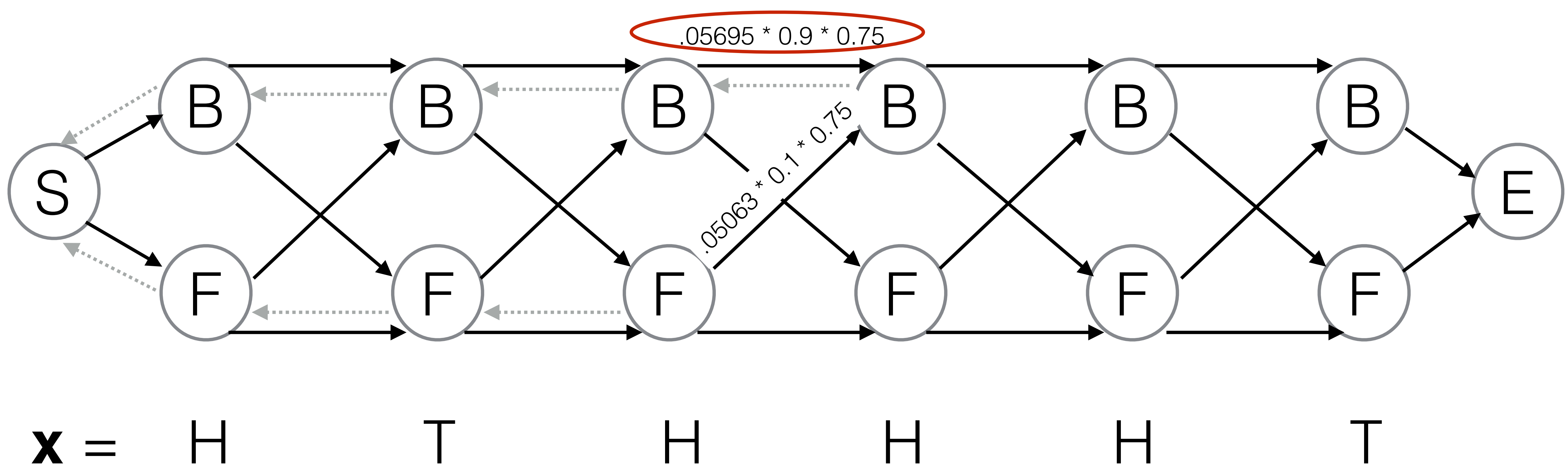
Biased	0.75	0.25
	H	T



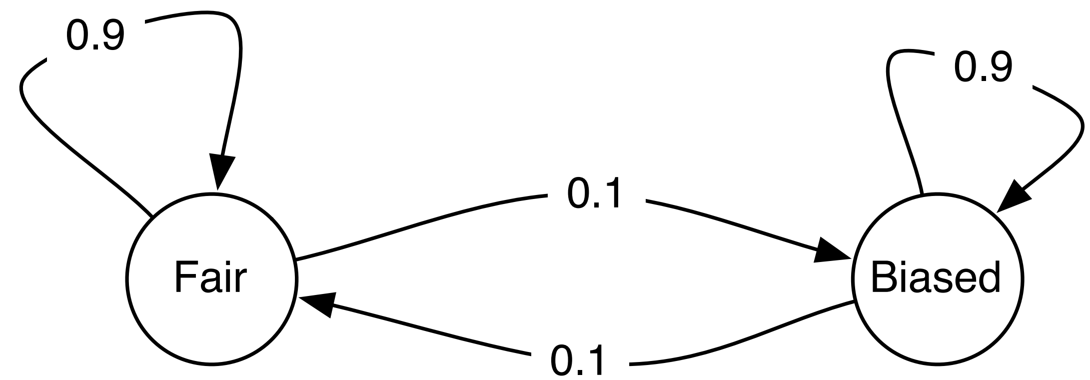
x = H T H H H T

Graph View of Viterbi



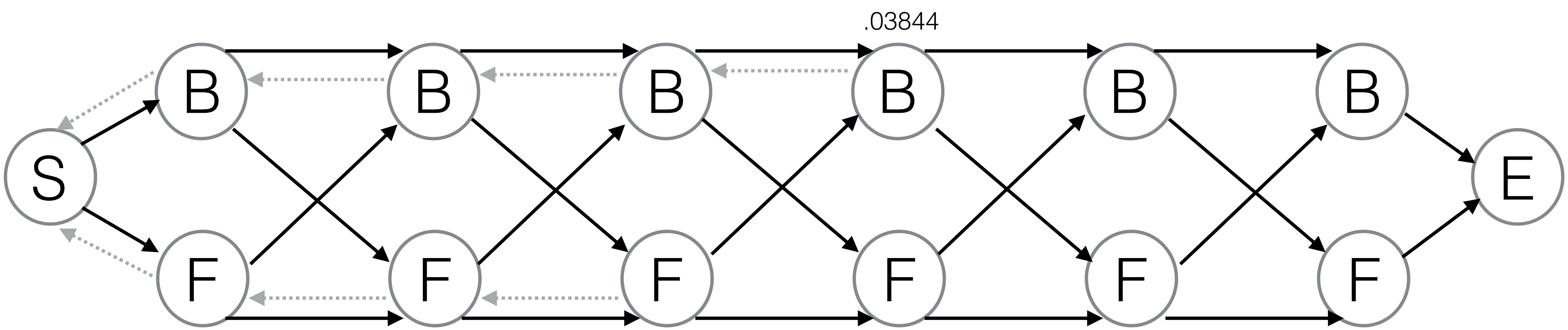


Graph View of Viterbi



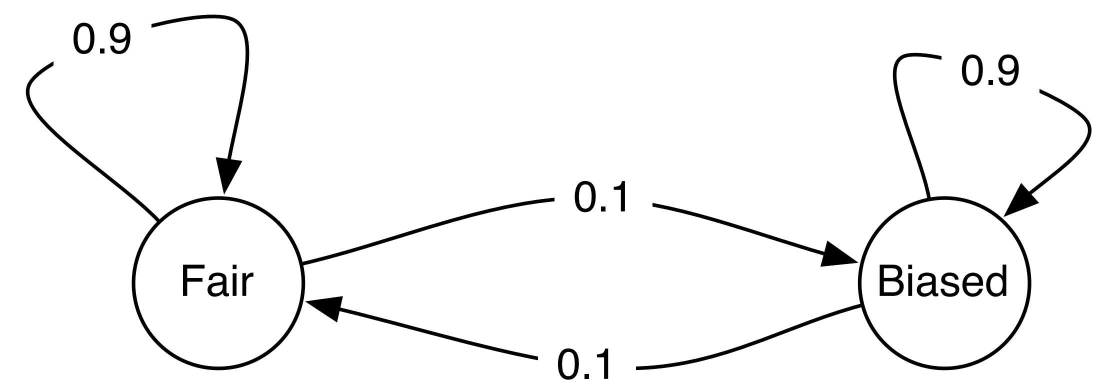
Fair	0.5	0.5
	H	T

Biased	0.75	0.25
	H	T

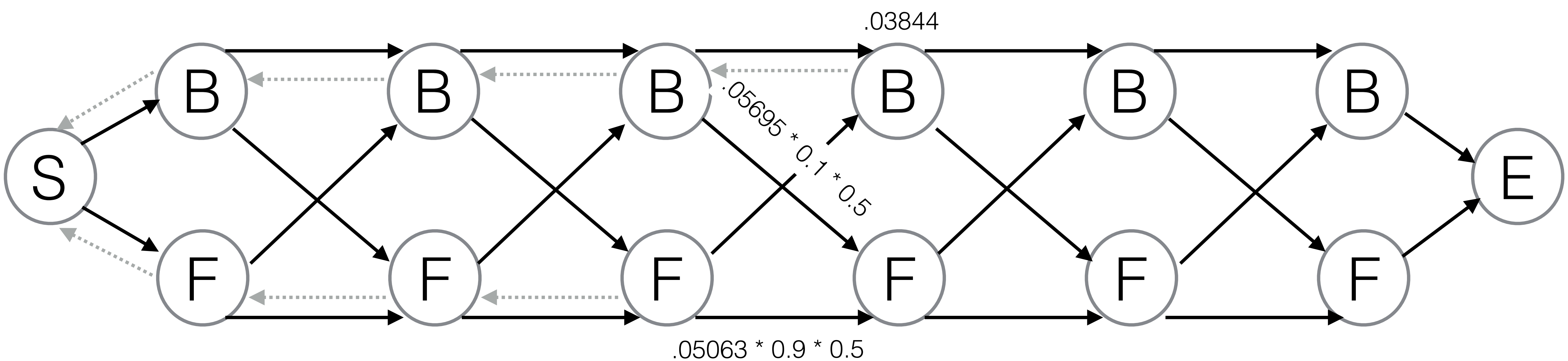


x = H T H H H T

Graph View of Viterbi

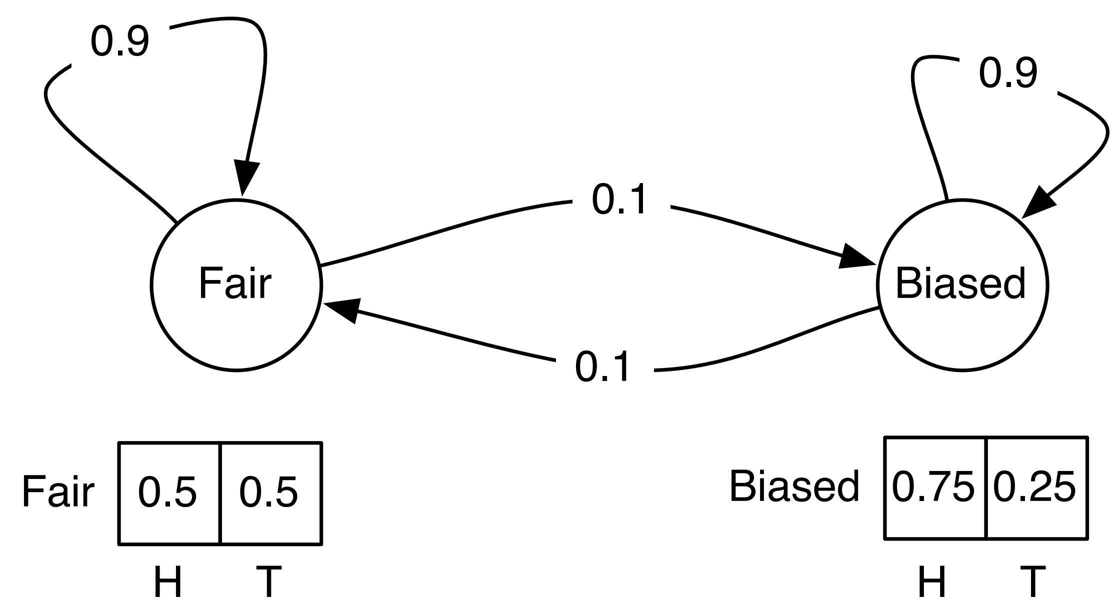


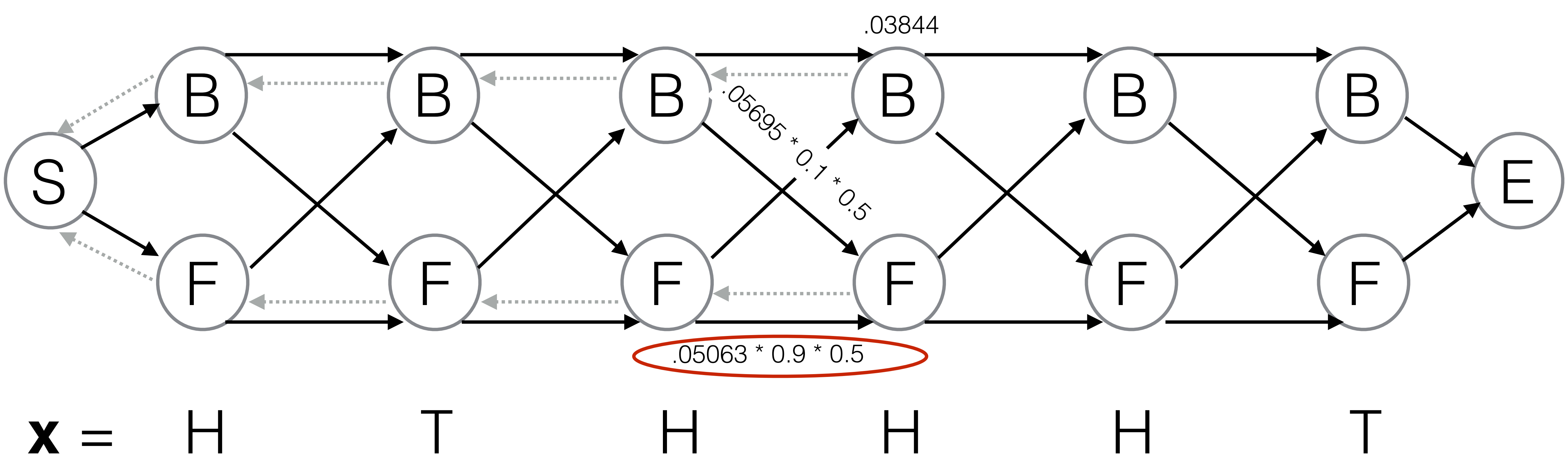
	H	T
Fair	0.5	0.5
Biased	0.75	0.25



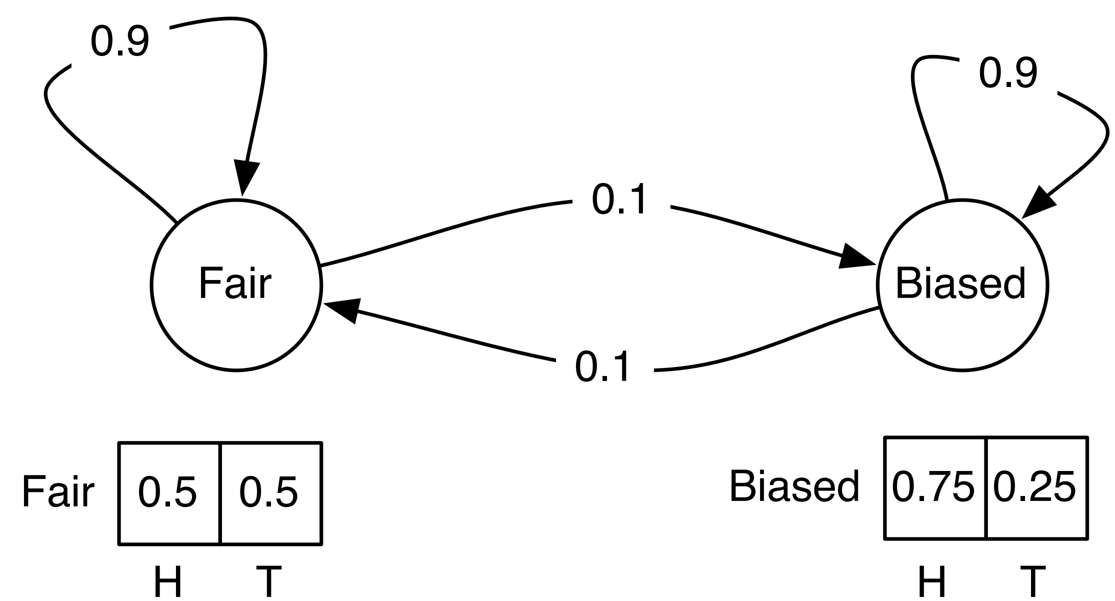
x = H T H H H T

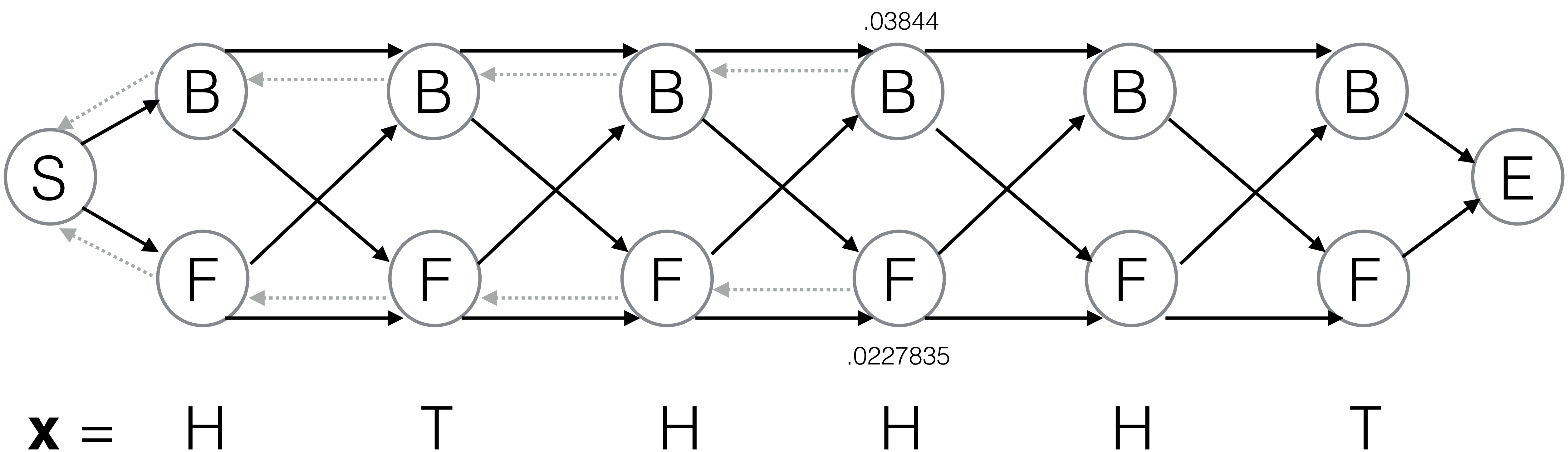
Graph View of Viterbi



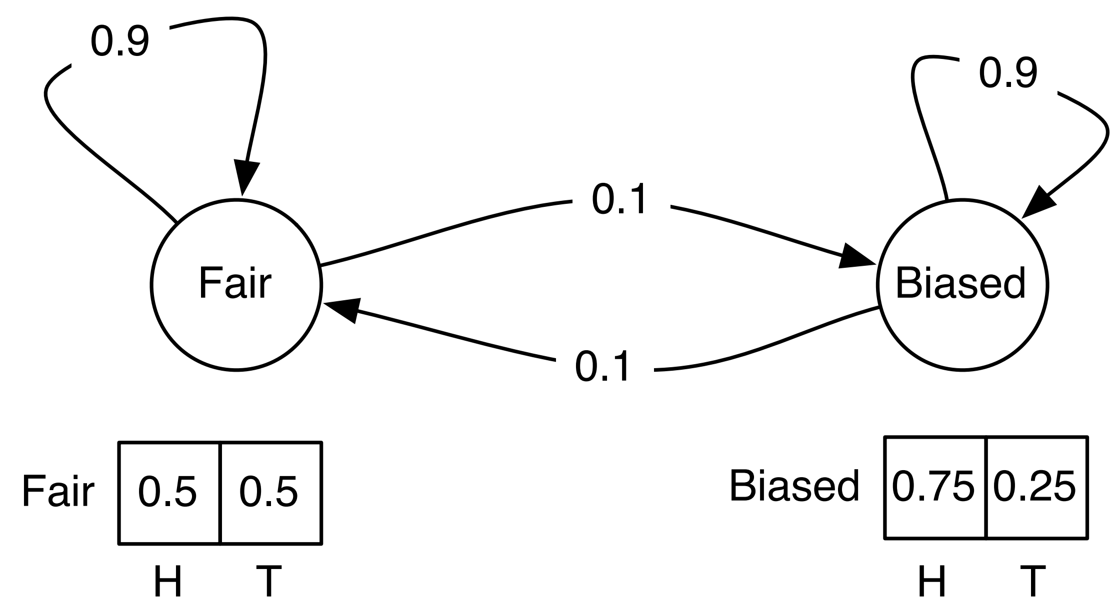


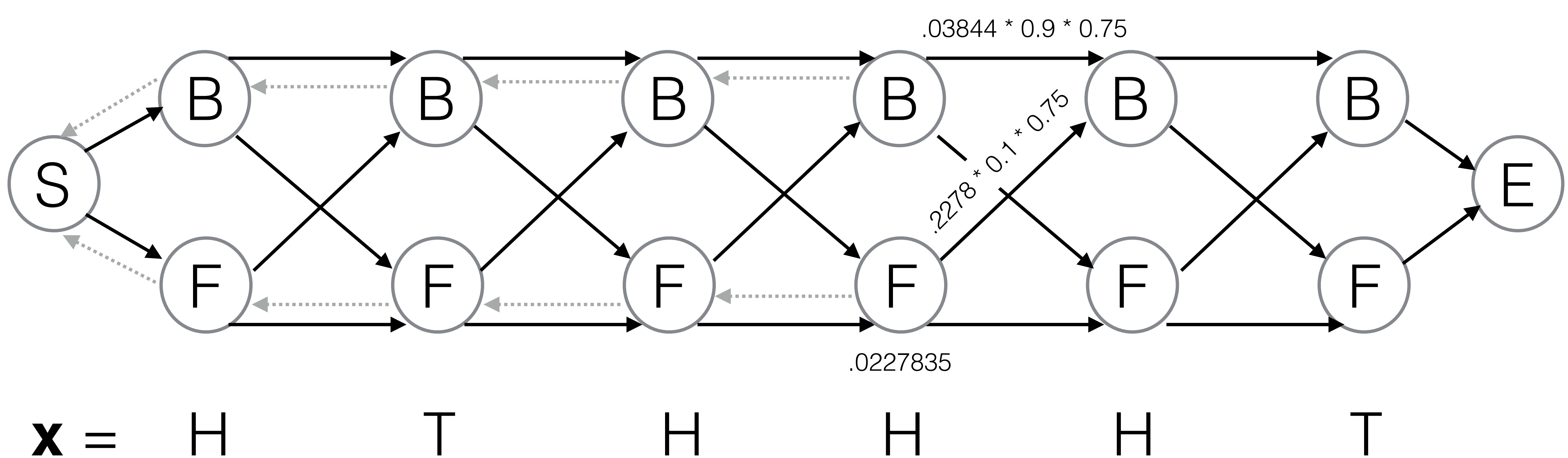
Graph View of Viterbi



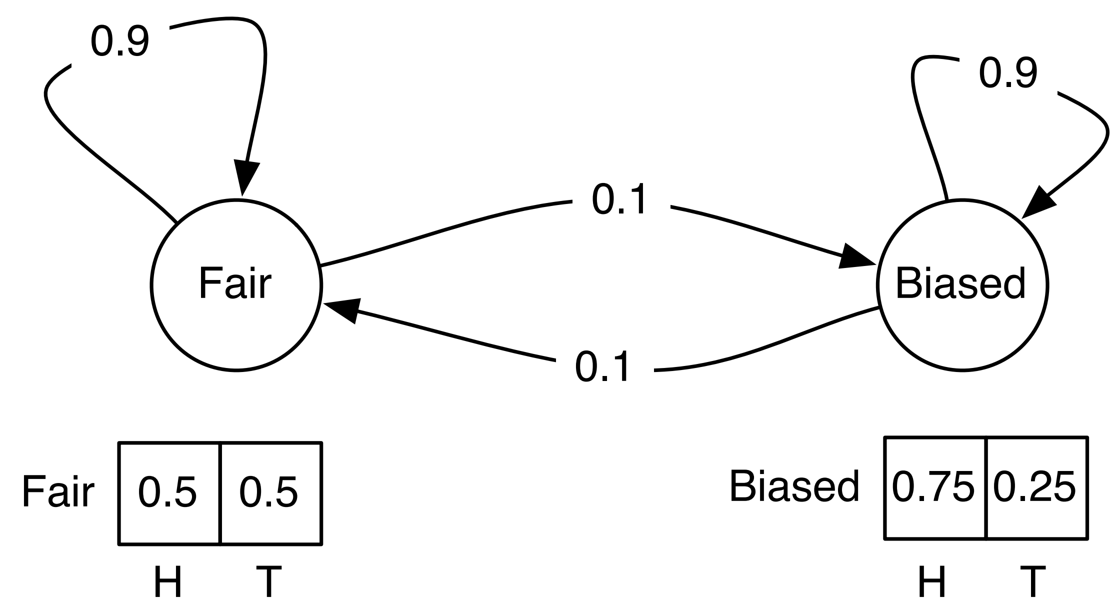


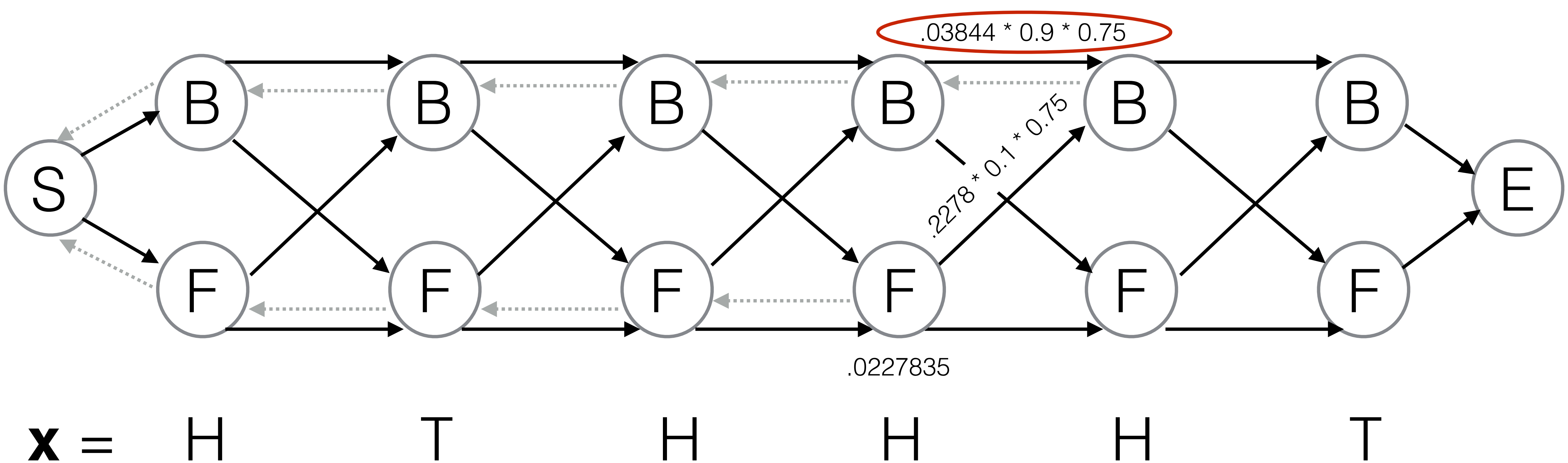
Graph View of Viterbi



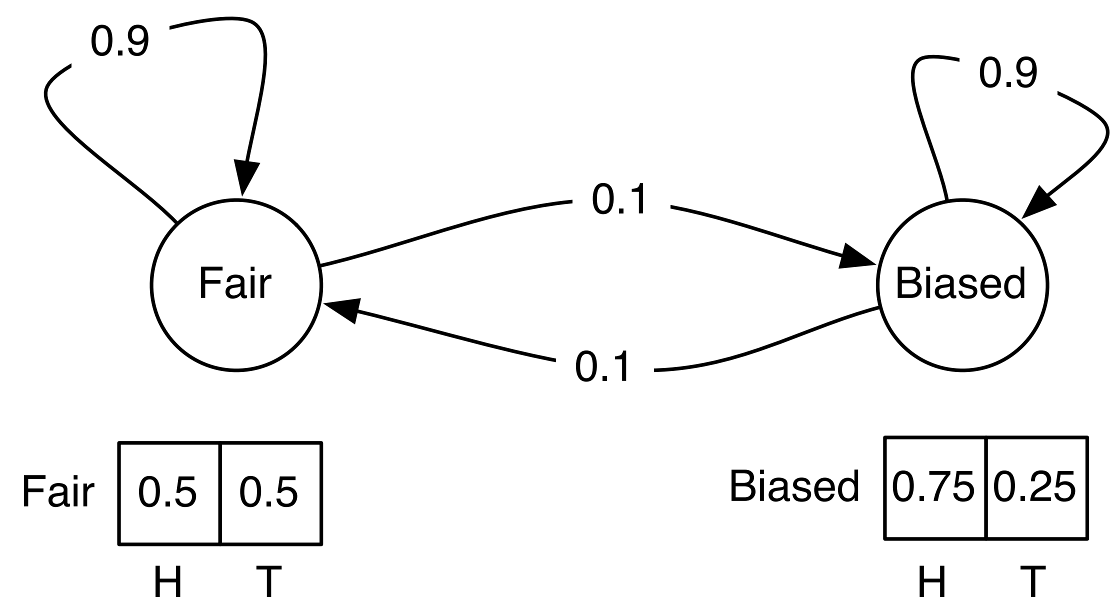


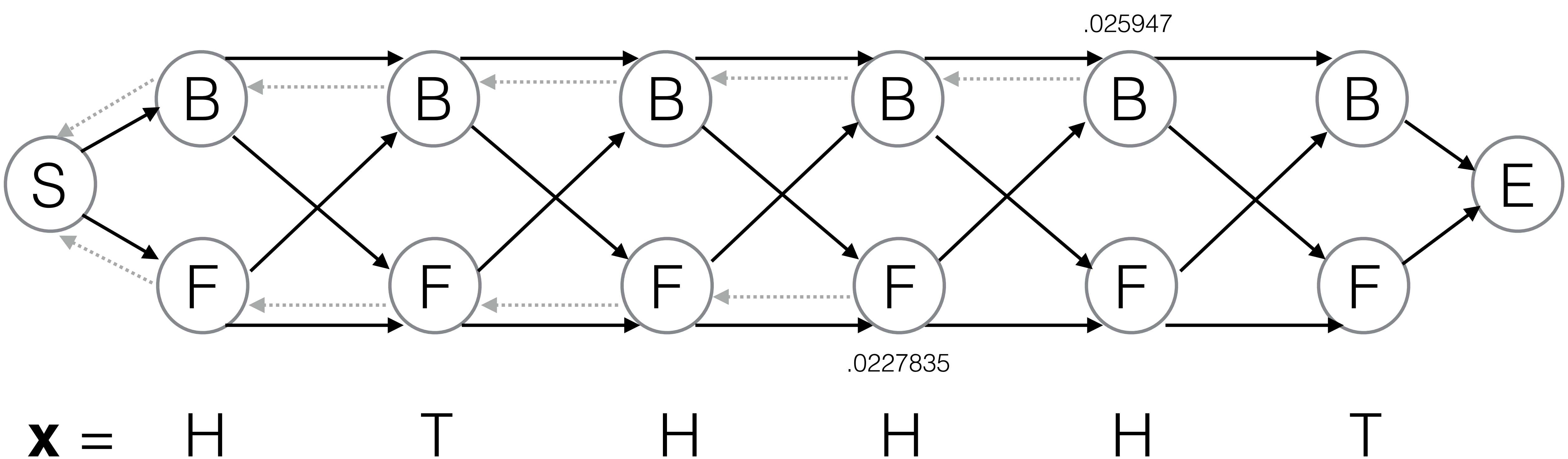
Graph View of Viterbi



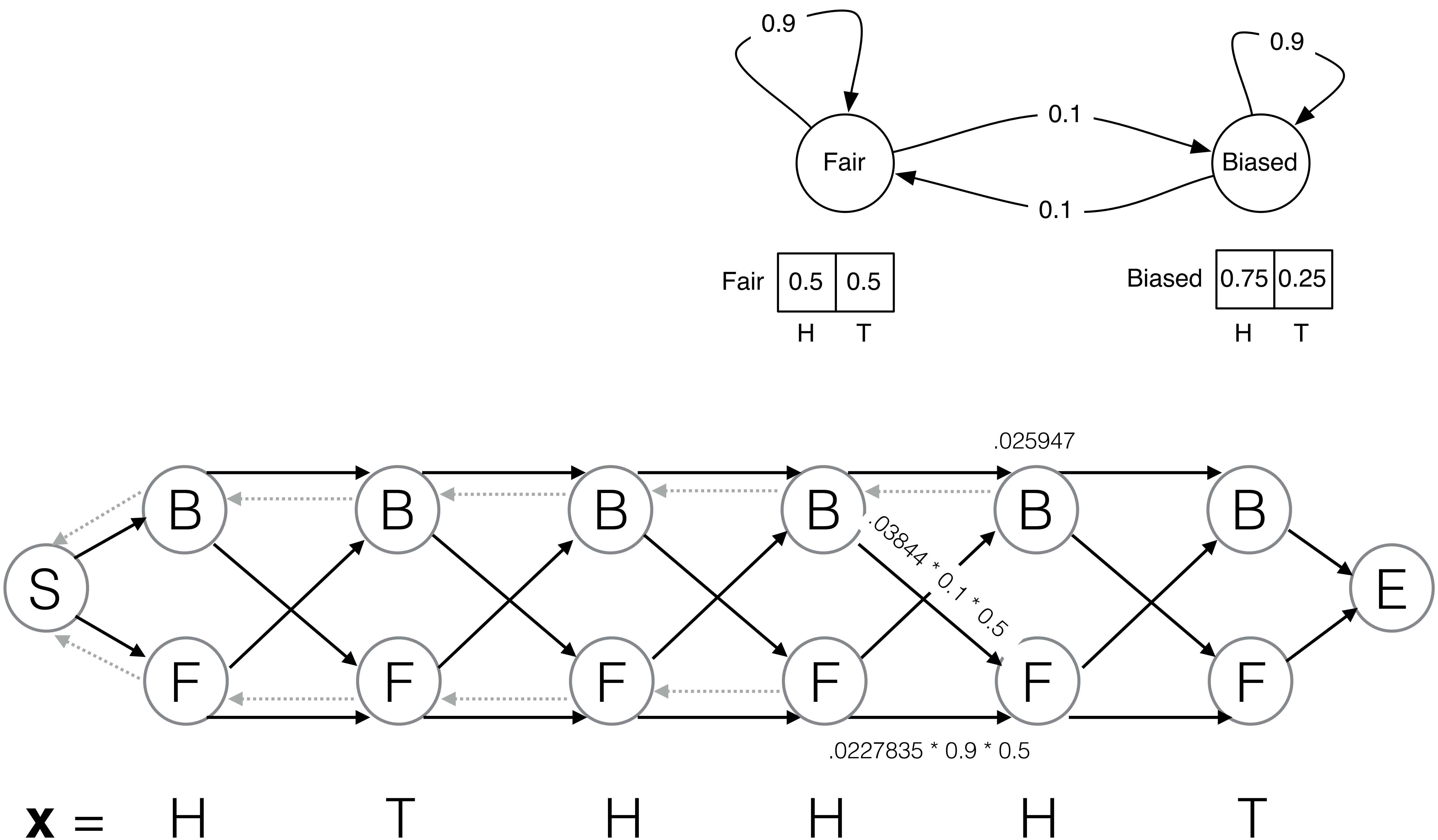


Graph View of Viterbi

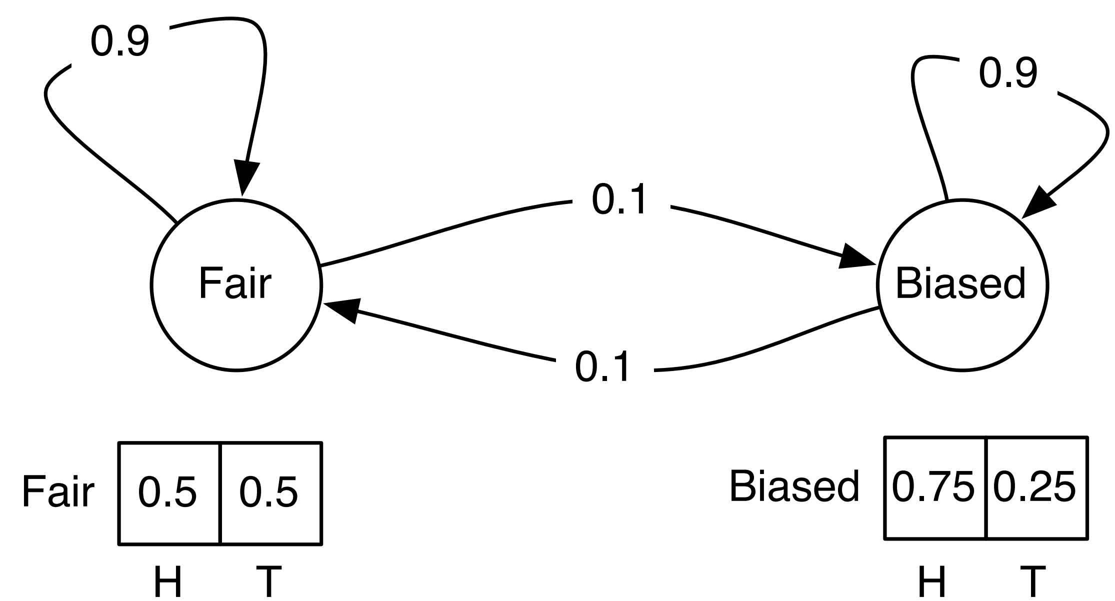


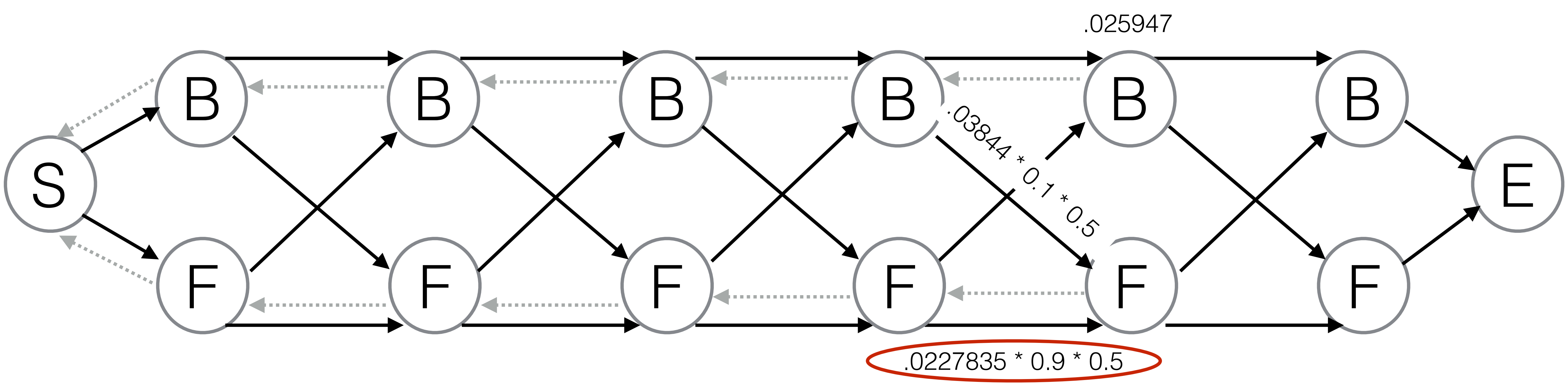


Graph View of Viterbi



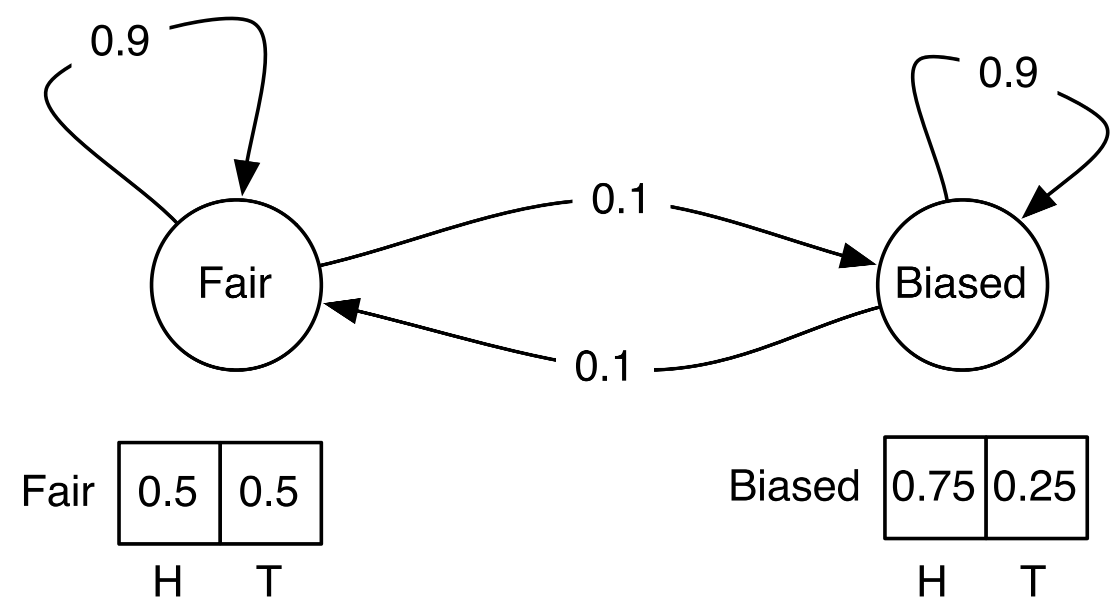
Graph View of Viterbi

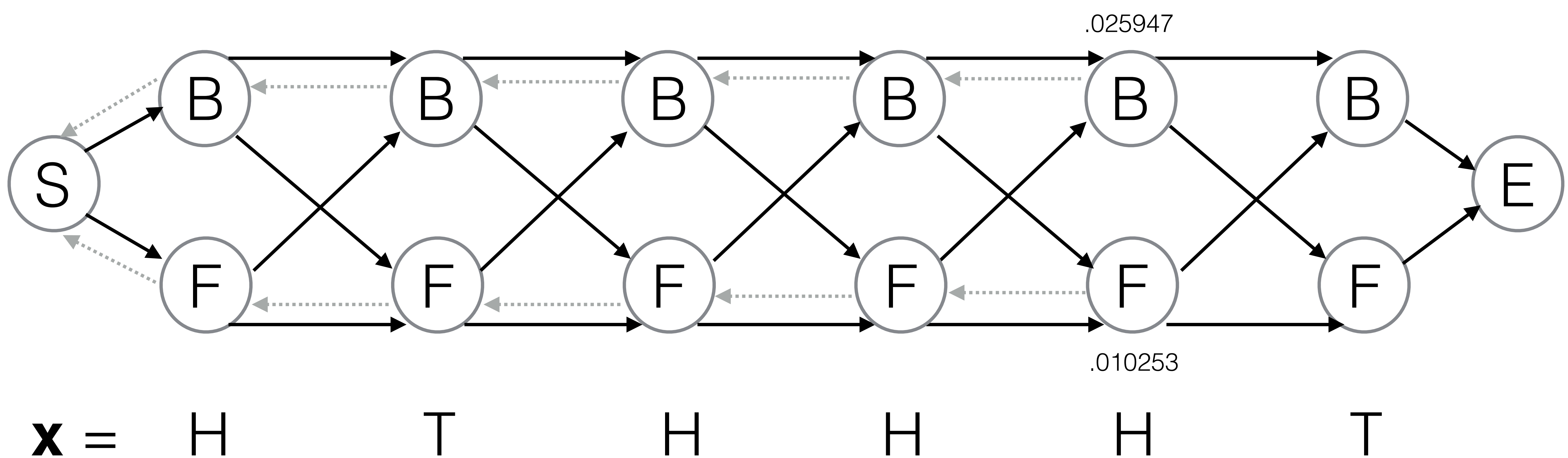




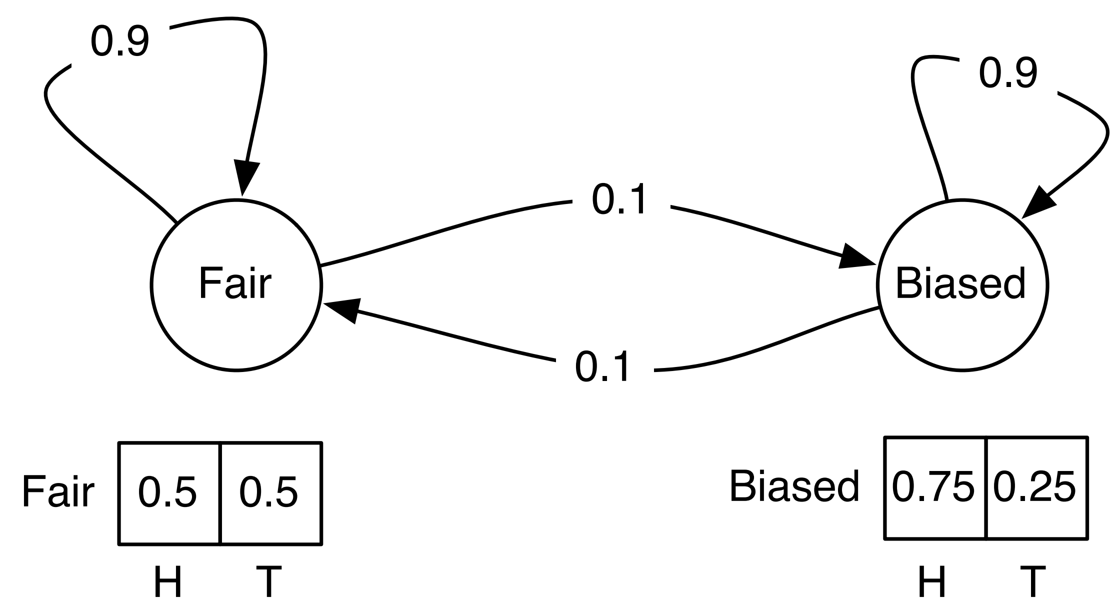
x = H T H H H T

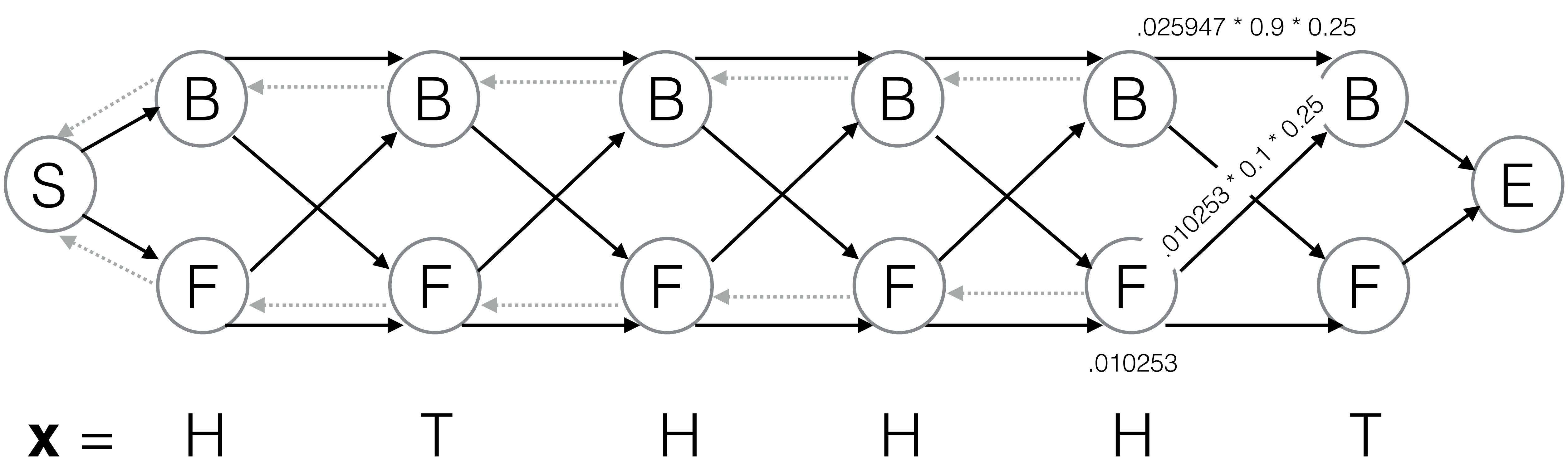
Graph View of Viterbi



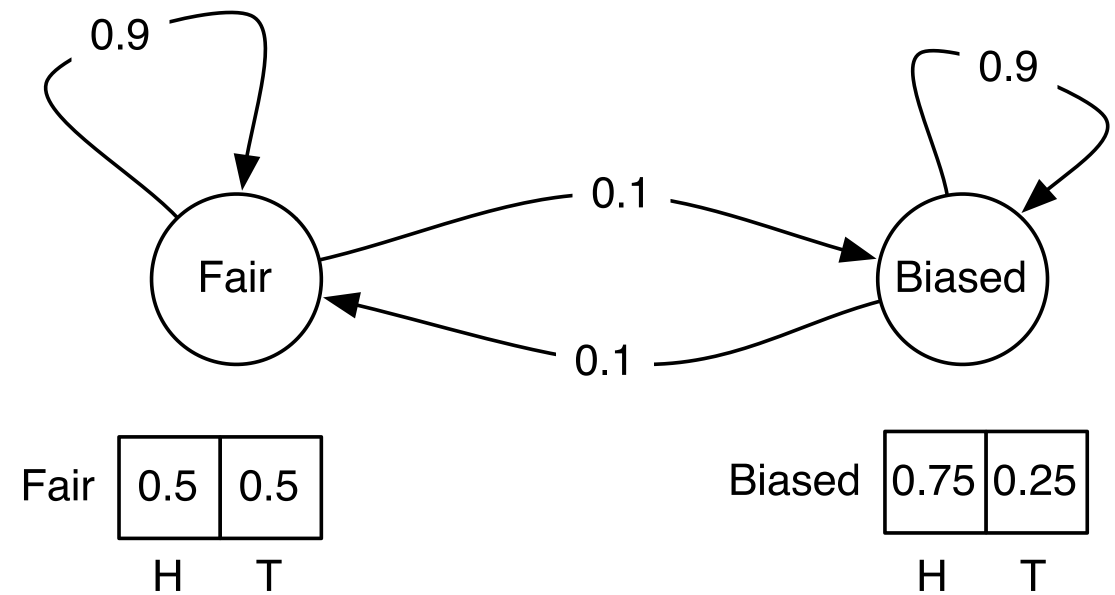


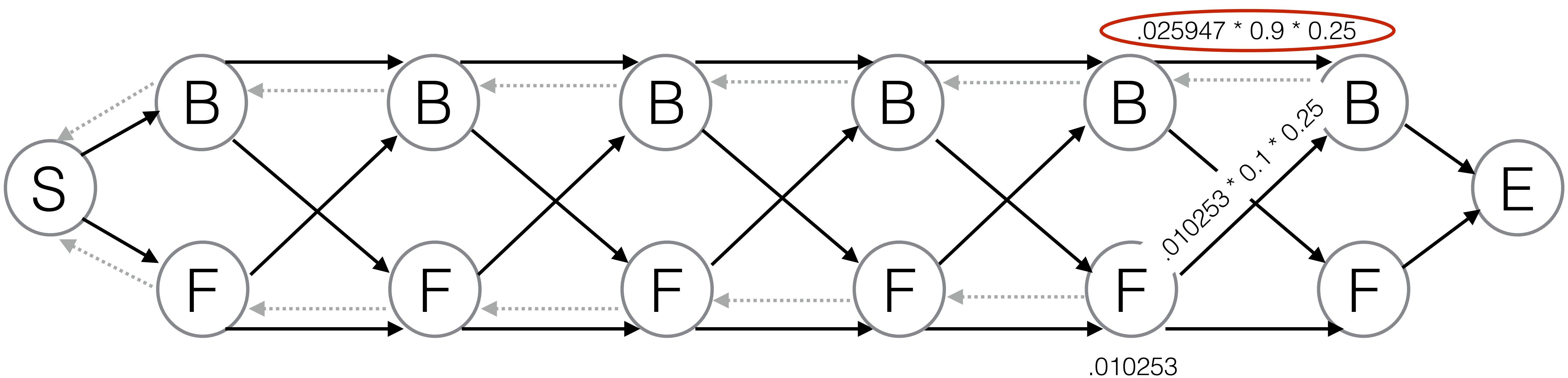
Graph View of Viterbi





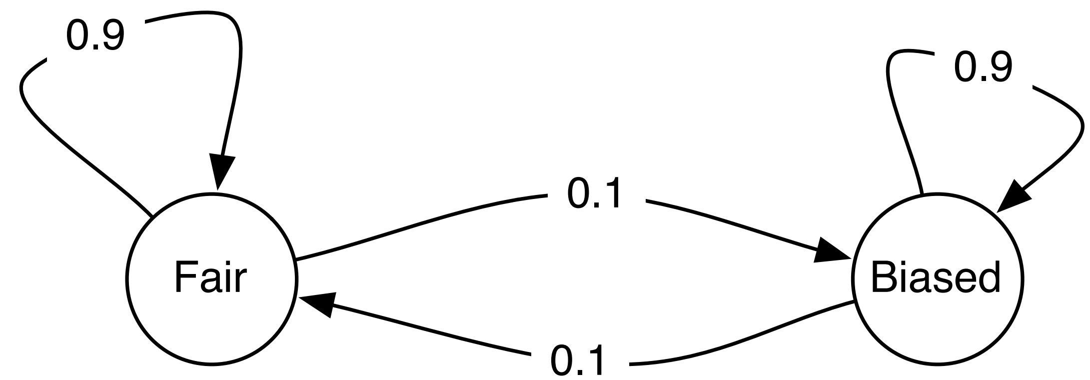
Graph View of Viterbi





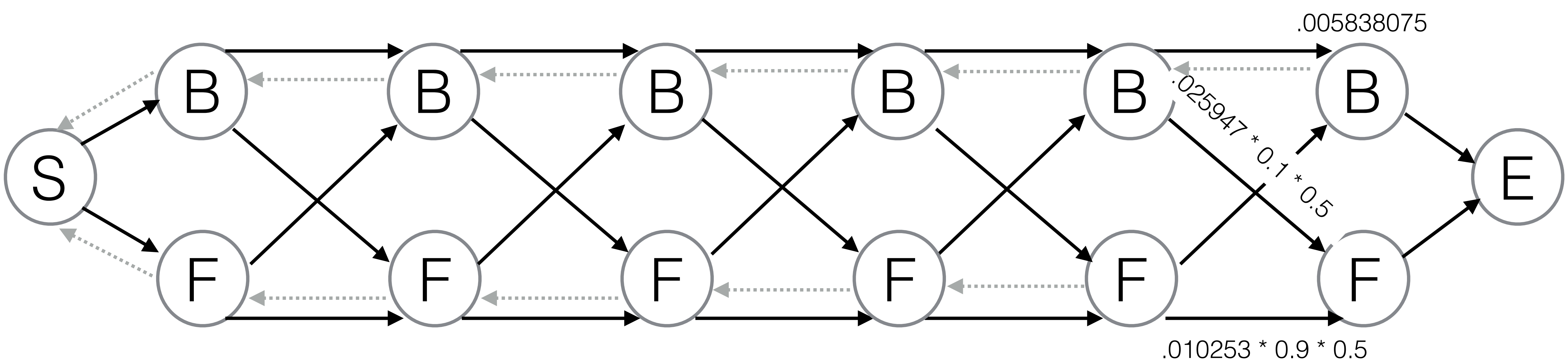
x = H T H H H T

Graph View of Viterbi



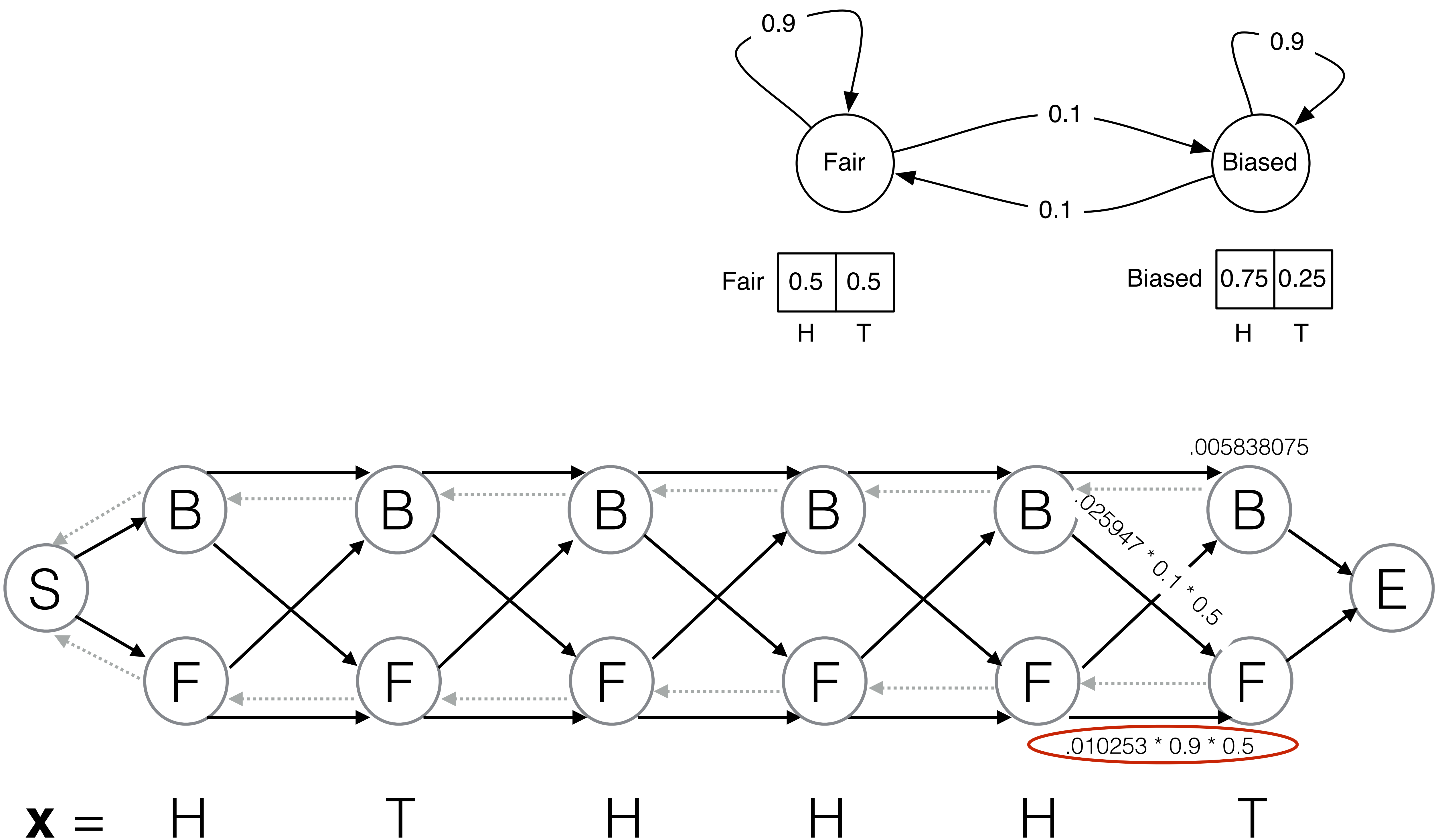
Fair	0.5	0.5
	H	T

Biased	0.75	0.25
	H	T

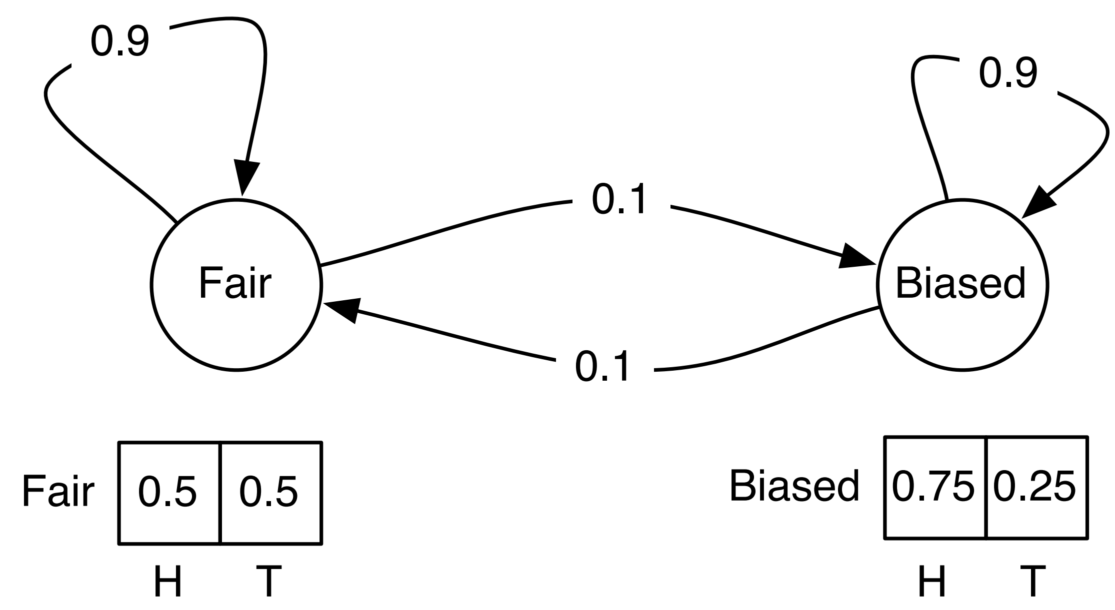


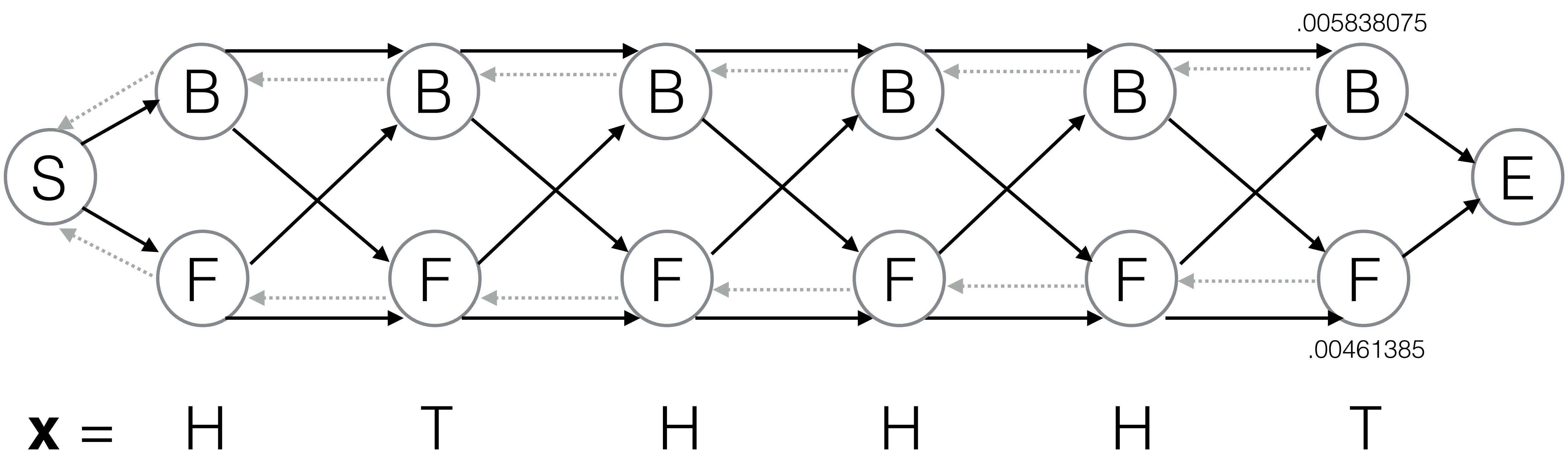
x = H T H H H T

Graph View of Viterbi

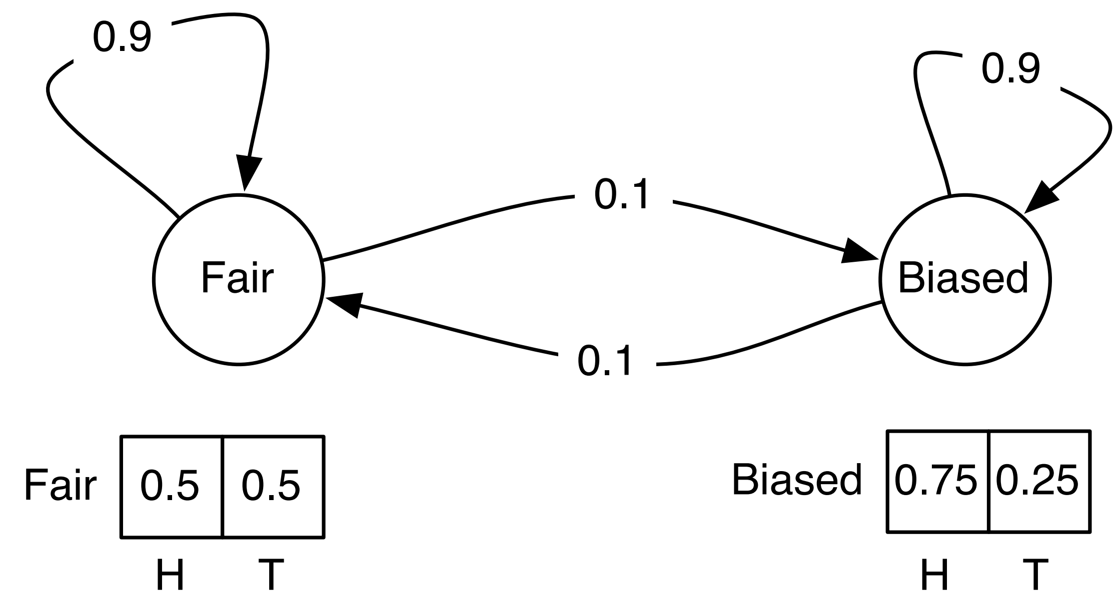


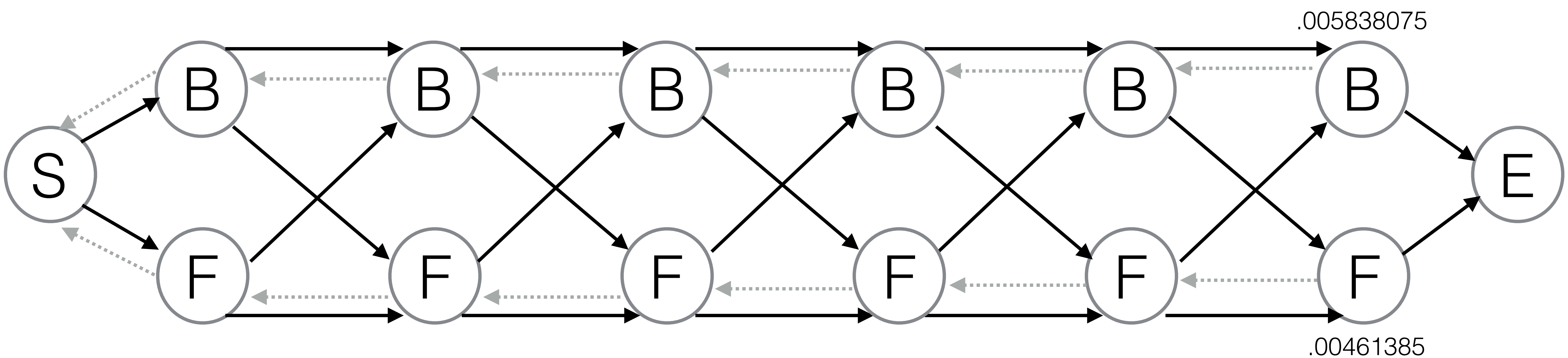
Graph View of Viterbi





Graph View of Viterbi

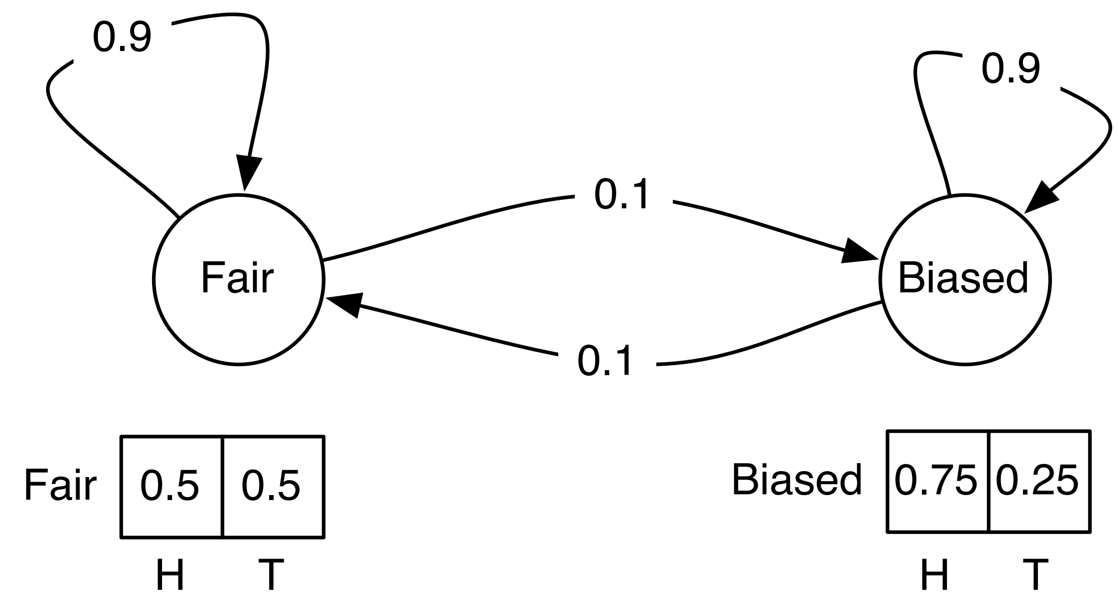


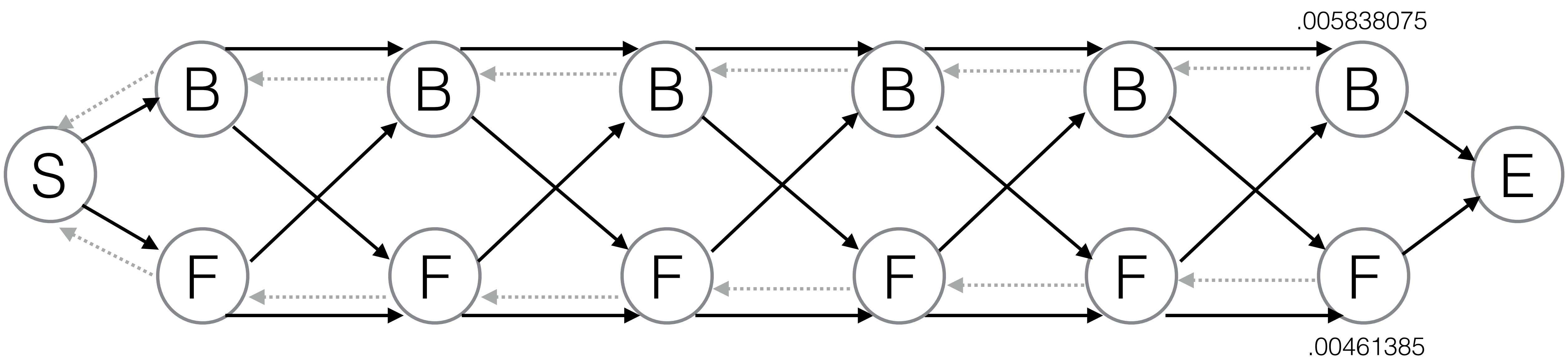


x = H T H H H T

In this case, transitions to the end state (emitting no symbol), won't matter

Graph View of Viterbi

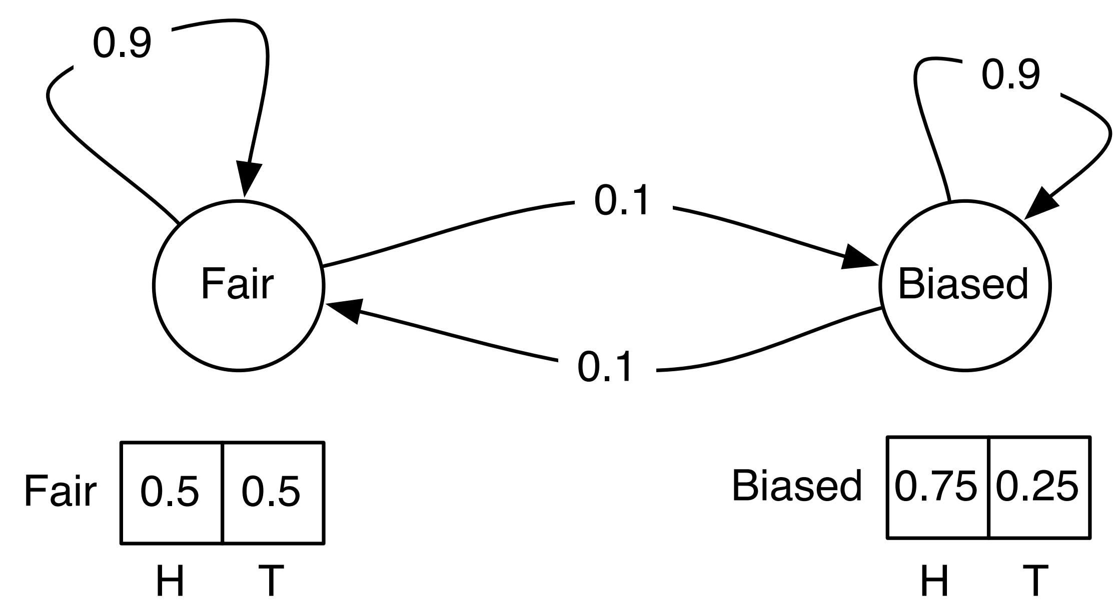


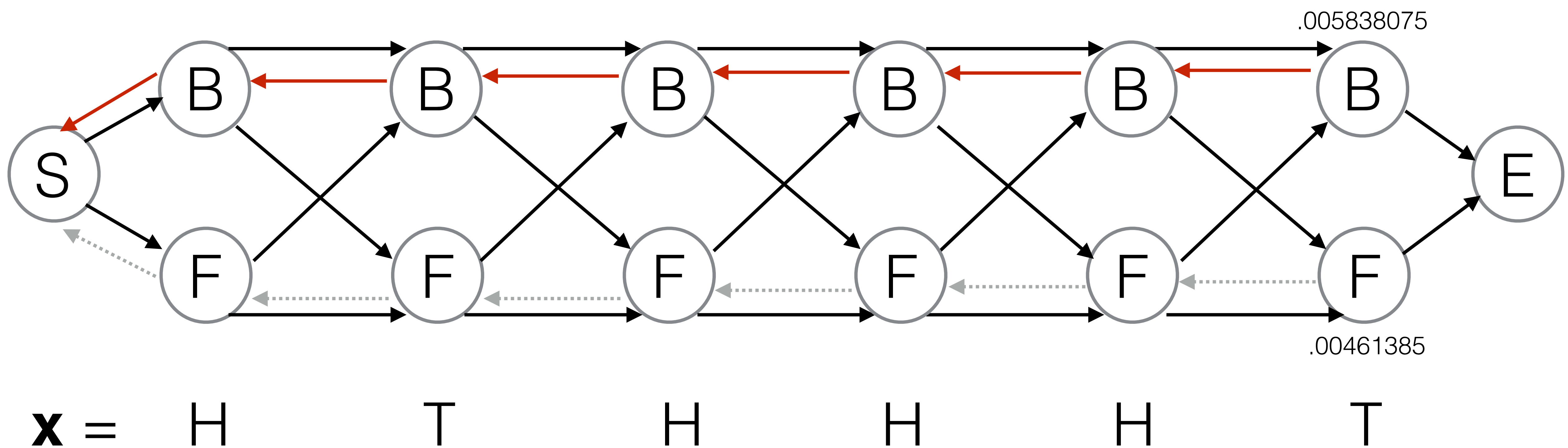


x = H T H H H T

We can “trace back” our path to determine the hidden states taken on when traversing the optimal path.

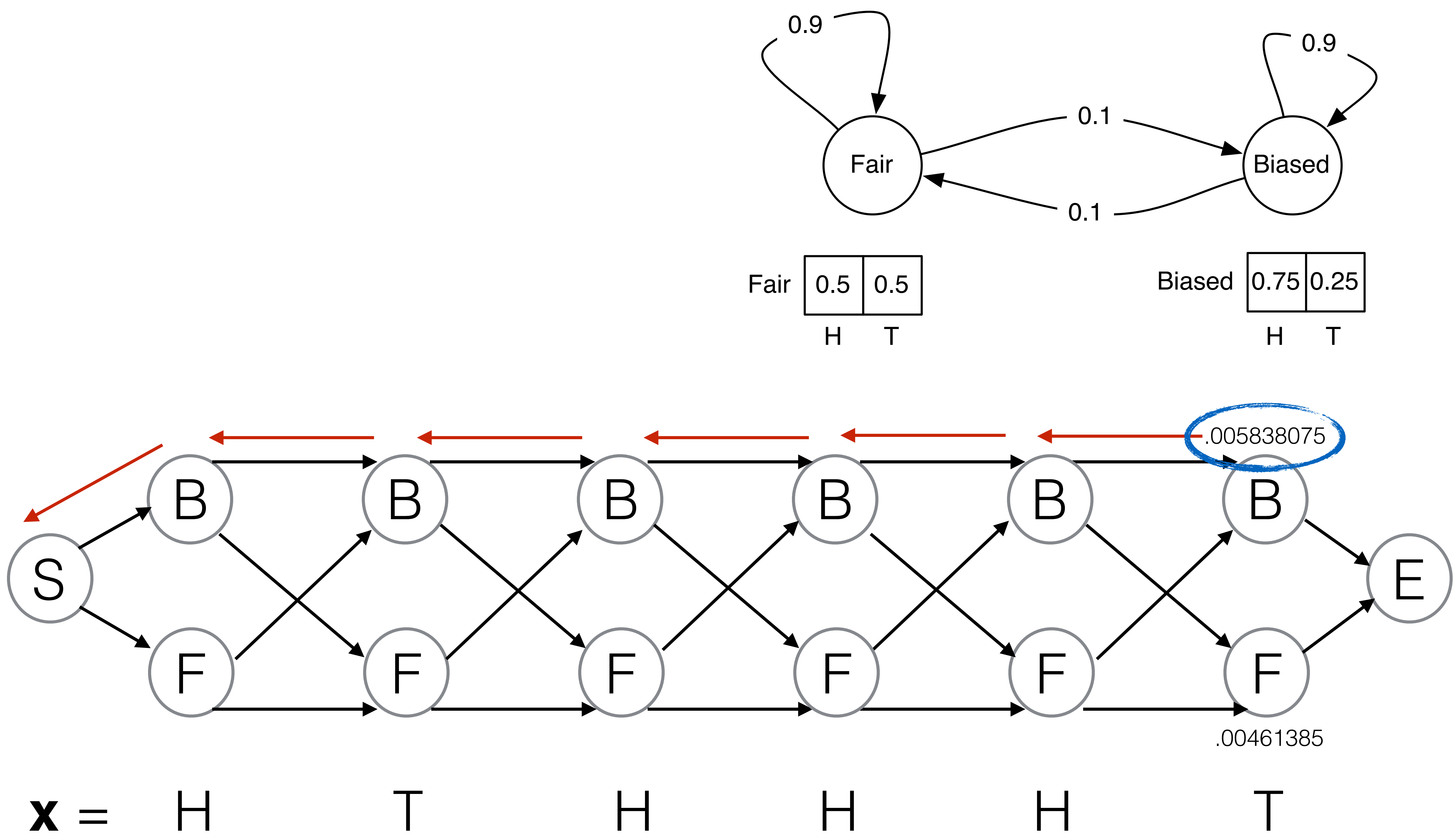
Graph View of Viterbi





In this case, the path was simple — implying a biased coin the entire time.

Graph View of Viterbi



Note that in practice, esp. with long sequences, the absolute prob. of the best path may be **very** small.

Running Time

- # of subproblems = $O(n|Q|)$, where n is the length of the sequence.
- Time to solve a subproblem = $O(|Q|)$
- Total running time: $O(n|Q|^2)$

*

Using Logs

Typically, we take the log of the probabilities to avoid multiplying a lot of terms:

$$\begin{aligned}\log(A[a,k]) &= \max_{b\in Q}\{\log(A[b,k-1]\times \Pr(b\to a)\times \Pr(x_k \mid \pi_k = a))\}\\ &= \max_{b\in Q}\{\log(A[b,k-1]) + \log(\Pr(b\to a)) + \log(\Pr(x_k \mid \pi_k = a))\}\end{aligned}$$

Remember: $\log(ab) = \log(a) + \log(b)$

Why do we want to avoid multiplying lots of terms?

Multiplying leads to very small numbers:

0.1 x 0.1 x 0.1 x 0.1 x 0.1 = 0.00001

This can lead to underflow.

Taking logs and adding keeps numbers bigger.

*

Estimating HMM Parameters

$$(x^{(1)},\pi^{(1)}) = \begin{matrix} x_1^{(1)} x_2^{(1)} x_3^{(1)} x_4^{(1)} x_5^{(1)} \ldots x_n^{(1)} \\ \pi_1^{(1)} \pi_2^{(1)} \pi_3^{(1)} \pi_4^{(1)} \pi_5^{(1)} \ldots \pi_n^{(1)} \end{matrix}$$

$$(x^{(2)},\pi^{(2)}) = \begin{matrix} x_1^{(2)} x_2^{(2)} x_3^{(2)} x_4^{(2)} x_5^{(2)} \ldots x_n^{(2)} \\ \pi_1^{(2)} \pi_2^{(2)} \pi_3^{(2)} \pi_4^{(2)} \pi_5^{(2)} \ldots \pi_n^{(2)} \end{matrix}$$

Training examples where outputs and paths are known.

# of times transition *a* → *b* is observed.

# of times *x* was observed to be output from state *a*.

$$\Pr(a \to b) = \frac{A_{ab}}{\sum_{q \in Q} A_{aq}}$$

$$\Pr(x \mid a) = \frac{E_{xa}}{\sum_{x' \in \Sigma} E_{x'a}}$$

*

Pseudocounts

\# of times transition $a \to b$ is observed.

\# of times x was observed to be output from state a.

$$\Pr(a \to b) = \frac{A_{ab}}{\sum_{q \in Q} A_{aq}} \qquad \Pr(x \mid a) = \frac{E_{xa}}{\sum_{x \in \Sigma} E_{xq}}$$

What if a transition or emission is never observed in the training data?
$\Rightarrow$ 0 probability

Meaning that if we observe an example with that transition or emission in the real world, we will give it 0 probability.

But it's unlikely that our training set will be large enough to observe *every* possible transition.

Hence: we take A_{ab} = (#times a $\to$ b was observed) + 1 $\leftarrow$ "pseudocount"
Similarly for E_{xa}.

*

Viterbi Training

- **Problem**: typically, in the real would we only have examples of the output x, and we don't know the paths π.

Viterbi Training Algorithm:

1. Choose a random set of parameters.
2. Repeat:
   1. Find the best paths.
   2. Use those paths to estimate new parameters.

This is a local search algorithm.

It's also an example of a "Gibbs sampling" style algorithm.

The Baum-Welch algorithm is similar, but doesn't commit to a single best path for each example. (basically EM for HMM training)

*

Some probabilities in which we are interested

What is the probability of observing a string *x* under the assumed HMM?

$$\Pr(x) = \sum_{\pi} \Pr(x, \pi)$$

What is the probability of observing *x* using a path where the ith state is *a*?

$$\Pr(x, \pi_i = a) = \sum_{\pi : \pi_i = a} \Pr(x, \pi)$$

What is the probability that the ith state is *a*?

$$\Pr(\pi_i = a | x) = \frac{\Pr(x, \pi_i = a)}{\Pr(x)}$$

*

How do we compute this:

$$\Pr(x, \pi_k = a) = \Pr(x_1, \ldots, x_i, \pi_i = a) \Pr(x_{i+1}, \ldots, x_n \mid \pi_i = a)$$

Recall the recurrence to compute **best** path for $x_1 \ldots x_k$ that ends at state a:

$$A[a, k] = \max_{b \in Q} \{A[b, k-1] \times \Pr(b \to a) \times \Pr(x_k \mid \pi_k = a)\}$$

We can compute the probability of emitting $x_1, \ldots, x_k$ using **any** path that ends in *a*:

$$F[a, k] = \sum_{b \in Q} F[b, k-1] \times \Pr(b \to a) \times \Pr(x_k \mid \pi_k = a)$$

*

The Forward Algorithm

We can compute the probability of emitting $x_1,\ldots,x_k$ using **any** path that ends in *a*:

$$F[a,k] = \sum_{b \in Q} F[b,k-1] \times \Pr(b \to a) \times \Pr(x_k \mid \pi_k = a)$$

The forward algorithm also allows us to solve the "Evaluation Problem".

Evaluation Problem:

Given an HMM λ = (∑, Q, A, E) and an observation **x**

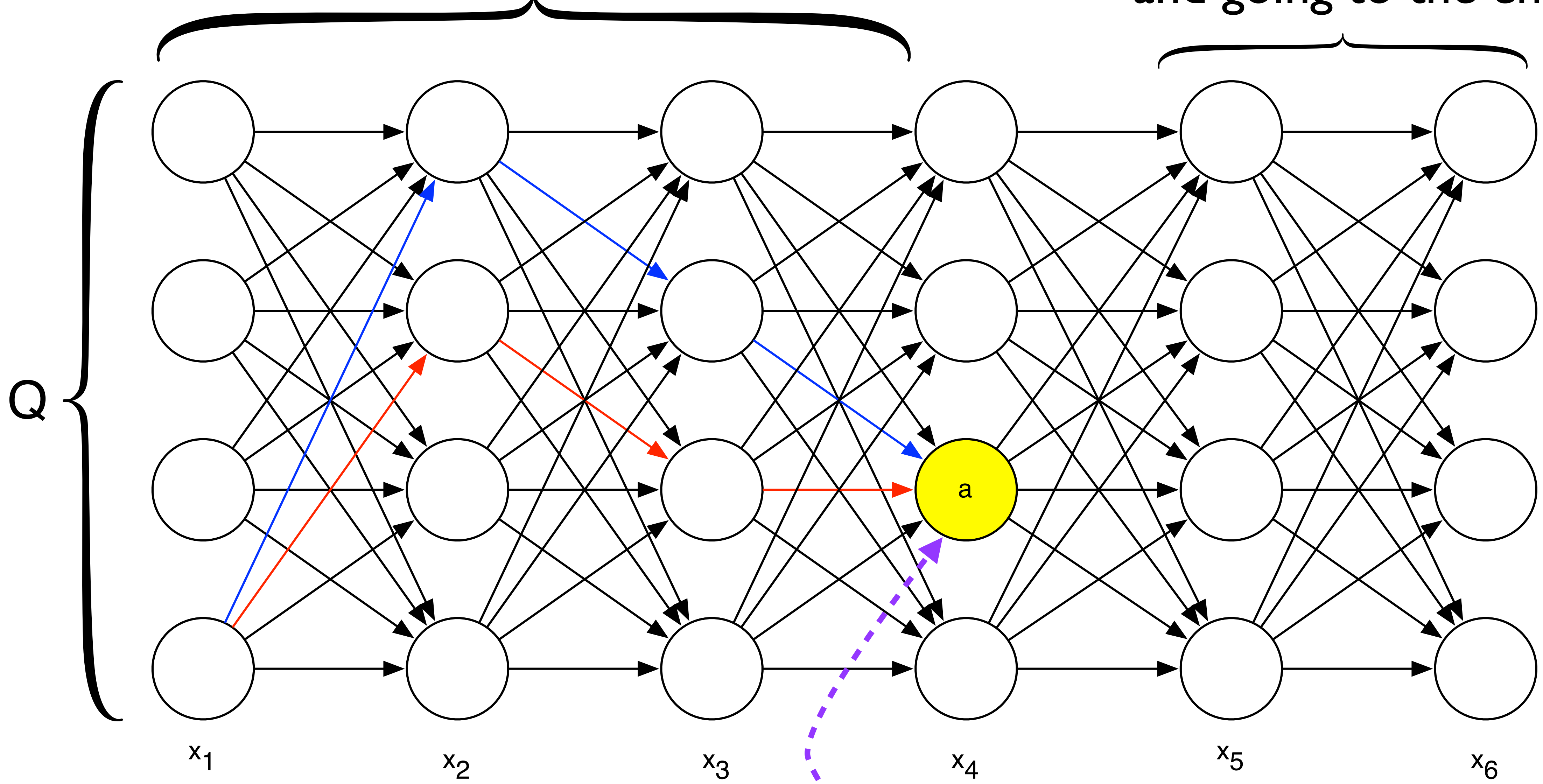
Find Pr(**x** | λ) — the prob. of the observations under the model

*

The Forward Algorithm

Computes the total probability of all the paths of length *k* ending in state *a*.

Still need to compute the probability of paths leaving *a* and going to the end.



F[a,4]

*

The Backward Algorithm

The same idea as the forward algorithm, we just start from the *end* of the input string and work towards the beginning:

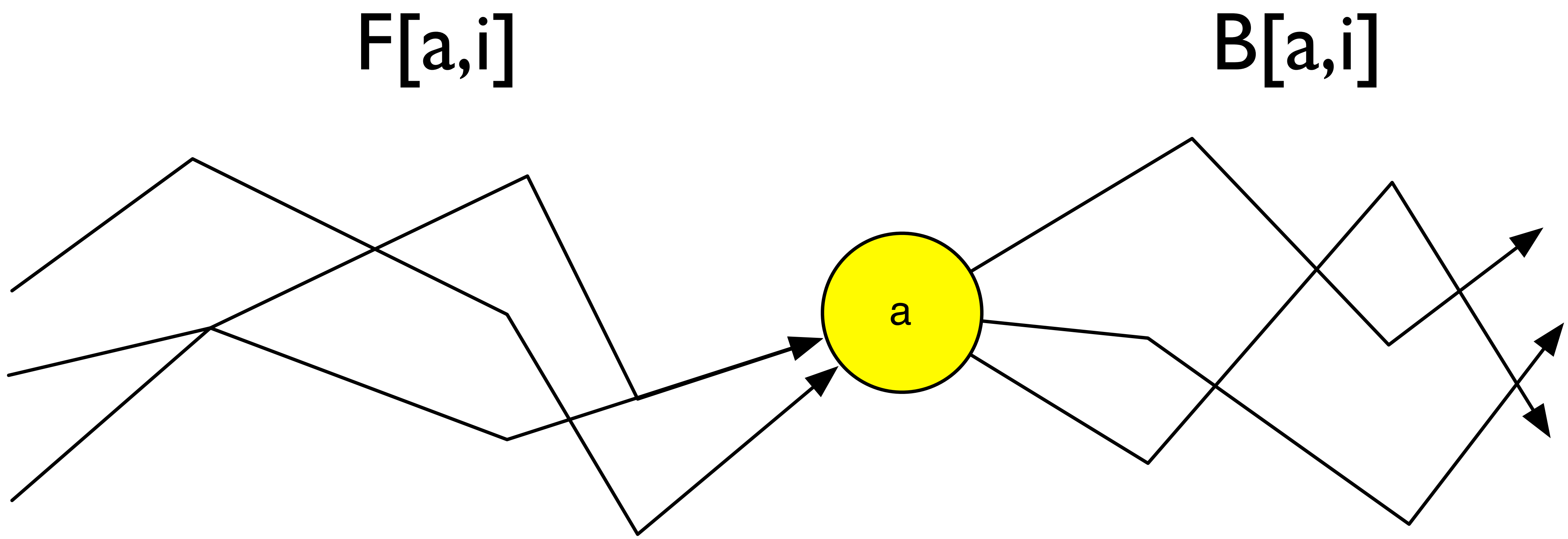
B[a,k] = "the probability of generating string $x_{k+1},\ldots,x_n$ starting from state b"

$$B[a,k] = \sum_{b \in Q} \underbrace{B[b,k+1]}_{\substack{\text{Prob for} \\ x_{k+1}..x_n \\ \text{starting in} \\ \text{state } b}} \times \underbrace{\Pr(a \to b)}_{\substack{\text{Probability} \\ \text{going from} \\ \text{state } a \text{ to } b}} \times \underbrace{\Pr(x_{k+1} \mid \pi_{k+1} = b)}_{\substack{\text{Probability of emitting} \\ x_{k+1} \text{ given that the next} \\ \text{state is b.}}}$$

*

The Forward-Backward Algorithm

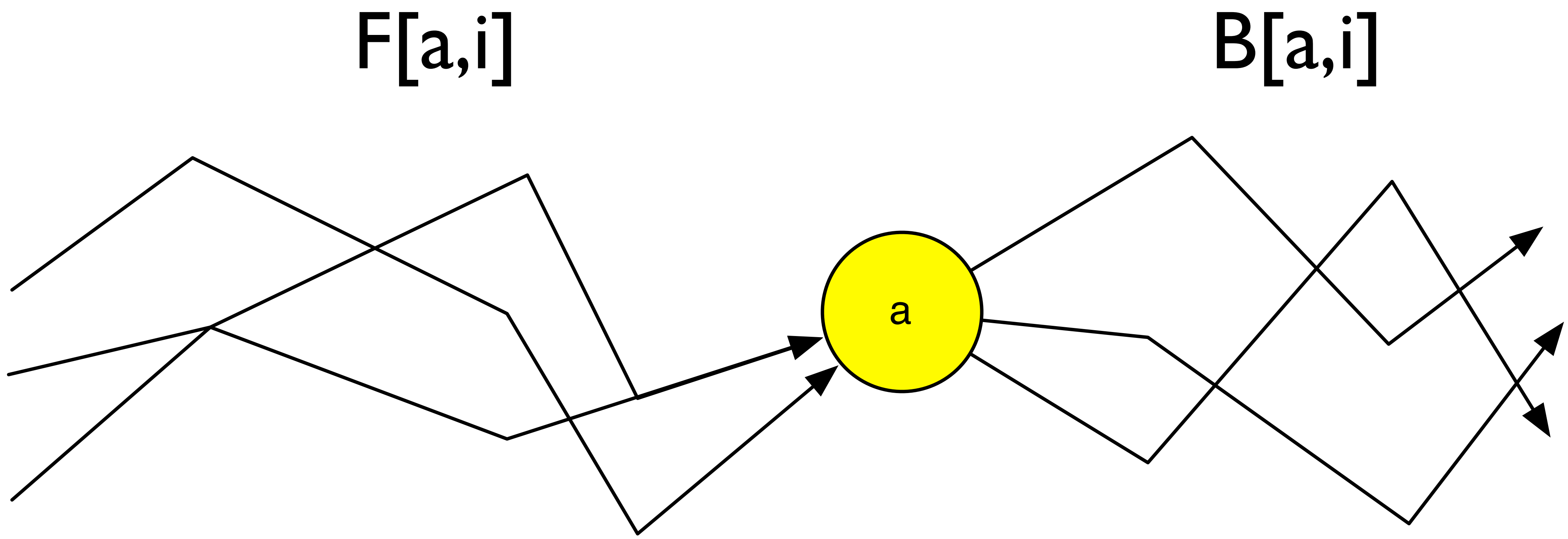
$$\Pr(\pi_i = a \mid x) = \frac{\Pr(x, \pi_i = k)}{\Pr(x)} = \frac{F[a,i] \cdot B[a,i]}{\Pr(x)}$$



*

The Forward-Backward Algorithm

$$\Pr(\pi_i = a \mid x) = \frac{\Pr(x, \pi_i = k)}{\Pr(x)} = \frac{F[a,i] \cdot B[a,i]}{\Pr(x)}$$



This works because F[a,i] is independent of B[a,i], given that we are in state **a** at time **i** (the Markovian assumption).

*

Alternative Training (Baum-Welch)

$\theta = (A, E, \pi)$ Initialize transition, emission and initial state distribution "randomly"

Y Training data such that Y_t represents the vector of observations at step t

While not converged:

Run the forward algorithm

Run the backward algorithm

Compute γ, the probability of being in each hidden state at each time:

$$\gamma_i(t) = \Pr\left(X_t = i \mid Y, \theta\right) = \frac{F[i,t] \cdot B[i,t]}{\sum_{j=1}^{|Q|} F[j,t] \cdot B[j,t]}$$

Compute ξ, the prob of being in state i at step t, j at t+1 and producing the observed output at t+1

$$\xi_{ij}(t) = \Pr\left(X_t = i, X_{t+1} = j \mid Y, \theta\right) = \frac{F[i,t] \cdot A[i,j] \cdot B[j,t+1] \cdot E[j, y_{t+1}]}{\sum_{i=1}^{|Q|} \sum_{j=1}^{|Q|} F[i,t] \cdot A[i,j] \cdot B[j,t+1] \cdot E[j, y_{t+1}]}$$

update parameters

$$\pi_i^* = \gamma_i(1)$$

$$A^*[i,j] = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

Use these updated parameter estimates in the next iteration of the algo.

$$E^*[i, v_k] = \frac{\sum_{t=1}^{T} 1_{y_t = v_k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

notation inspired by: https://en.wikipedia.org/wiki/Baum%E2%80%93Welch_algorithm

Baum-Welch

Tries to find the maximum likelihood parameters given observations

Application of the EM algorithm (which we'll see again in RNA-seq quantification) to training of HMMs

Not guaranteed to find a global maximum

Can overfit the data i.e., possible that $P(Y \mid \theta^{*}) > P(Y \mid \theta^{real})$

However, it is an effective and widely-used algorithm for HMM training. It often works very well in practice (given sufficient, unbiased, training data)

Recap

- Hidden Markov Model (HMM) model the generation of sequences of symbols.

- They are governed by a symbol alphabet (Σ), a set of states (Q), a set of transition probabilities A, and a set of emission probabilities for each state (E).

- Given a string and an HMM, we can compute:

  The most probable path the HMM took to generate the sequence (Viterbi).

  The probability that the HMM was in a particular state at a given step (forward-backward algorithm).

- Algorithms are based on dynamic programming.

- *Finding* good parameters is a much harder problem.
  The Baum-Welch algorithm is an oft-used heuristic algorithm.

*