

Hidden Markov Models

Adapted from slides by Steven Bedrick (OHSU)

Formally, an HMM is fully described as:

$Q = q_1, q_2, q_3 \dots q_n$	A set of N hidden states
$A = a_{11}a_{12} \dots a_{n1} \dots a_{nn}$	A transition probability matrix giving the probabilities of going from state i to j
$O = o_1o_2 \dots o_T$	A sequence of T observations
$B = b_i(o_t)$	A set of observation likelihoods (aka <i>emission probabilities</i>) of observation o_t being generated from state b_i .
q_0, q_F	Special start and stop states, together with transition probabilities $a_{01} \dots$

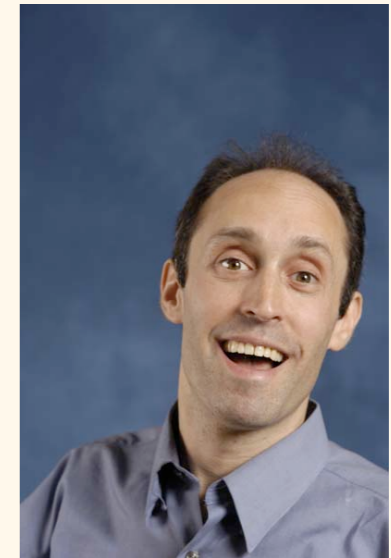
There are three fundamental kinds of questions that we can ask with an HMM:

1. *Likelihood*: Given an HMM, how likely is a given observation sequence?

2. *Decoding*: Given an observation sequence and a fully-specified HMM, what is the most likely sequence of states to have produced that observation?

3. *Learning*: Given an observation sequence and a set of states, what are the likely transition and emission probabilities (A and B)?

We will steal an example
from Jason Eisner.



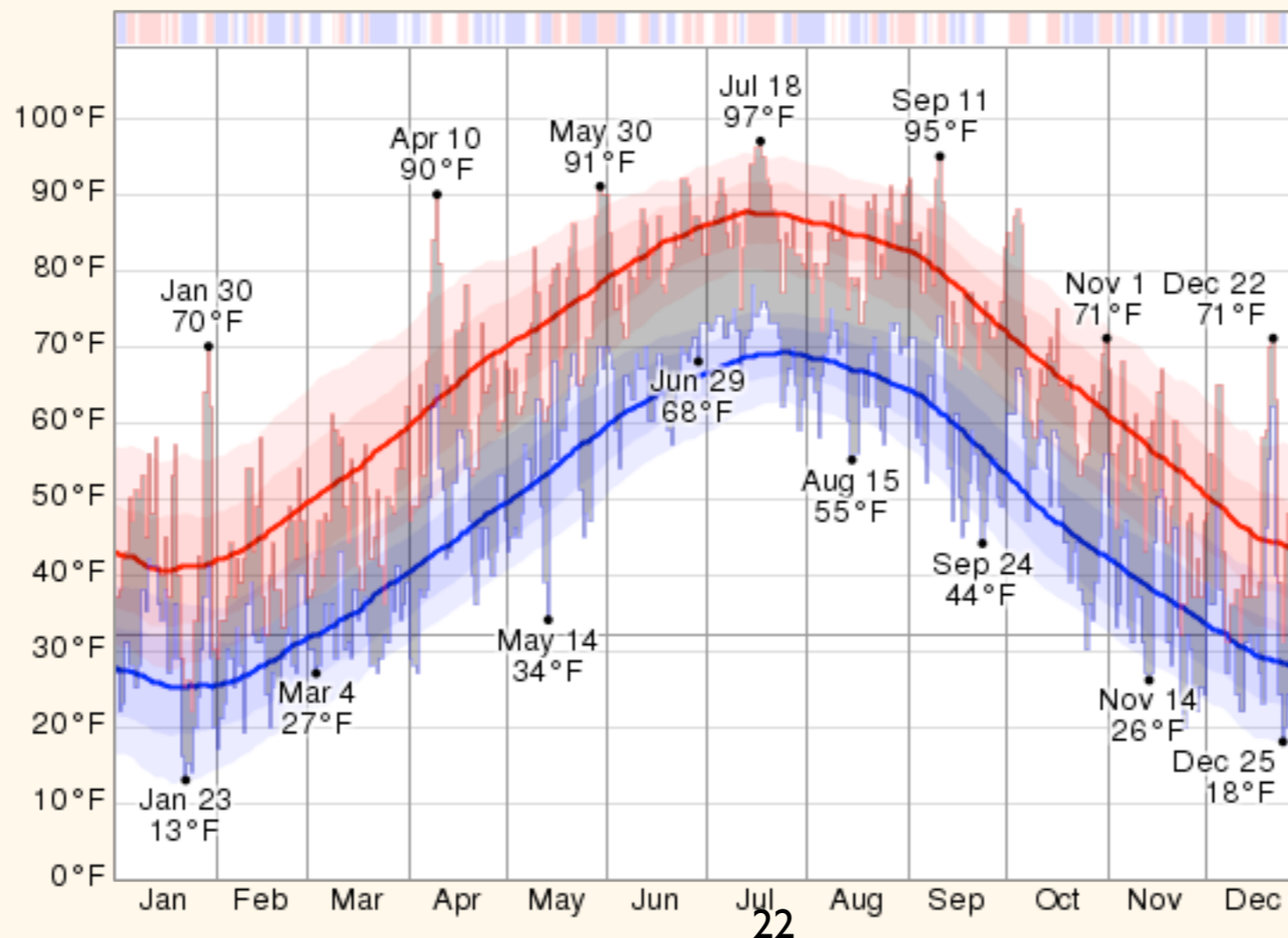
Jason Eisner
??? – Present

It is 2799; you are a climatologist studying the
history of global warming.

Following the Zombie Apocalypse of 2325, all
records of 20th-century weather were destroyed...

... however, archaeologists excavating the ruins of Baltimore recently discovered Jason's diary...

... in which he obsessively recorded how often he ate ice cream over the summer of 2002.



We can infer that the weather influenced how much ice cream Jason ate on any given day.

We can further infer that today's weather is at least somehow related to yesterday's weather.

An HMM will let us model the situation:

Observed variable: Ice cream consumption

Hidden variable: Weather

Let's simplify things and say that there are two kinds of weather ("hot" and "cold"), and that he either ate 1, 2, or 3 units of ice cream per day.

Transition matrix

A:

	Hot	Cold
Hot	0.7	0.3
Cold	0.4	0.6

Emission matrix

B:

	1	2	3
Hot	0.2	0.4	0.4
Cold	0.5	0.4	0.1

$a_{0,Hot/Cold}$:

	Start
Hot	0.8
Cold	0.2

Transition matrix

A:

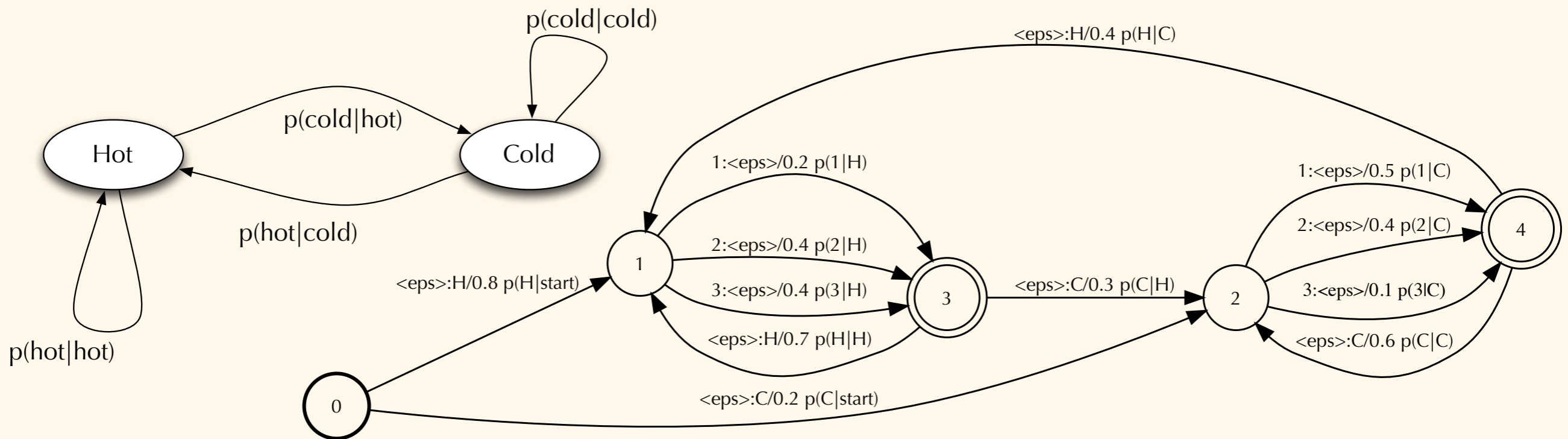
	Hot	Cold
Hot	0.7	0.3
Cold	0.4	0.6

Emission matrix

B:

	1	2	3
Hot	0.2	0.4	0.4
Cold	0.5	0.4	0.1

We can represent parts of the using WFSTs!



A note about starting and stopping conditions:

$a_{0,Hot/Cold}$:	Start
Hot	0.8
Cold	0.2

In this example, we know *a priori* that the journal is from the summer months, so $P(Hot)$ is higher than $P(Cold)$.

We don't have any reason to believe that the weather affected when Jason stopped his diary, so the stop probabilities are identical.

Can you think of an HMM problem where they might not be?
(Hint: think POS tagging)

There are three fundamental kinds of questions that we can ask with an HMM:

1. *Likelihood*: Given a sequence of states, what is the most likely observed sequence? *or*; how likely is a given observation sequence?

2. *Decoding*: Given an observation sequence and a fully-specified HMM, what is the most likely sequence of states to have produced that observation?

3. *Learning*: Given an observation sequence and a set of states, what are the likely transition and emission probabilities (A and B)?

Let's say we have a sequence of diary entries:

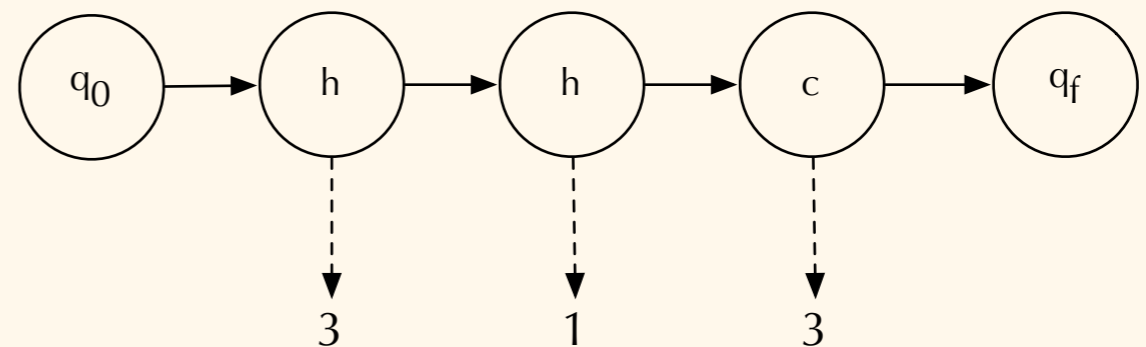
$$O = 3, 1, 3$$

How likely is this sequence given the model described earlier? $P(O|\lambda)$

We start with a simpler problem: calculating the probability of a specific observation/state sequence pair.

$$Q = \text{hot, hot, cold}$$

$$P(O|Q) = \prod_{i=1}^T P(o_i|q_i)$$



$$P(3, 1, 3|h, h, c) = P(3|h) \times P(1|h) \times P(3|c)$$

$$O = 3, 1, 3$$

$$Q = \text{hot, hot, cold}$$

$$P(O|Q) = \prod_{i=1}^T P(o_i|q_i)$$

$$P(3, 1, 3|h, h, c) = P(3|h) \times P(1|h) \times P(3|c)$$

But that's not the full story, since Q itself is only one of many sequences our machine can generate. So:

$$P(O, Q) = P(O|Q) \times P(Q) = \prod_{i=1}^n P(o_i|q_i) \times \prod_{i=1}^n P(q_i|q_{i-1})$$

$$P([3, 1, 3], [h, h, c]) = \times P(3|h) \times P(1|h) \times P(3|c) \\ P(h|start) \times P(h|h) \times P(c|h)$$

Now that we can find out the joint probability of an observation and a given state sequence...

... we know how to find the probability of the observation itself:

$$P(O) = \sum_Q P(O, Q) = \sum_Q P(O|Q)P(Q)$$

Intuition: the probability of an observation is the sum of the probabilities of all the different ways for the model to generate that observation.

$$P(3, 1, 3) = P([3, 1, 3], [h, h, h]) + P([3, 1, 3], [h, h, c]) + \\ P([3, 1, 3], [h, c, h]) + P([3, 1, 3], [c, h, h]) \dots$$

Problem: for N states and T observations, calculating $P(O)$ in this way is $O(N^T)$.

$$P(O) = \sum_Q P(O, Q) = \sum_Q P(O|Q)P(Q)$$

Often, N and T are large!*

Instead, we can use the $O(N^2T)$ *forward algorithm* to compute $P(O)$.

This is a simple instance of dynamic programming!

**Not that they have to be very large in order to cause problems!
20 states, 10 observations = tens of trillions of calculations.*

The key insight: build a *trellis* that keeps track of the probabilities of different paths through the machine.

This is represented by a T (# of states) by N (# of observations) matrix \mathbf{a} .

Each $\mathbf{a}_t(j)$ represents the probability of the machine being in state j given the first t observations (“forward probability”).

Formally: $\alpha_t(j) = P(o_1, o_2 \dots o_t, q_t = j | \lambda)$

$q_t = j$: “the t^{th} state in the sequence is state j ”

Calculating $\alpha_t(j) = P(o_1, o_2 \dots o_t, q_t = j | \lambda)$ is fairly straightforward:

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

Calculating $\alpha_t(j) = P(o_1, o_2 \dots o_t, q_t = j | \lambda)$ is fairly straightforward:

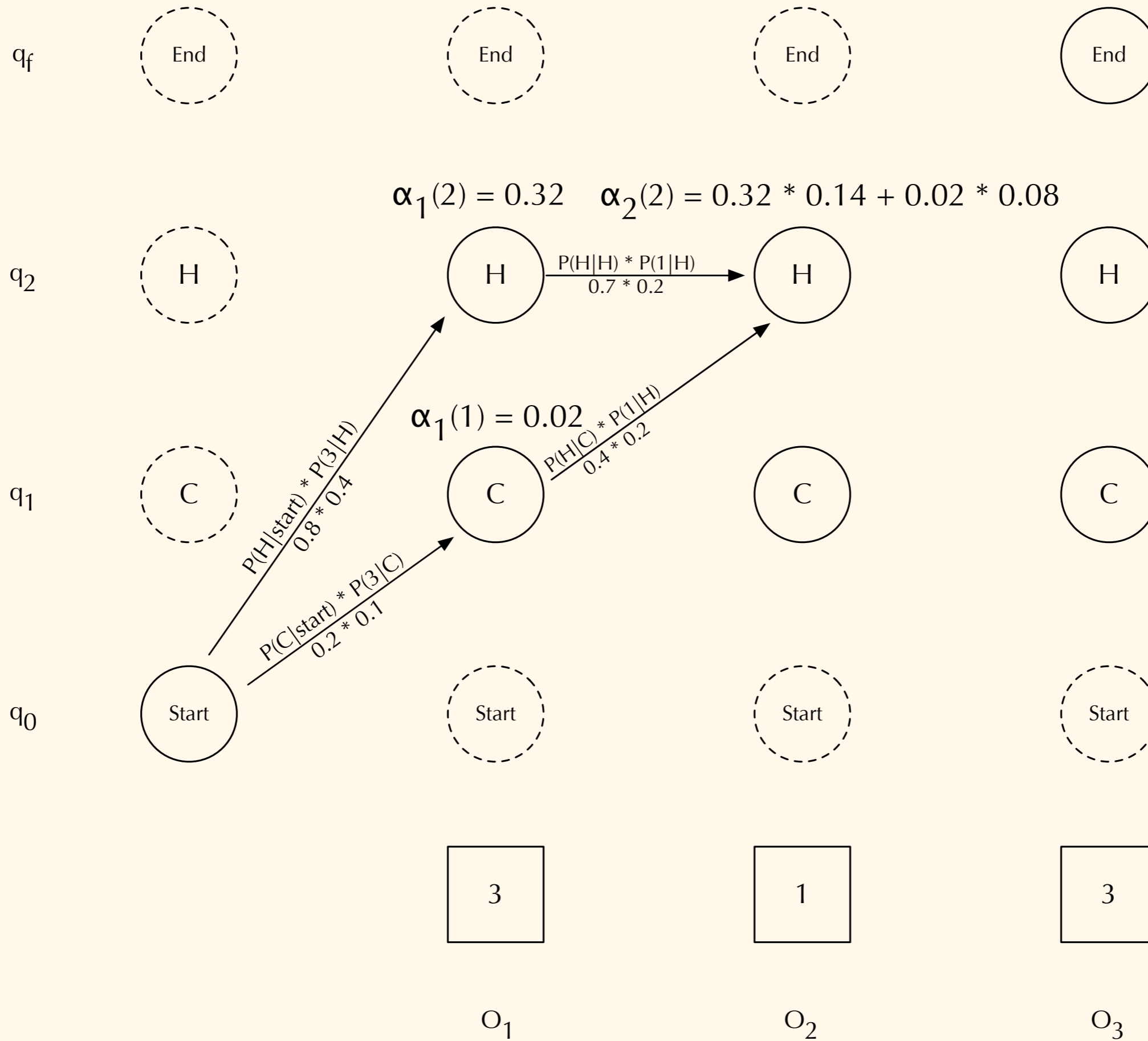
$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

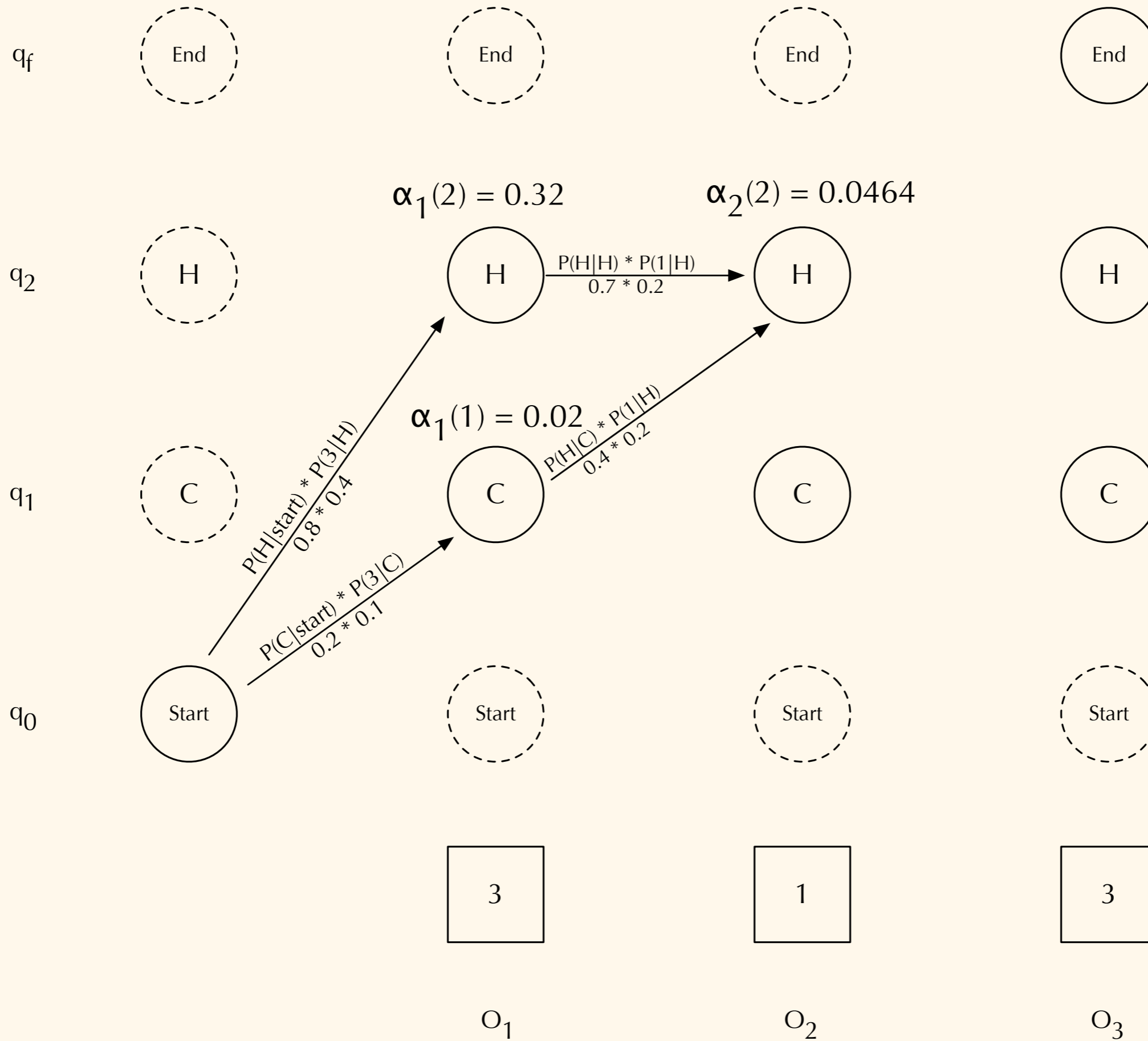
For each possible hidden state...

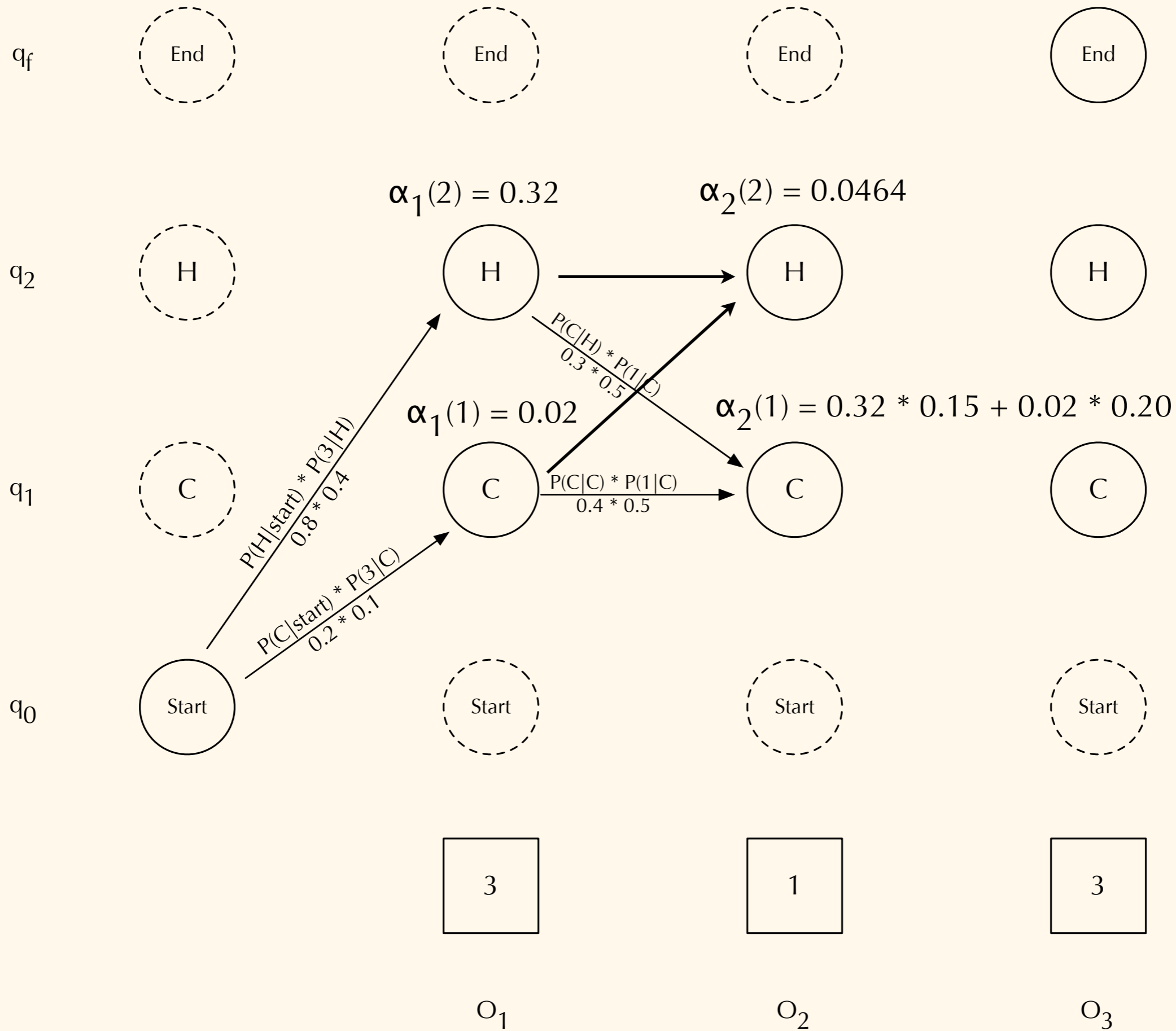
Previous time step's forward probability for state i

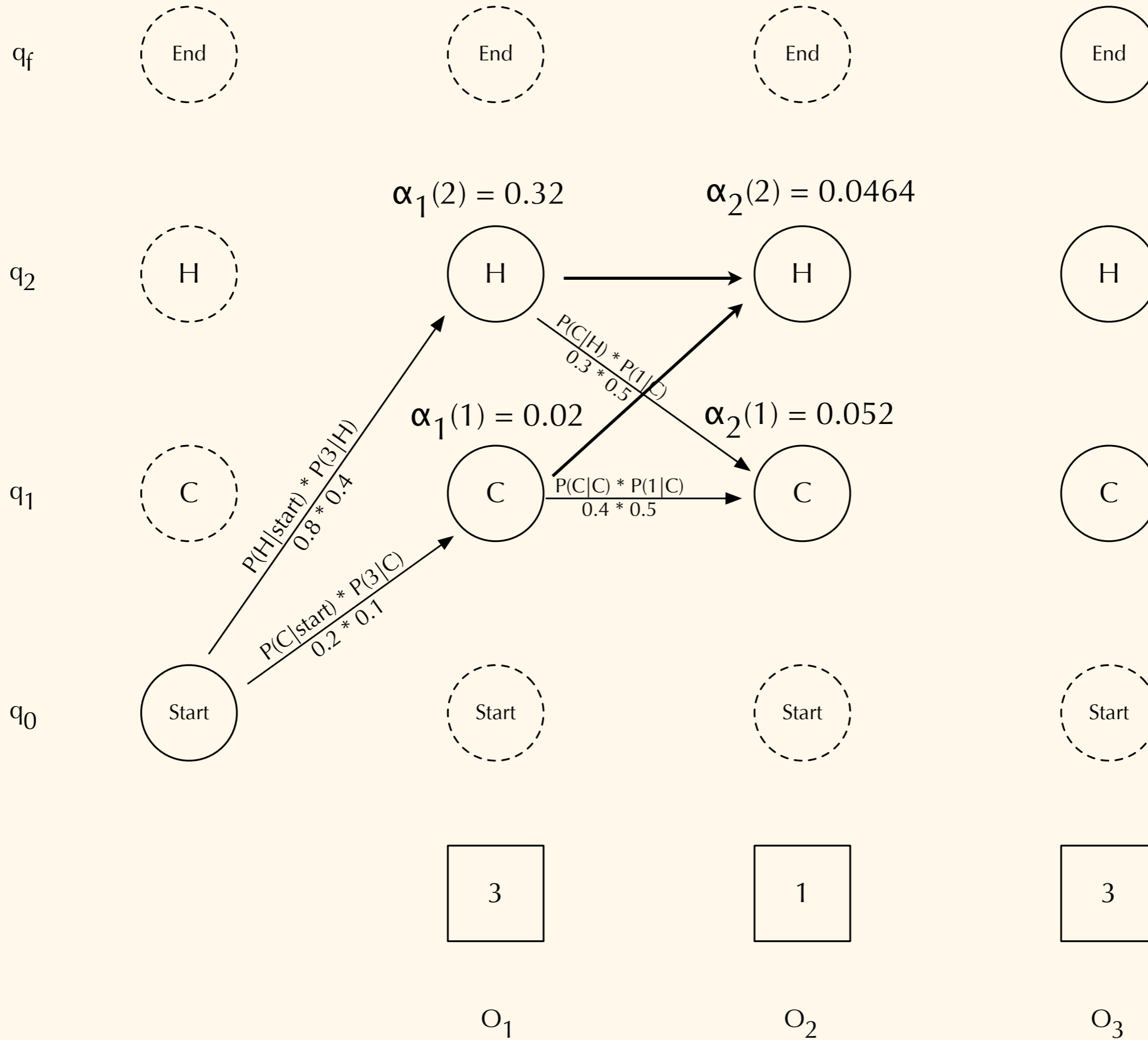
Transition prob. from previous state i to current state j

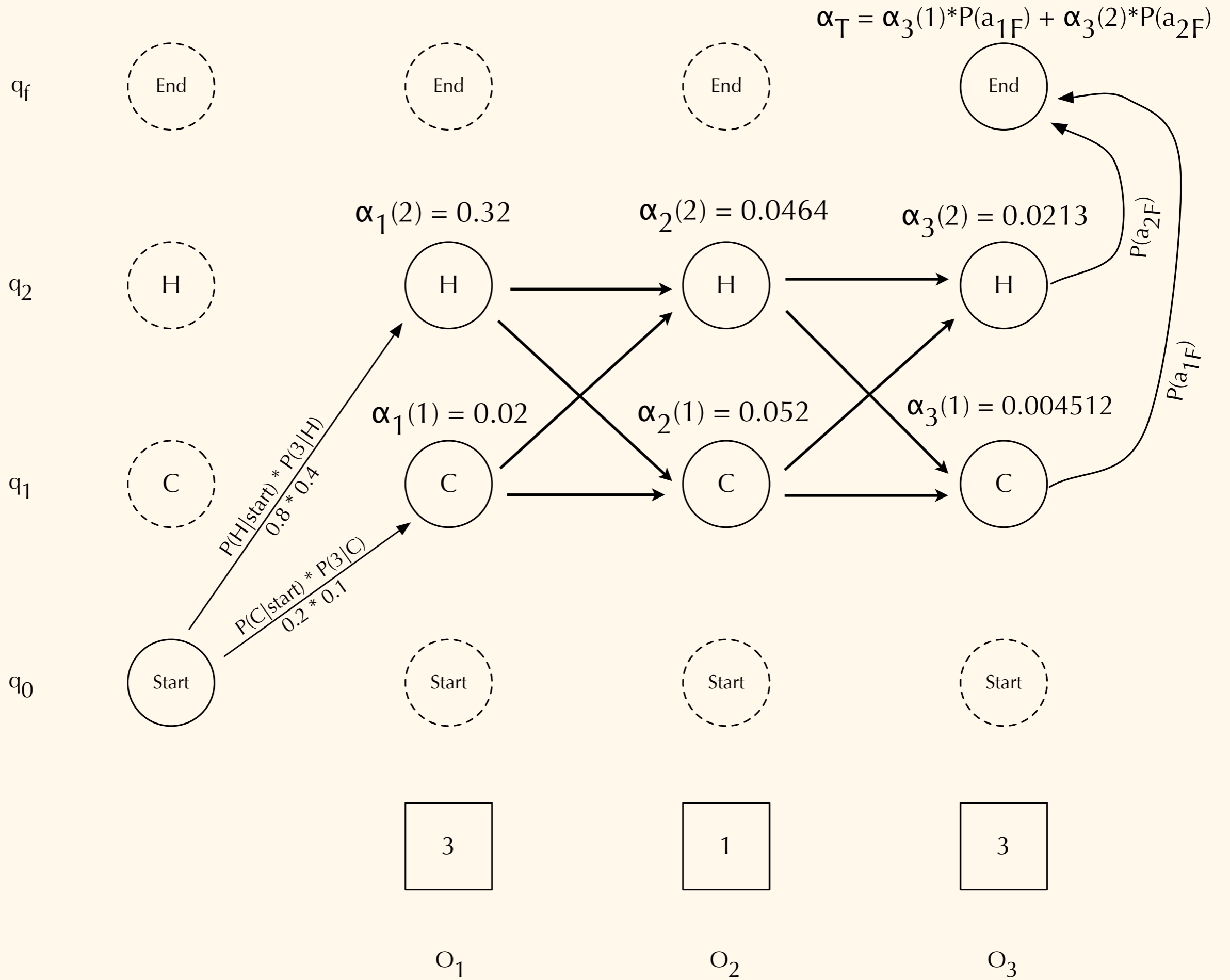
Emission likelihood for symbol o_t given current state j

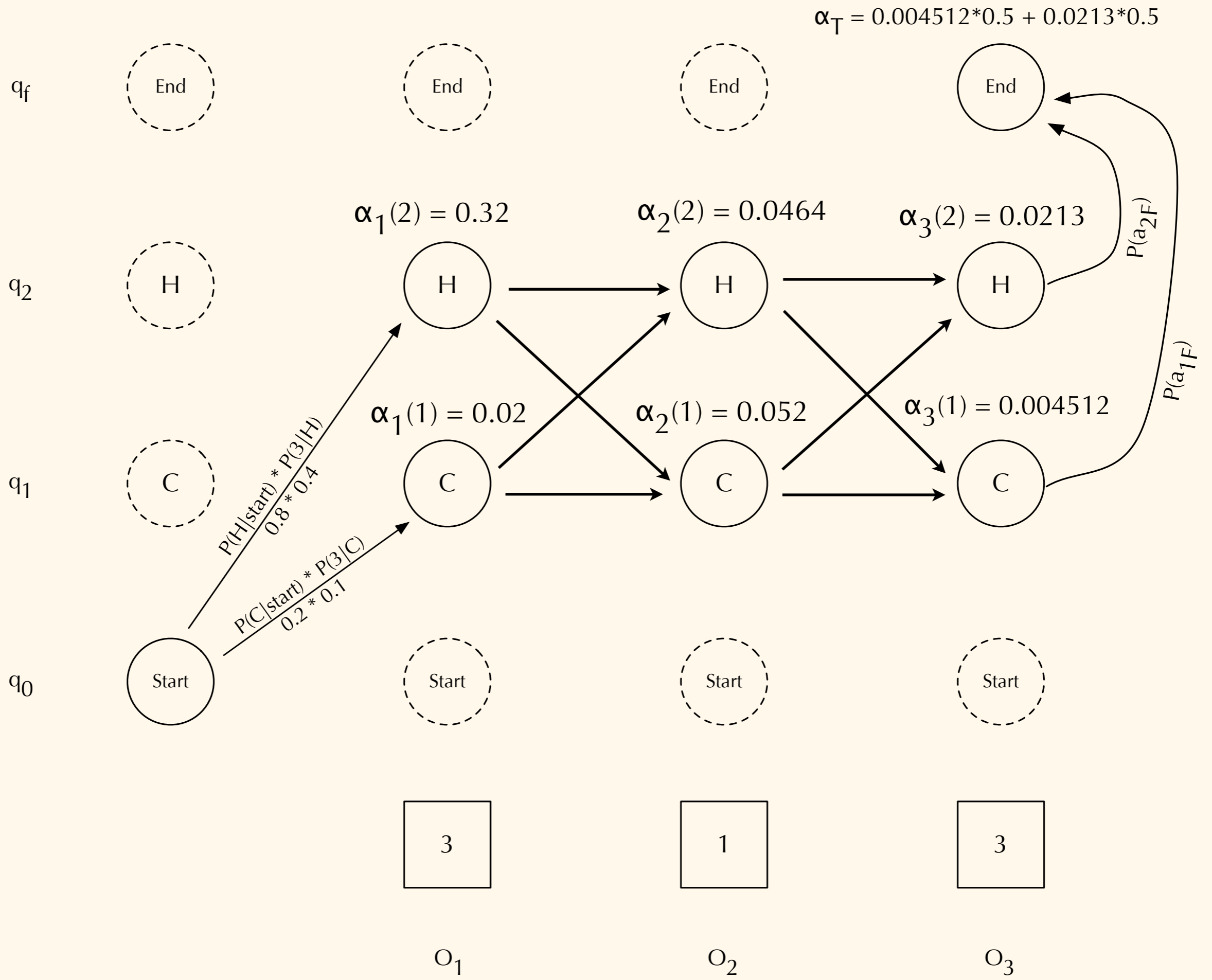


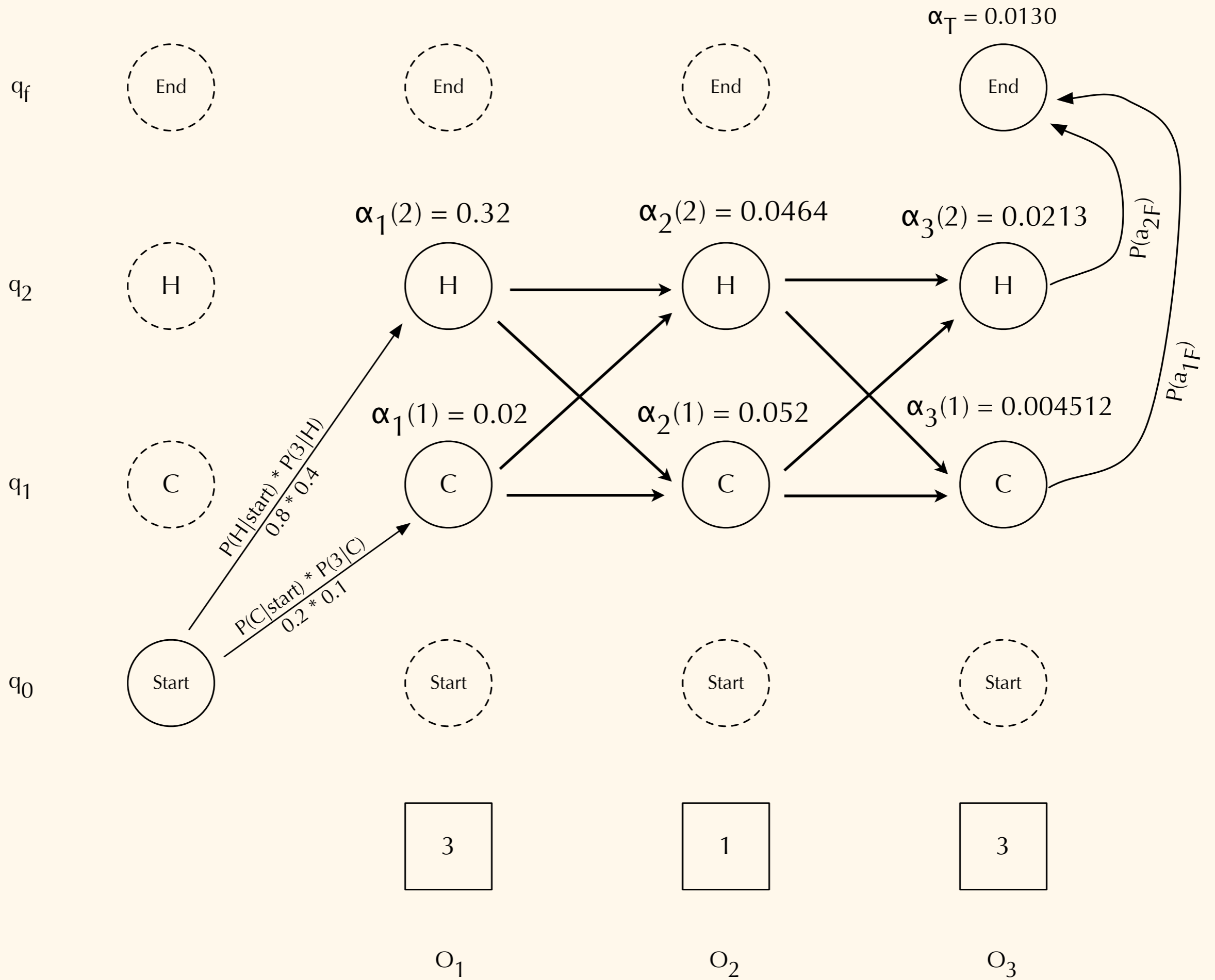












There are three fundamental kinds of questions that we can ask with an HMM:

1. *Likelihood*: Given a sequence of states, what is the most likely observed sequence? *or*; how likely is a given observation sequence?

2. *Decoding*: Given an observation sequence and a fully-specified HMM, what is the most likely sequence of states to have produced that observation?

3. *Learning*: Given an observation sequence and a set of states, what are the likely transition and emission probabilities (A and B)?

There are three fundamental kinds of questions that we can ask with an HMM:

1. *Likelihood*: Given a sequence of states, what is the most likely observed sequence? *or*; how likely is a given observation sequence?

2. *Decoding*: Given an observation sequence and a fully-specified HMM, what is the most likely sequence of states to have produced that observation?

3. *Learning*: Given an observation sequence and a set of states, what are the likely transition and emission probabilities (A and B)?

Formally: $\operatorname{argmax}_Q P(Q|O)$

“Given an observation O , what was the most probable sequence of states Q ?”

Decoding and likelihood estimation have certain similarities...

One solution: run the forward algorithm over each possible state sequence...

... which has the same issue as the naïve solution to the likelihood problem!

$O(N^T)$ possible solutions...

Decoding and likelihood estimation have the same problems...

... and they share a solution.

Modifying the forward algorithm slightly gives us the *Viterbi algorithm* for decoding.

The main difference: instead of *summing* possible paths to each state, we take the *max*...

... and *keep track* of which one it was!

Forward algorithm trellis locations:

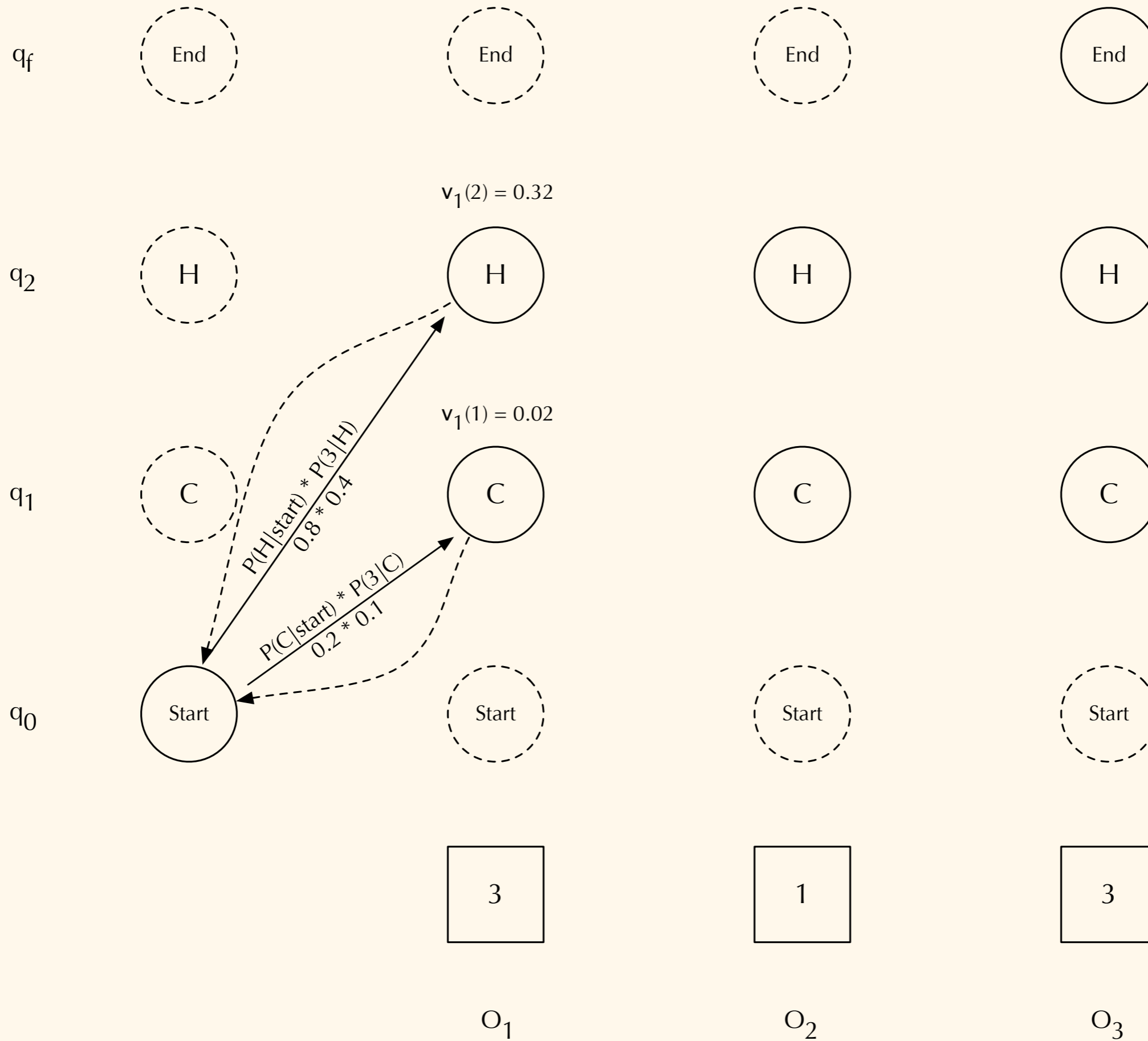
$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

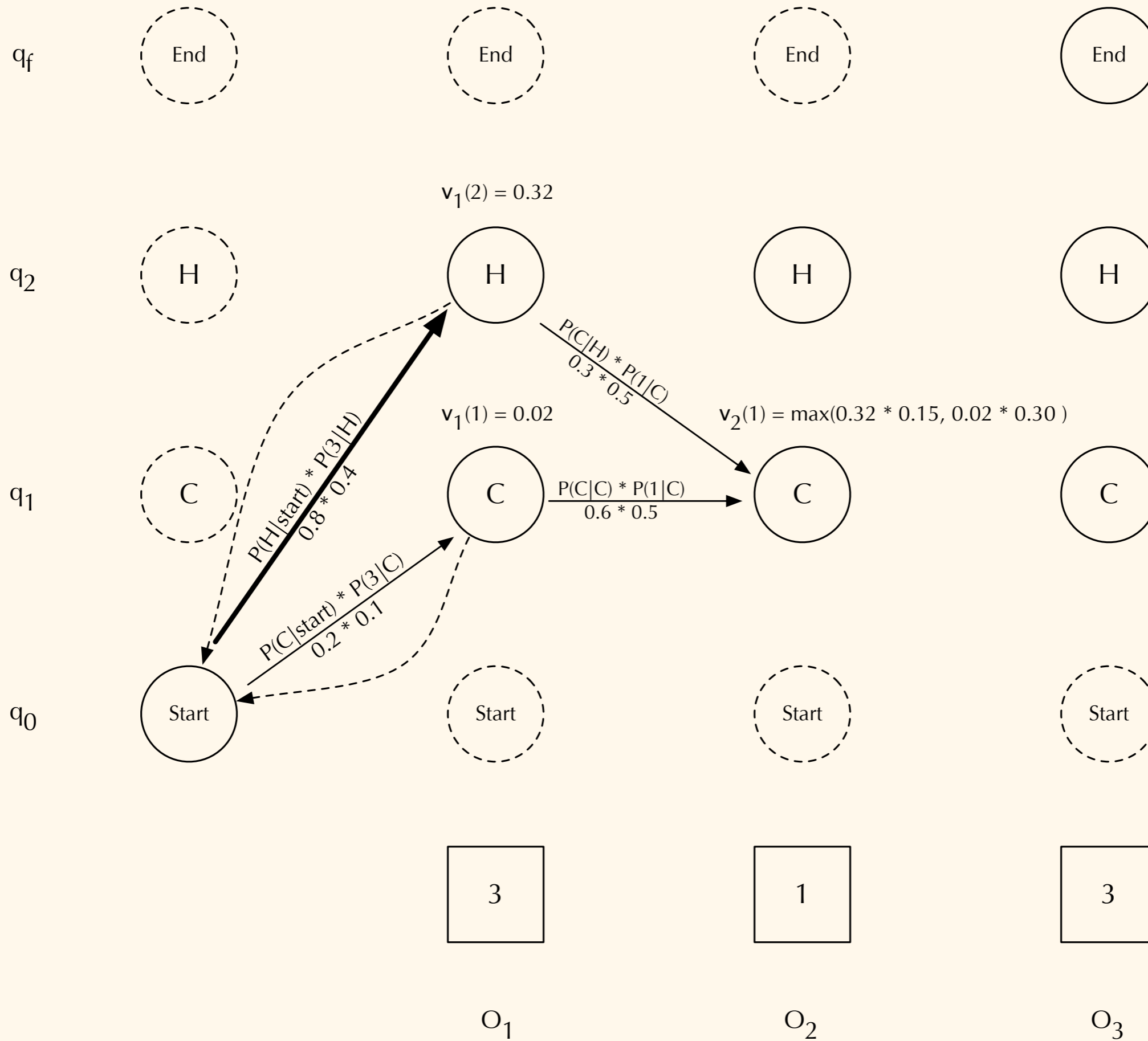
Viterbi algorithm trellis locations:

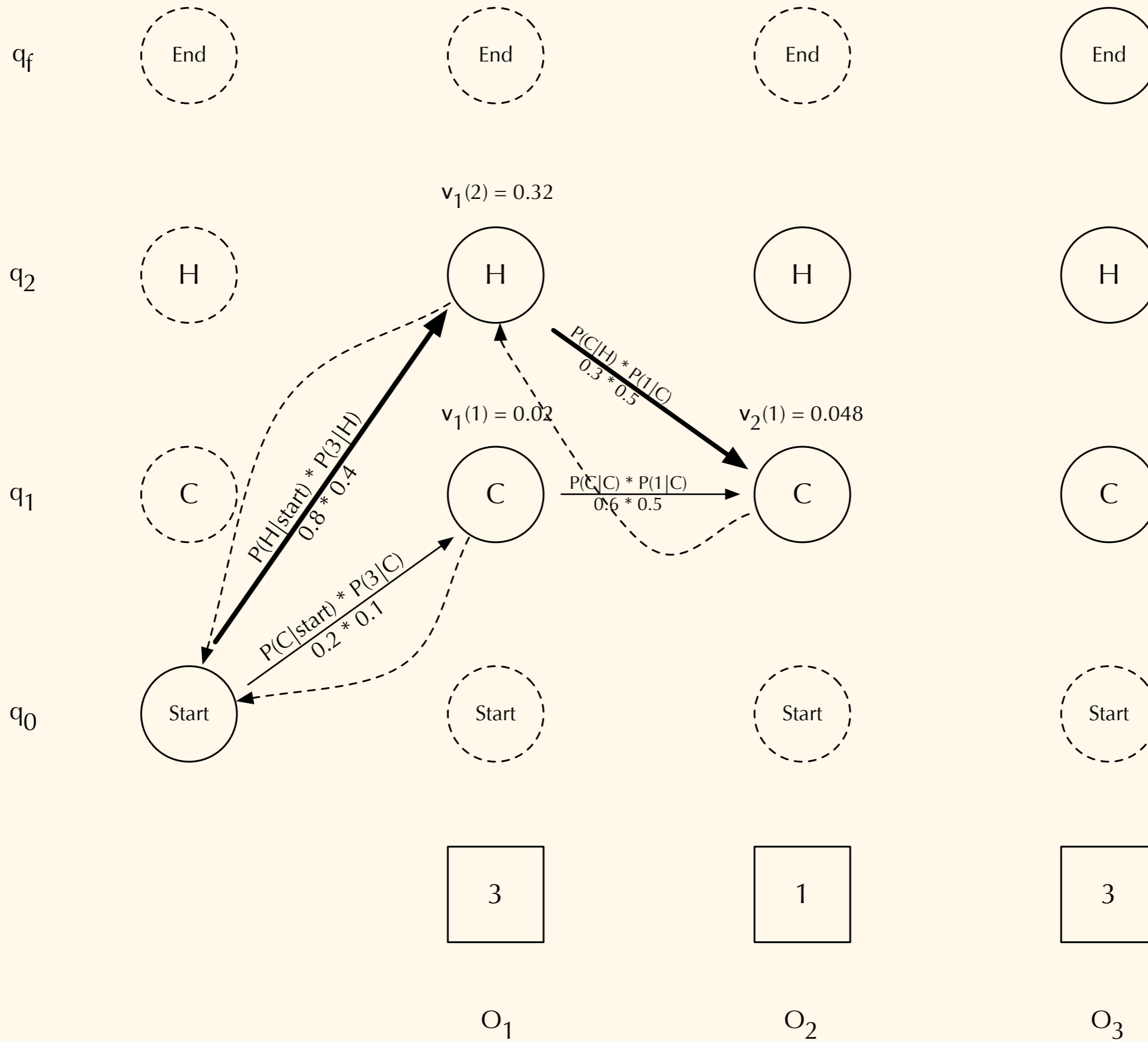
$$v_t(j) = \max_{i=1}^N v_{t-1} a_{ij} b_j(o_t)$$

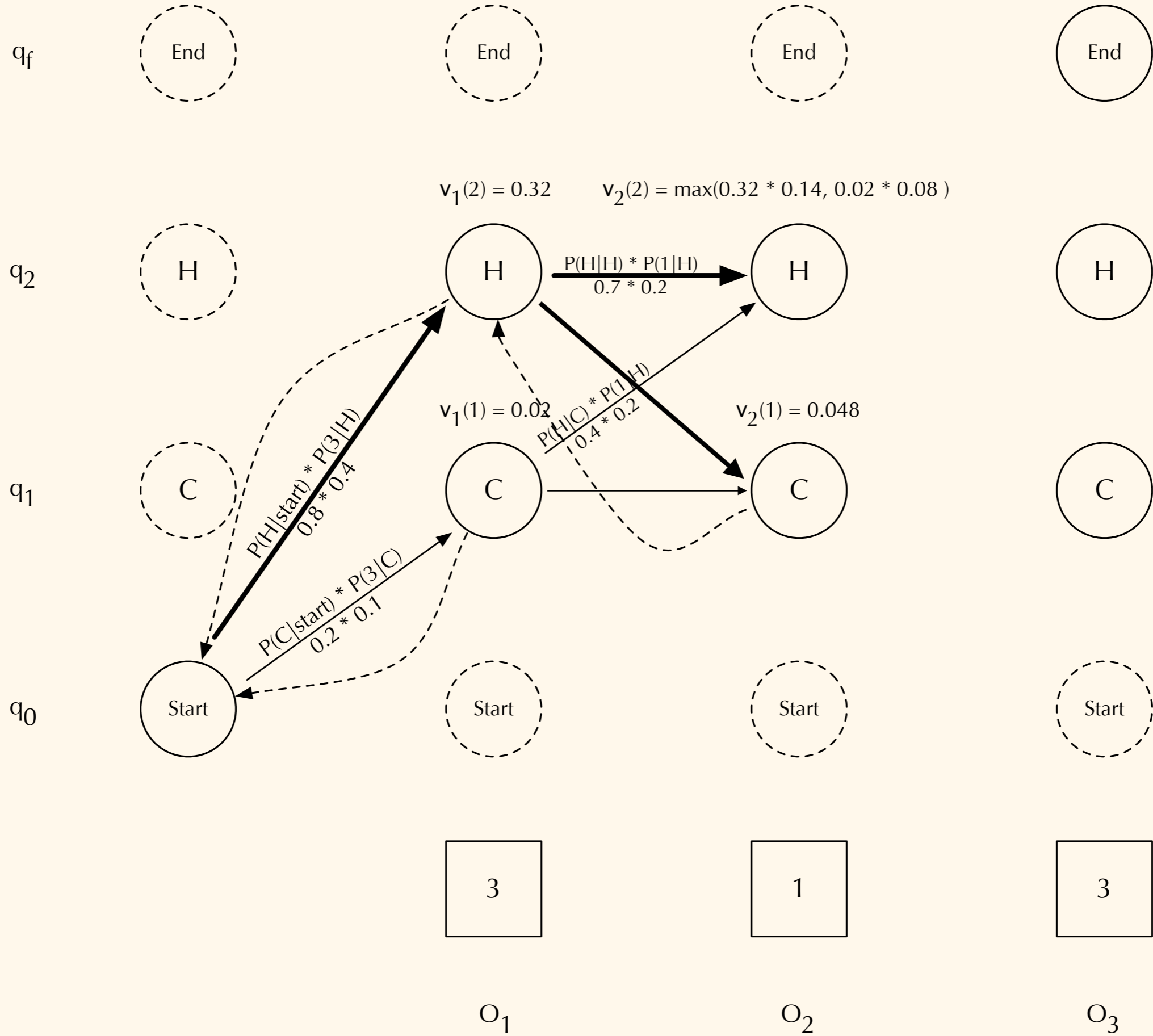
We also save a backtrace through the most-likely states:

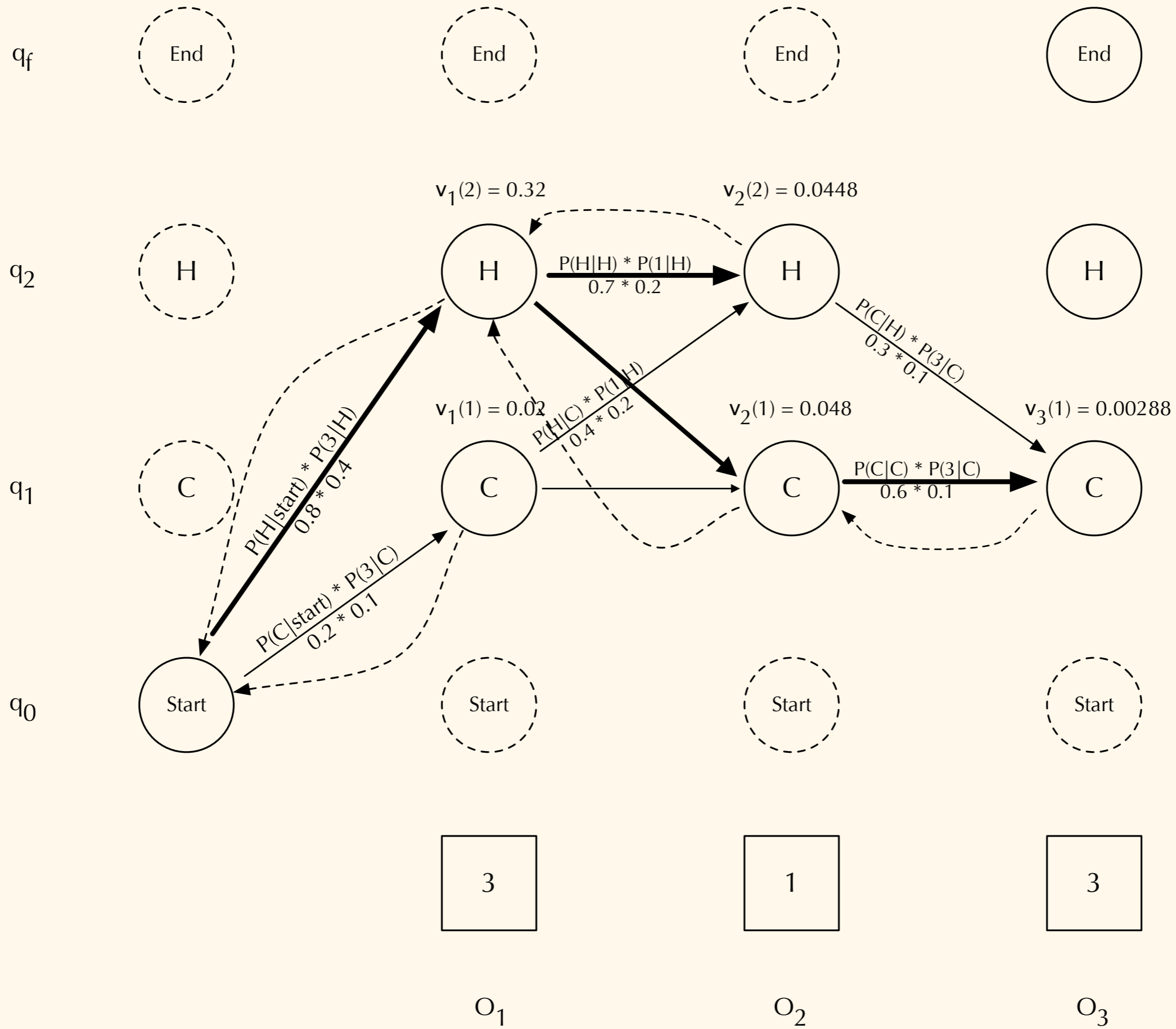
$$bt_t(j) = \operatorname{argmax}_{i=1}^N v_{t-1} a_{ij} b_j(o_t)$$

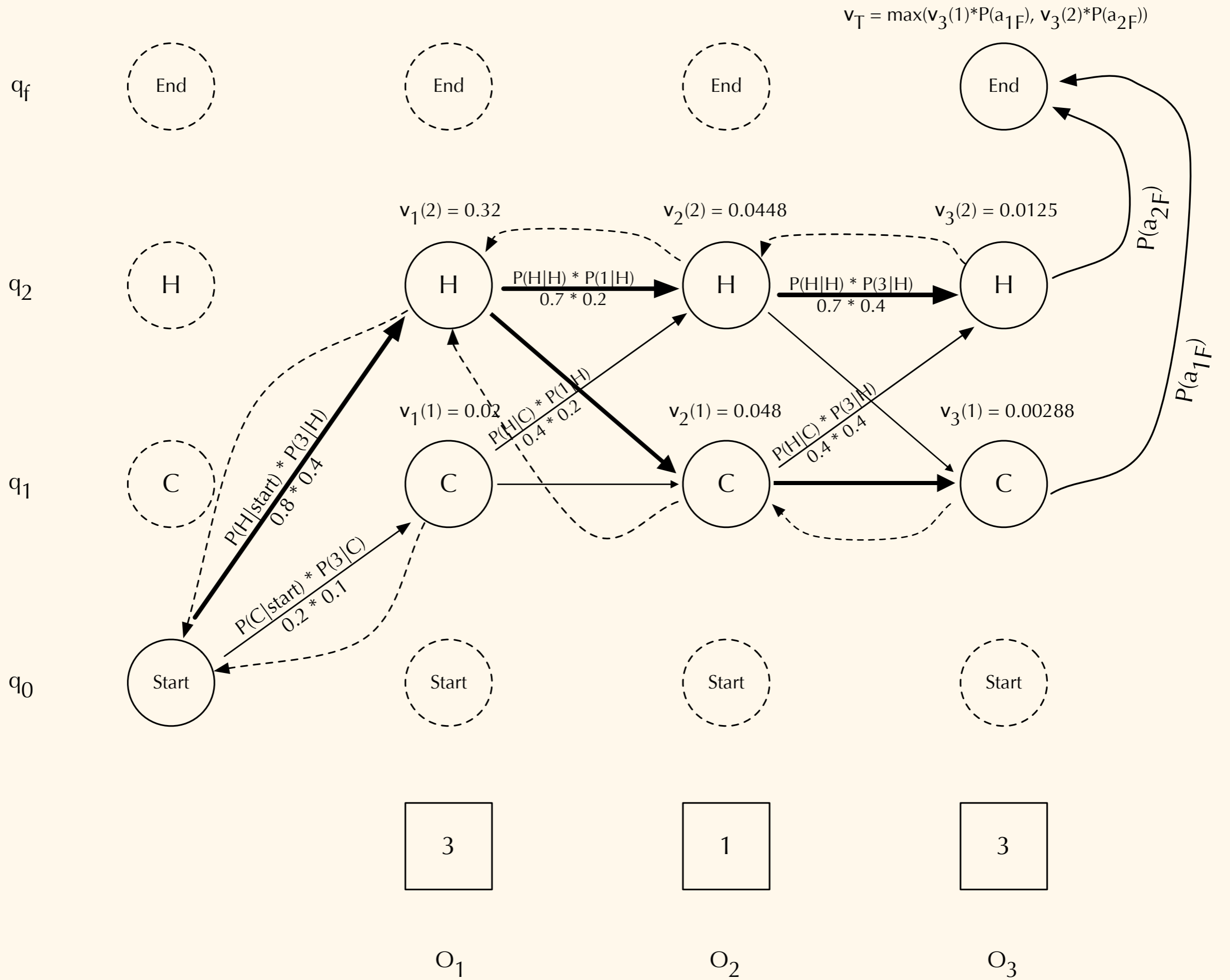


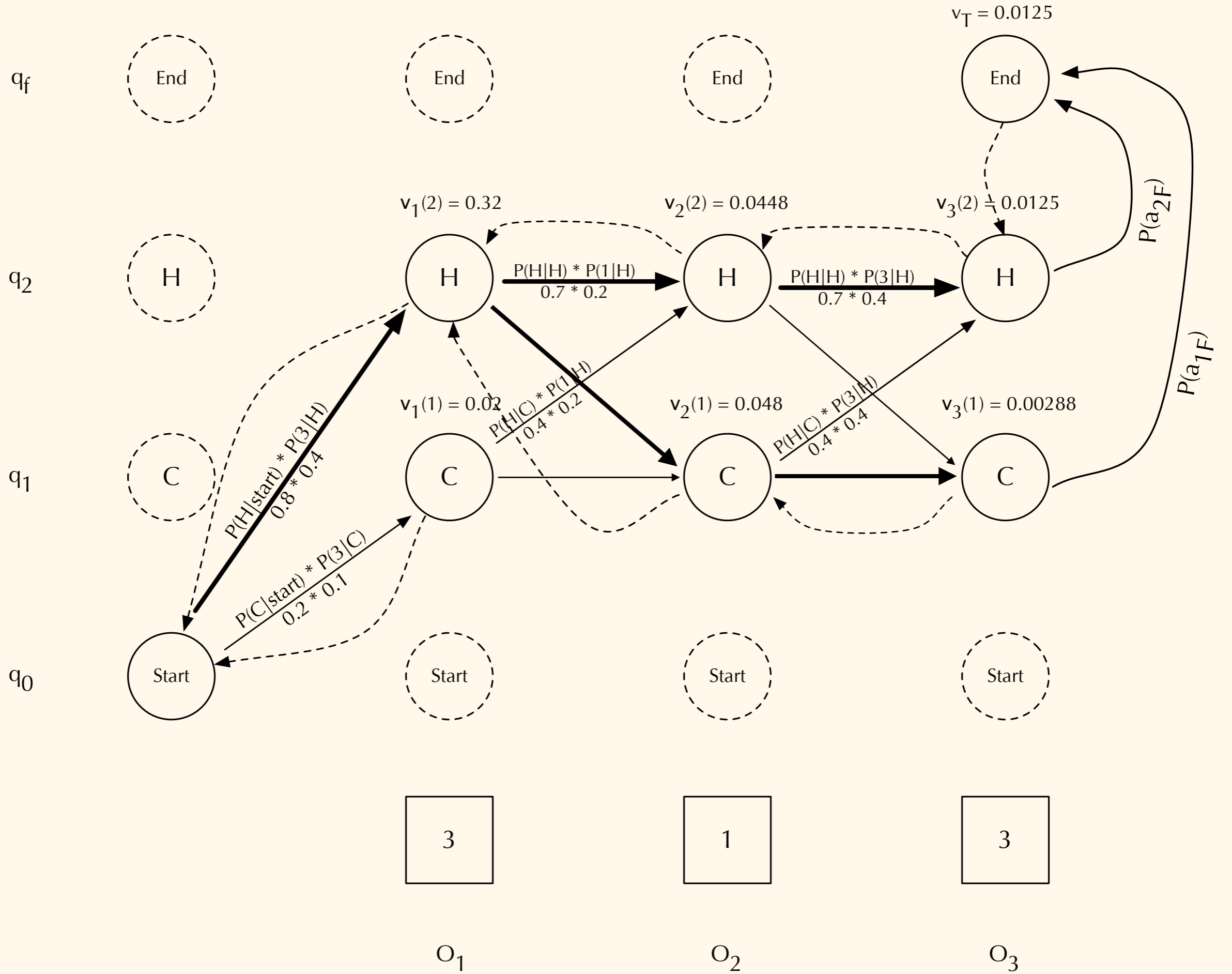












There are three fundamental kinds of questions that we can ask with an HMM:

1. *Likelihood*: Given a sequence of states, what is the most likely observed sequence? *or*; how likely is a given observation sequence?

2. *Decoding*: Given an observation sequence and a fully-specified HMM, what is the most likely sequence of states to have produced that observation?

3. *Learning*: Given an observation sequence and a set of states, what are the likely transition and emission probabilities (A and B)?

There are three fundamental kinds of questions that we can ask with an HMM:

1. *Likelihood*: Given a sequence of states, what is the most likely observed sequence? *or*; how likely is a given observation sequence?
2. *Decoding*: Given an observation sequence and a fully-specified HMM, what is the most likely sequence of states to have produced that observation?
3. *Learning*: Given an observation sequence and a set of states, what are the likely transition and emission probabilities (A and B)?

So, we have O , and know what our state vocabulary is...

... but we don't know transition or emission probabilities.

We can use *Baum-Welch algorithm* (a.k.a. *Forward-Backward algorithm*) to iteratively estimate A and B .



Leonard Baum



Lloyd Welch

Recall from before that the *forward probability* $\alpha_t(i)$ is the probability of ending up in state i given observations $O_{1:t}$.

A related property is the *backward probability* $\beta_t(i)$, which represents the probability of seeing observations $O_{t:T}$, given that we are currently in state i at time t .

This is calculated using the *backward algorithm*, which is very similar to the forward algorithm (but in reverse!).

This is calculated using the *backward algorithm*, which is very similar to the forward algorithm (but in reverse!)

$\beta_T(i) = a_{i,F}$ Probability of finishing (i.e., reaching end state) the observed sequence from state i .

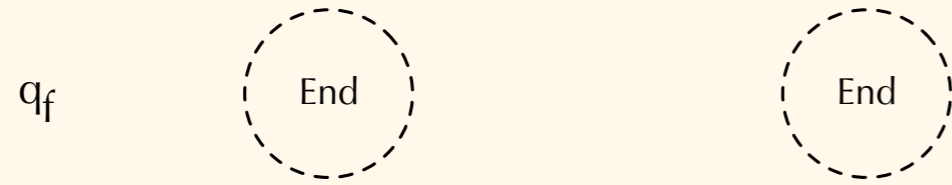
$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

Sum of the backwards probabilities of the different paths through the model that could happen from state i and time t .

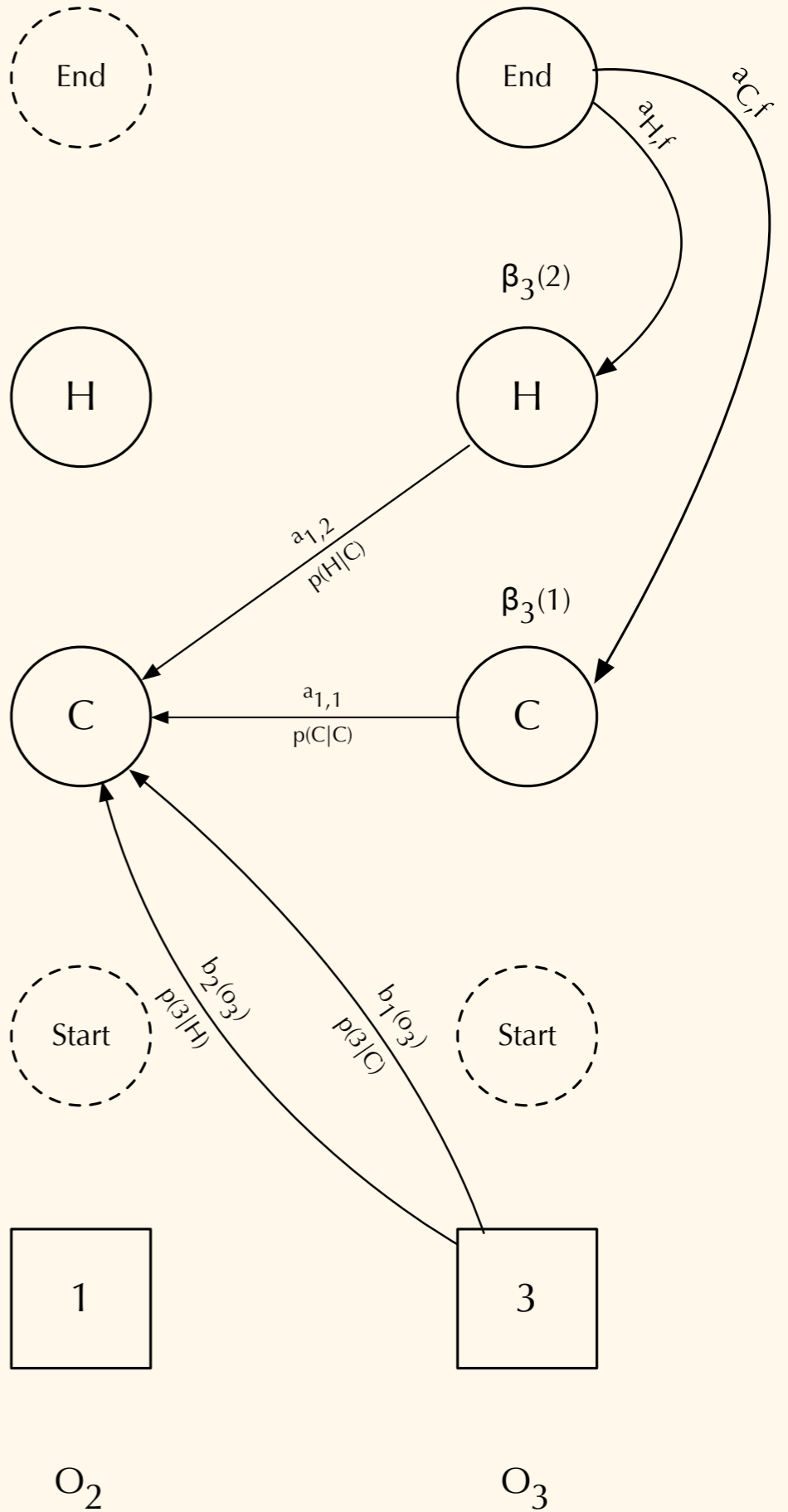
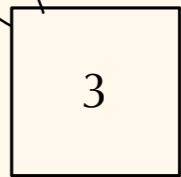
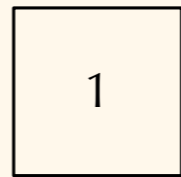
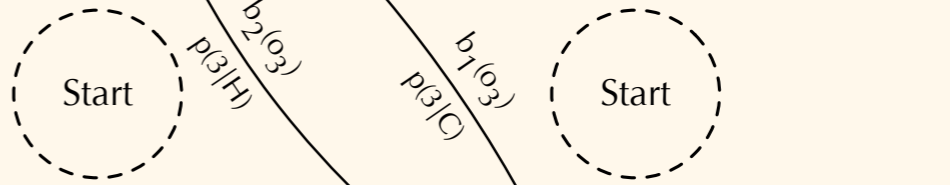
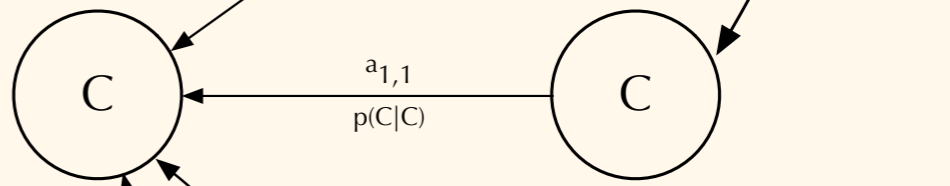
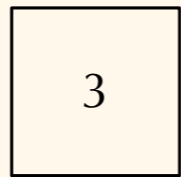
$$P(O|\lambda) = \alpha_T(q_F) = \beta_1(0) = \sum_{j=1}^N a_{0j} b_j(o_1) \beta_1(j)$$

Final forward probability of observation given model.

Of course, we compute all of this using the same dynamic programming approach we've already seen:



$$\beta_2(1) = \beta_3(1) * p(C|C) * P(3|C) + \beta_3(2) * p(H|C) * p(3|H)$$



$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

Now, how to use these to estimate A and B ?

Baum-Welch is a variation on *Expectation-Maximization*...

... as such, we start with a “guess” for A and B , and iteratively improve it.

We begin by attempting to find:


$$\hat{a}_{ij} = \frac{\text{expected } \# \text{ transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

If we had an estimate of the probability of transition $i \rightarrow j$ occurring at each time t , we could sum them to get the total count for $i \rightarrow j$.

More formally: let $\xi_t(i, j)$ be the probability of being in state i at time t and in state j at time $t+1$.

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j | O, \lambda)$$

We can't quite calculate this, but we *can* calculate something similar:

$$\tilde{\xi}_t(i, j) = P(q_t = i, q_{t+1} = j, O | \lambda)$$


We have all the pieces we need to get this:

$$\tilde{\xi}_t(i, j) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

We have all the pieces we need to get this:

$$\tilde{\xi}_t(i, j) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

Forward probability of
observations up to this arc

Transition probability
between states i and j

Emission probability
of the next symbol

Backward probability
after this arc

Furthermore, because of $P(X|Y, Z) = \frac{P(X, Y|Z)}{P(Y|Z)}$

We can transform $\tilde{\xi}_t(i, j)$, or $P(q_t = i, q_{t+1} = j, O|\lambda)$

into $\xi_t(i, j)$, or $P(q_t = i, q_{t+1} = j|O, \lambda)$, simply by dividing by $P(O|\lambda)$.

$$P(O|\lambda) = \alpha_T(q_F) = \beta_1(0) = \sum_{j=1}^N a_{0j} b_j(o_1) \beta_1(j)$$

So, the final equation is:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)}$$

Remember:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)}$$

“The probability of going from state i to state j at time t .
(given a current estimate of the model).”

Summing over *all* times t gives us \hat{a}_{ij} :

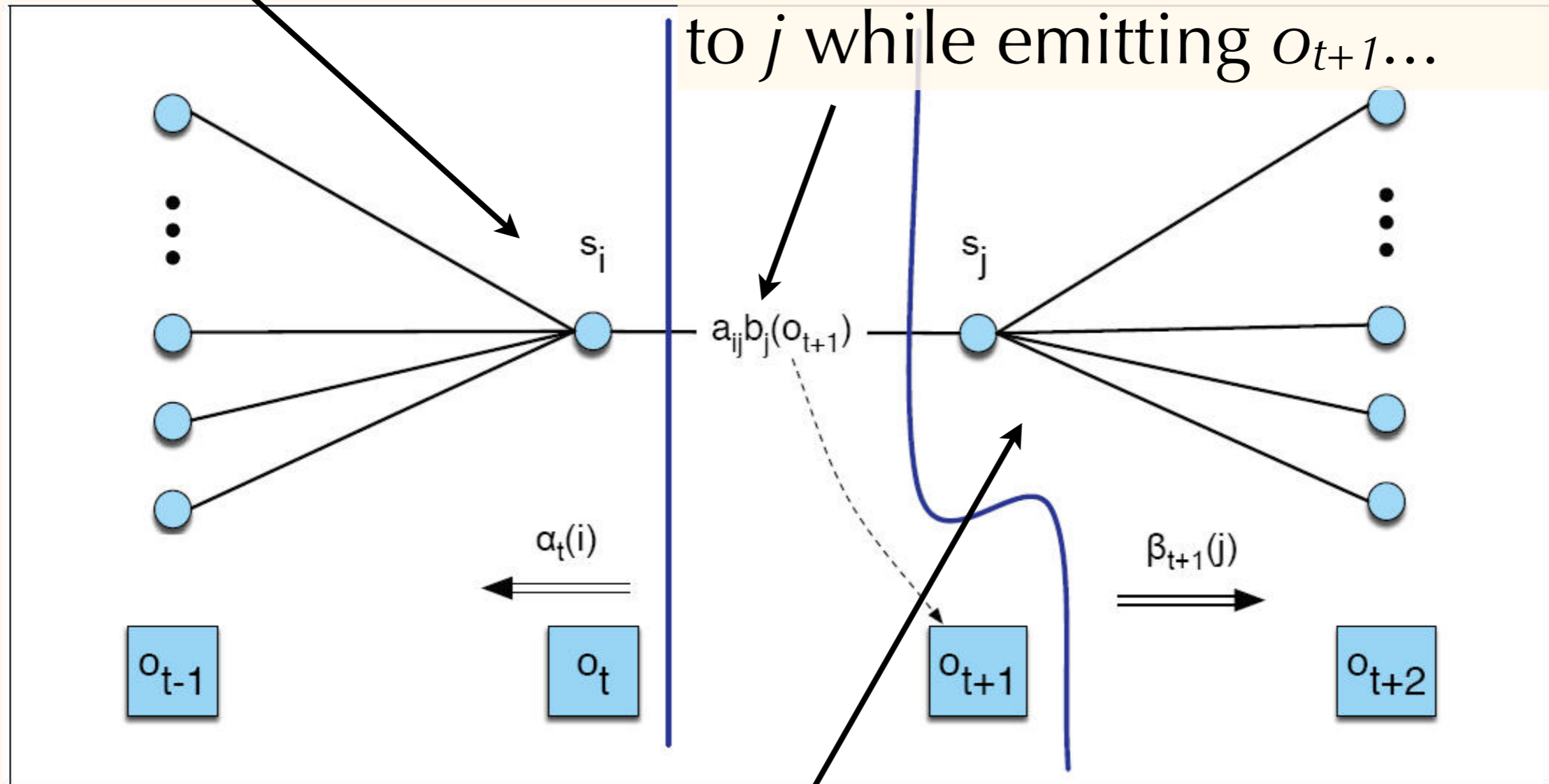
$$\hat{a}_{ij} = \frac{\sum_{t=1}^T \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)}$$

Which looks a lot like:

$$\hat{a}_{ij} = \frac{\text{expected \# transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

All of the ways the model could have gotten into state i at time t ...

... the likelihood of going from i to j while emitting O_{t+1} ...



... all of the ways the model could finish from state j at time $t+1$.

We follow a similar process to estimate B .

$$\hat{b}_j(v_k) = \frac{\text{expected \# times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

We'll need to know the probability of being in state j at time t :

$$\gamma_t(j) = P(q_t = j | O, \lambda)$$

Using the same trick as before:

$$\gamma_t(j) = \frac{P(q_t = j, O | \lambda)}{P(O | \lambda)} \quad P(q_t = j, O | \lambda) = \alpha_t(j)\beta_t(j)$$

“Probability of getting to this state at this time point, times the probability of the rest of the observations given this state and this time point”

Using the same trick as before:

$$\gamma_t(j) = \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)} \quad P(q_t = j, O|\lambda) = \alpha_t(j)\beta_t(j)$$

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)}$$

“Probability of getting to this state at this time point, times the probability of the rest of the observations given this state and this time point”

$$\hat{b}_j(v_k) = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

“Only count observations where the observed emission was v_k .”

$\approx \frac{\text{“Total probability mass from being in this state and observing this symbol”}}{\text{“Total probability mass from being in this state”}}$

Now, that we have new estimates for A and B ...

$$\hat{a}_{ij} = \frac{\sum_{t=1}^T \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)} \quad \hat{b}_j(v_k) = \frac{\sum_{t=1 \text{ s.t. } O_t=v_k}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

We can go back, calculate *new* forward and backward trellises based on these estimates, and re-compute A and B .

Wash, rinse, and repeat until things converge...
or we get bored.

In practice, much depends on our initial estimates, and so we often use additional information when possible (e.g., encoding impossible transitions, etc.).