Hidden Markov Models

Let's play a game

- I will write a sequence of part-of-speech tags and of words on the board.

- You take turns in giving me POS tags and words, and I will write them down.

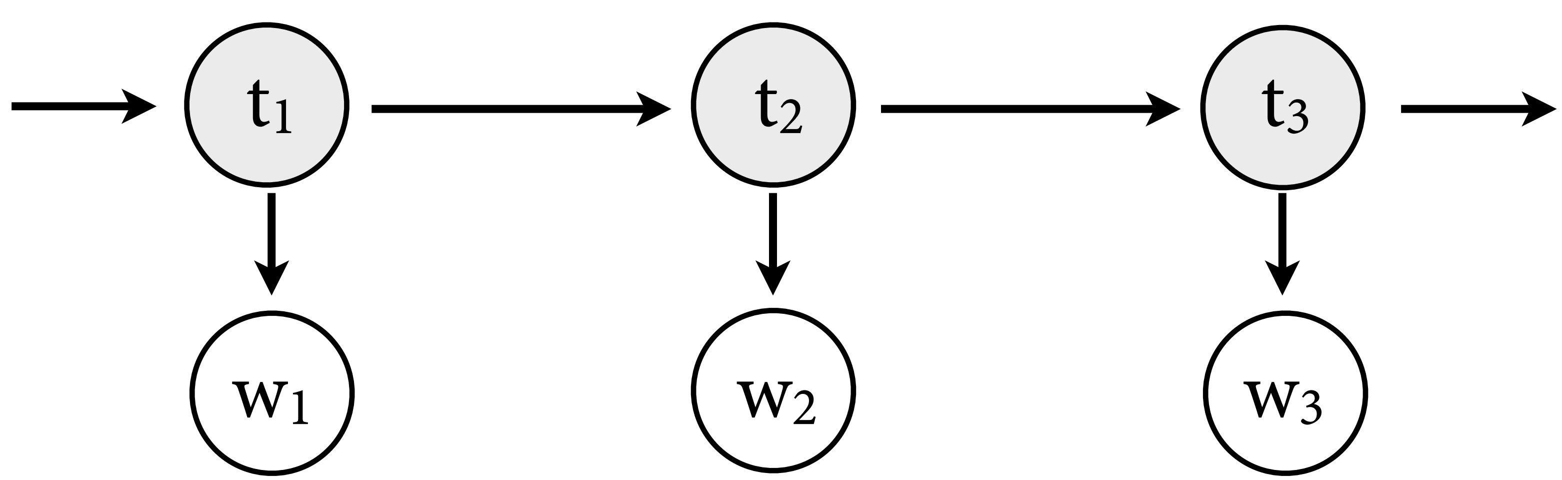
Penn Treebank POS tags

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	“to”	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential ‘there’	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	$
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	“	Left quote	(‘ or “)
POS	Possessive ending	*’s*	”	Right quote	(’ or ”)
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one’s*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Hidden Markov Models

- Last week's generative story: generate words at random from n-gram model $P(w_n \mid w_1,\ldots,w_{n-1})$.

- Replace with new generative story:
  - Language is generated by a two-step process.
  - First, generate sequence of hidden POS tags $t_1, \ldots, t_T$ tag by tag, left to right from bigram model $P(t_i \mid t_{i-1})$.
  - Independently, generate an observable word w_i from each t_i, at random from model $P(w_i \mid t_i)$.



Question 1: Language modeling

- Given an HMM and a string $w_1, \ldots, w_T$, what is the likelihood $P(w_1 \ldots w_T)$?

- We can compute $P(w_1 \ldots w_T)$ efficiently with the *forward algorithm.*

DT	NN	VBD	NNS	IN	DT	NN	
The	representative	put	chairs	on	the	table.	p_1
DT	JJ	NN	VBZ	IN	DT	NN	
The	representative	put	chairs	on	the	table.	p_2

$P(\text{sentence}) = p_1 + p_2$

Question 2: Tagging (aka Decoding)

- Given an HMM and an observed string $w_1, \ldots, w_T$, what is the most likely sequence of hidden tags $t_1, \ldots, t_T$?

- We can compute $\underset{t_1,\ldots,t_T}{\arg\max}\, P(t_1, w_1, \ldots, t_T, w_T)$

  efficiently with the *Viterbi algorithm.*

DT	NN	VBD	NNS	IN	DT	NN	
The	representative	put	chairs	on	the	table.	p_1
DT	JJ	NN	VBZ	IN	DT	NN	
The	representative	put	chairs	on	the	table.	p_2

Question 2: Tagging (aka Decoding)

- Given an HMM and an observed string $w_1, \ldots, w_T$, what is the most likely sequence of hidden tags $t_1, \ldots, t_T$?

- We can compute $\arg\max_{t_1,\ldots,t_T} P(t_1, w_1, \ldots, t_T, w_T)$

  efficiently with the *Viterbi algorithm*.

DT	NN	VBD	NNS	IN	DT	NN	
The	representative	put	chairs	on	the	table.	p_1

DT	JJ	NN	VBZ	IN	DT	NN	
The	representative	put	chairs	on	the	table.	p_2

Question 3a: Learning

- Given a set of POS tags and *annotated* training data $(w_1,t_1), \ldots, (w_T,t_T)$, compute parameters for HMM that maximize likelihood of training data.

- Do it efficiently with maximum likelihood training plus smoothing.

DT	NN	VBD	NNS	IN	DT	NN
The	representative	put	chairs	on	the	table.

NNP	VBZ	VBN	TO	VB	NR
Secretariat	is	expected	to	race	tomorrow.

→ next time

Question 3b: Learning

- Given a set of POS tags and *unannotated* training data $w_1, \ldots, w_T$, compute parameters for HMM that maximize likelihood of training data.

- Do it with the *forward-backward algorithm* (an instance of *Expectation Maximization*).

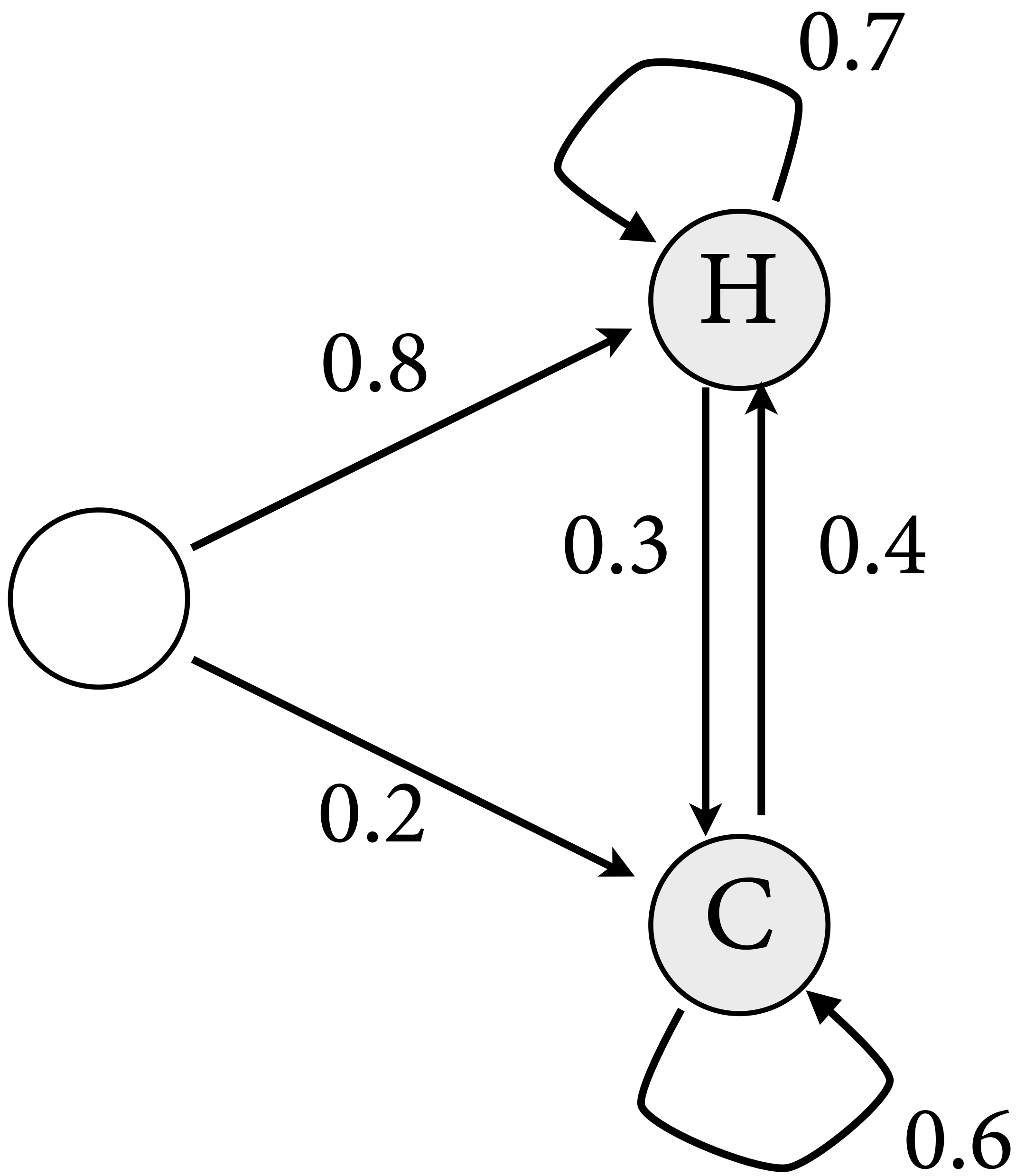
The representative put chairs on the table.
Secretariat is expected to race tomorrow.

→ next time

Hidden Markov Models

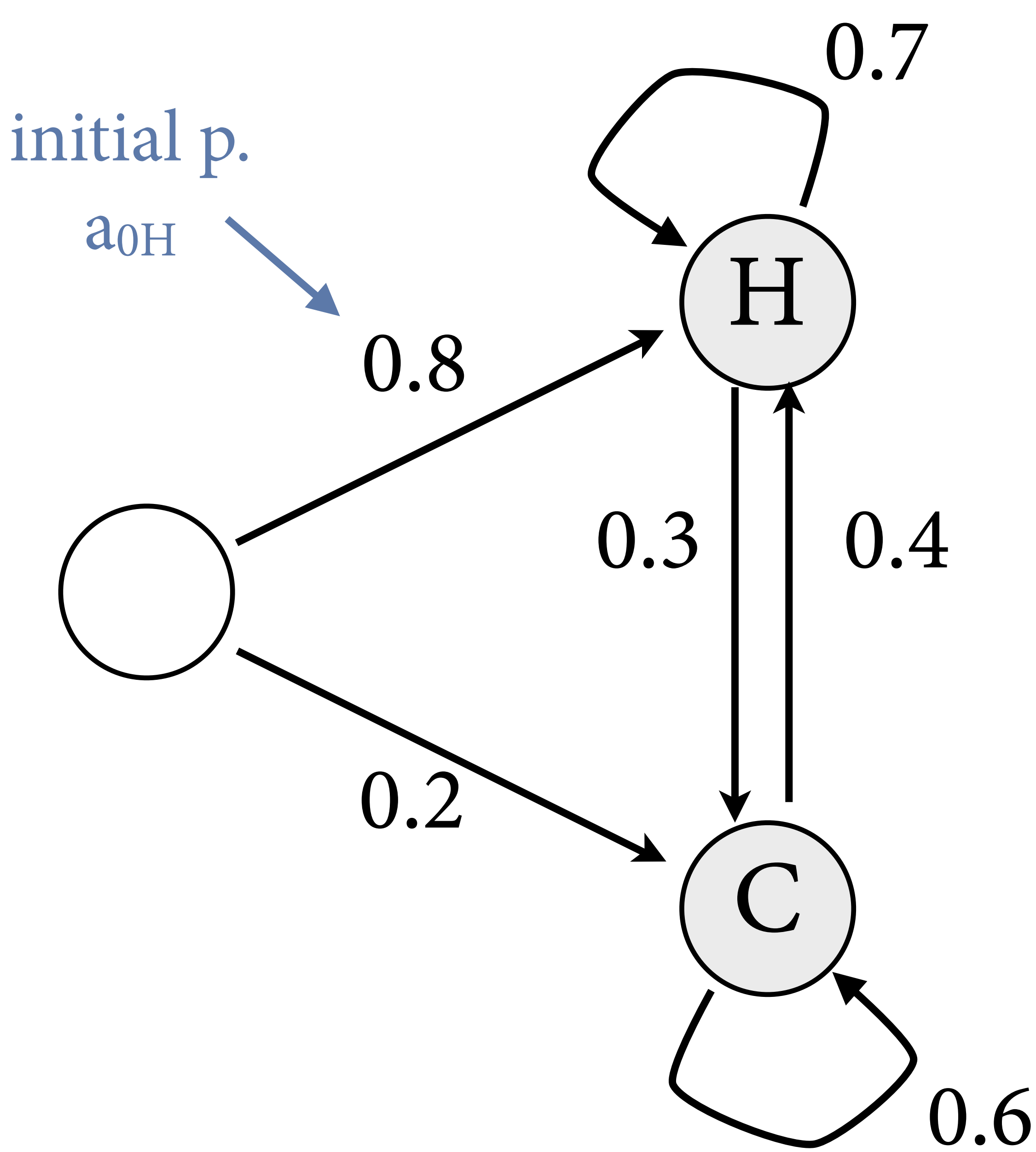
- A *Hidden Markov Model* is 5-tuple consisting of
  - finite set $Q = \{q_1, \ldots, q_N\}$ of *states* (= POS tags)
  - finite set O of possible *observations* (= words)
  - *transition probabilities* $a_{ij} = P(X_{t+1} = q_j \mid X_t = q_i)$ $\sum_{j=1}^{N} a_{ij} = 1$
  - *initial probabilities* $a_{0i} = P(X_1 = q_i)$ $\sum_{i=1}^{N} a_{0i} = 1$
  - *emission probabilities* $b_i(o) = P(Y_t = o \mid X_t = q_i)$ $\sum_{o \in O} b_i(o) = 1$
- The HMM describes two coupled random processes:
  - event $X_t = q_i$: At time t, HMM is in state q_i.
  - event $Y_t = o$: At time t, HMM emits observation o.

Example: Eisner's Ice Cream



States represent weather on a given day: Hot, Cold
Outputs represent number of ice creams Jason eats that day

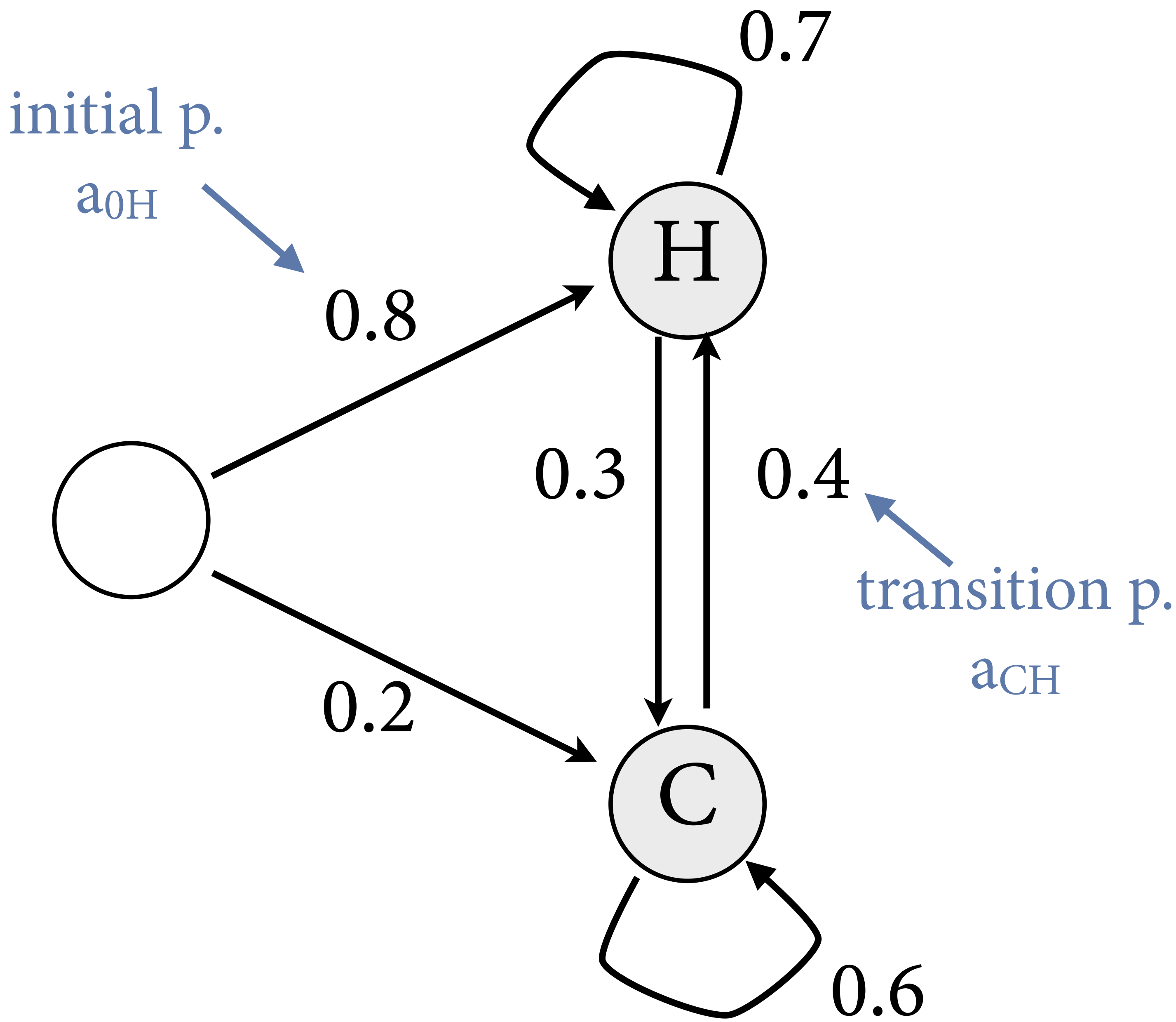
Example: Eisner's Ice Cream



States represent weather on a given day: Hot, Cold

Outputs represent number of ice creams Jason eats that day

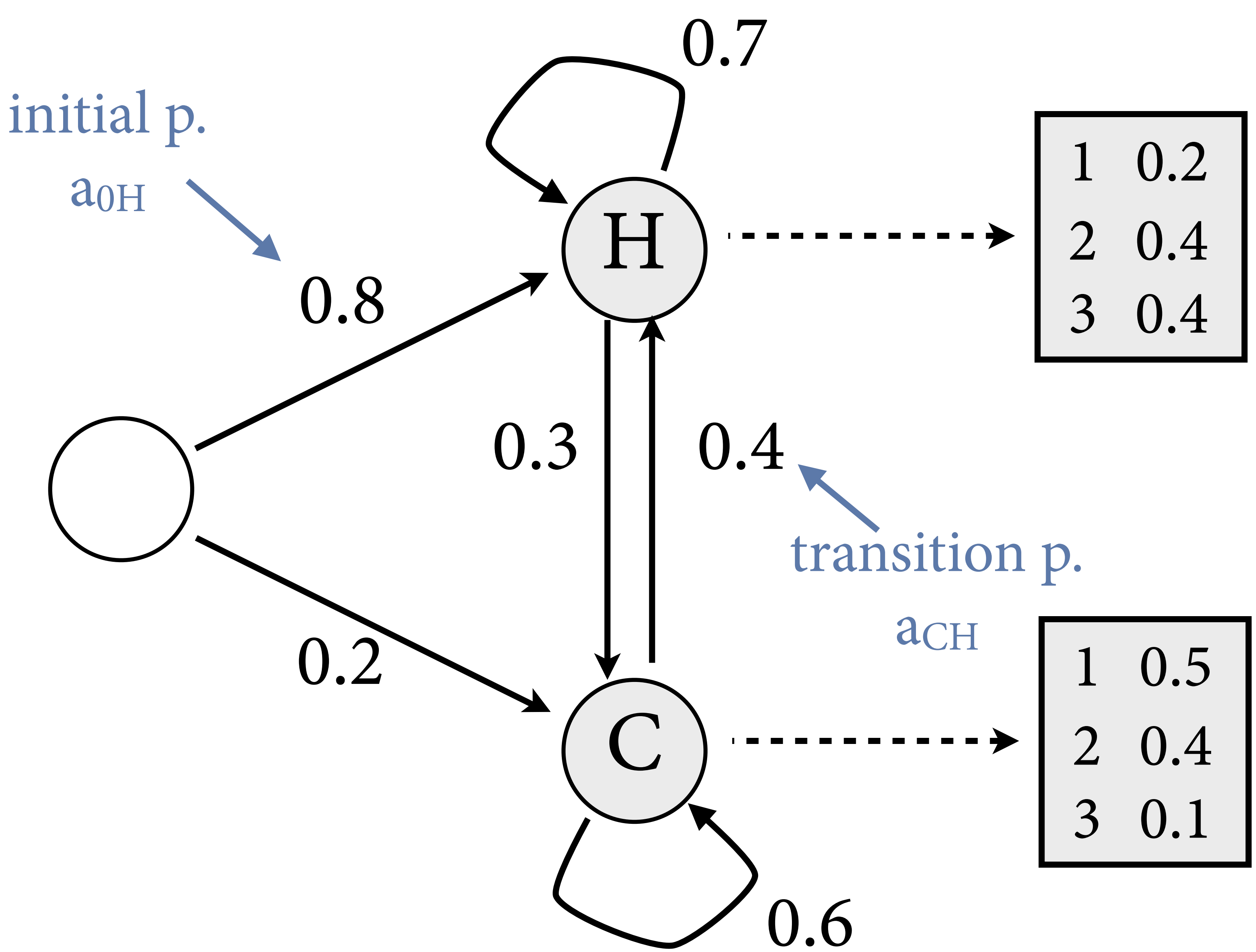
Example: Eisner's Ice Cream



States represent weather on a given day: Hot, Cold

Outputs represent number of ice creams Jason eats that day

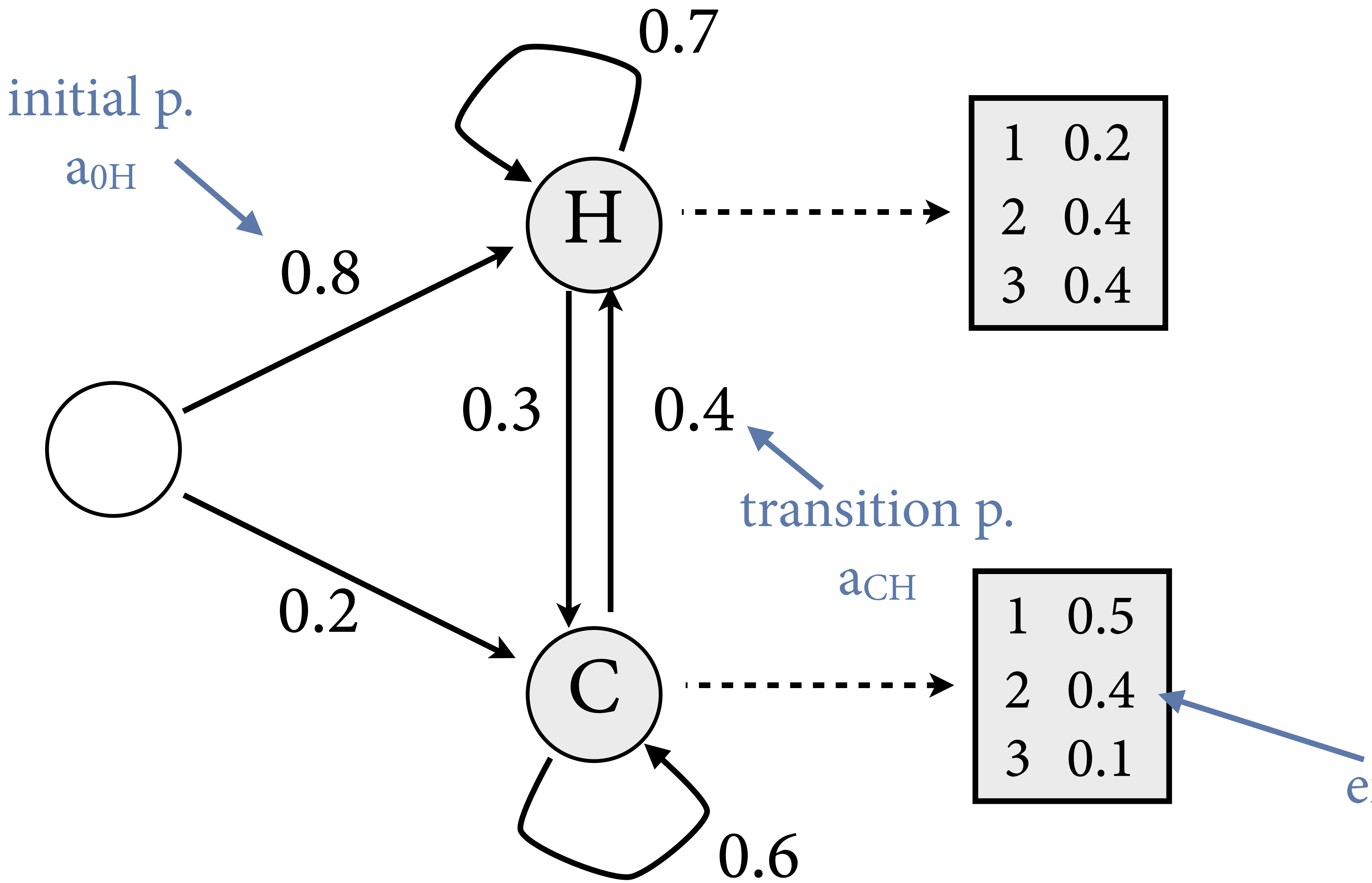
Example: Eisner's Ice Cream



States represent weather on a given day: Hot, Cold

Outputs represent number of ice creams Jason eats that day

Example: Eisner's Ice Cream



States represent weather on a given day: Hot, Cold

Outputs represent number of ice creams Jason eats that day

HMMs joint model of x, y

- Coupled random processes of HMM directly give us model for *joint* probability P(x, y) where
  - y = $y_1 \ldots y_T$ sequence of observations
  - x = $x_1 \ldots x_T$ sequence of hidden states

- Defined as follows:

$$
\begin{aligned}
P(x,y) &= P(x) \cdot P(y \mid x) \\
&= \prod_{t=1}^{T} P(X_t = x_t \mid X_{t-1} = x_{t-1}) \cdot \prod_{t=1}^{T} P(Y_t = y_t \mid X_t = y_t) \\
&= \prod_{t=1}^{T} a_{x_{t-1}x_t} \cdot \prod_{t=1}^{T} b_{x_t}(y_t)
\end{aligned}
$$

Question 1: Likelihood, P(y)

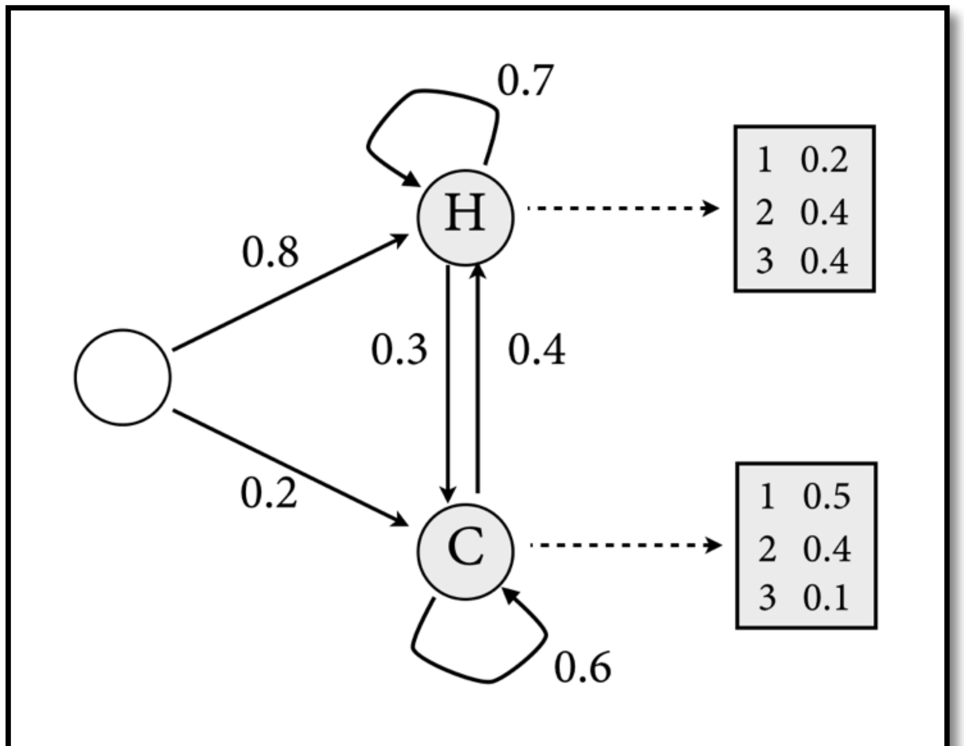
- How likely is it that Jason Eisner ate 3 ice creams on day 1, 1 ice cream on day 2, 3 ice creams on day 3?

- Want to compute: P(3, 1, 3).

- We can easily compute things like P(H, 3, C, 1, H, 3).
  - But 3, 1, 3 can be emitted by many different state sequences.
  - Need to sum over all of those.

Naive approach

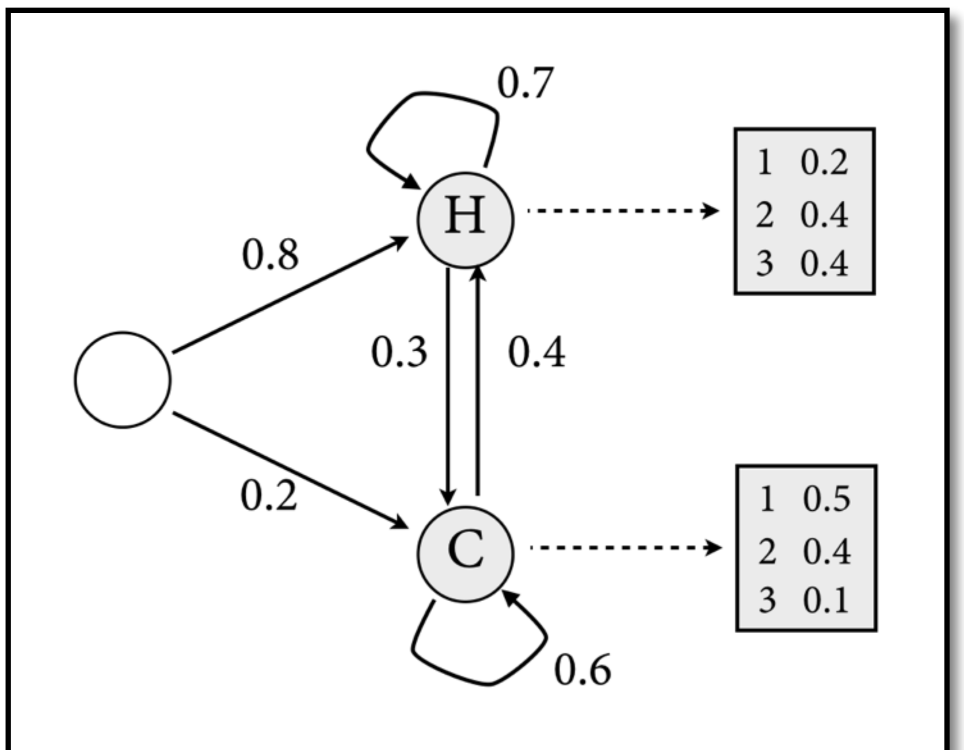
- Compute P(3,1,3) by summing over all possible state sequences.



P(3,1,3)	
= P(H,3,H,1,H,3)	0.013
+ P(H,3,H,1,C,3)	0.001
+ P(H,3,C,1,H,3)	0.008
+ ...	. . .
+ P(C,3,C,1,C,3)	0.0004
=	0.026

Naive approach

- Compute P(3,1,3) by summing over all possible state sequences.



P(3,1,3)	
= P(H,3,H,1,H,3)	0.013
+ P(H,3,H,1,C,3)	0.001
+ P(H,3,C,1,H,3)	0.008
+ ...	. . .
+ P(C,3,C,1,C,3)	0.0004
=	0.026

- Technical term for "summing up": *marginalization*.

$$P(3,1,3) = \sum_{x_1,x_2,x_3 \in Q} P(x_1,3,x_2,1,x_3,3)$$

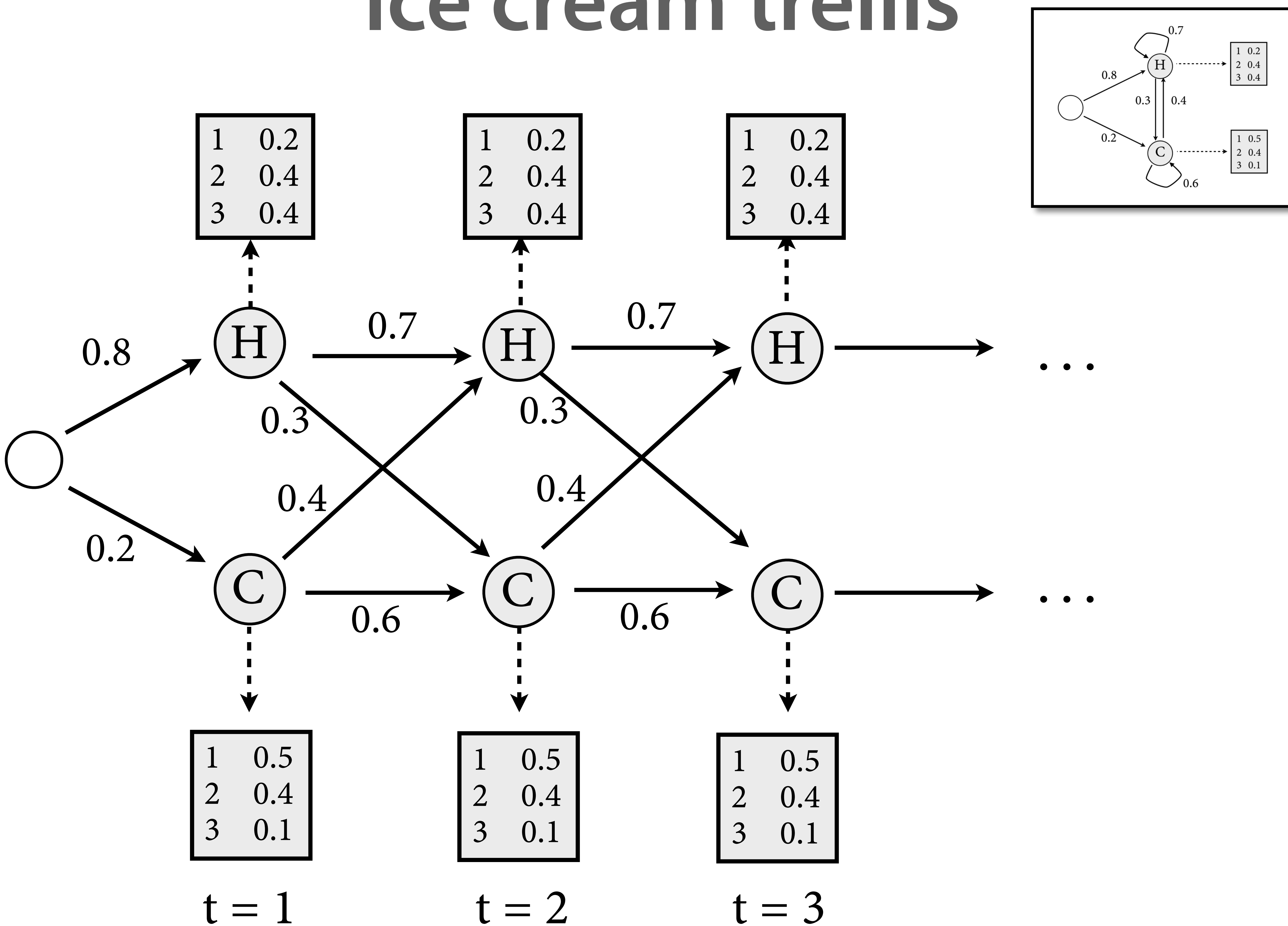
Too expensive

- Naive approach sums over exponential set of terms. This is too slow for practical use.

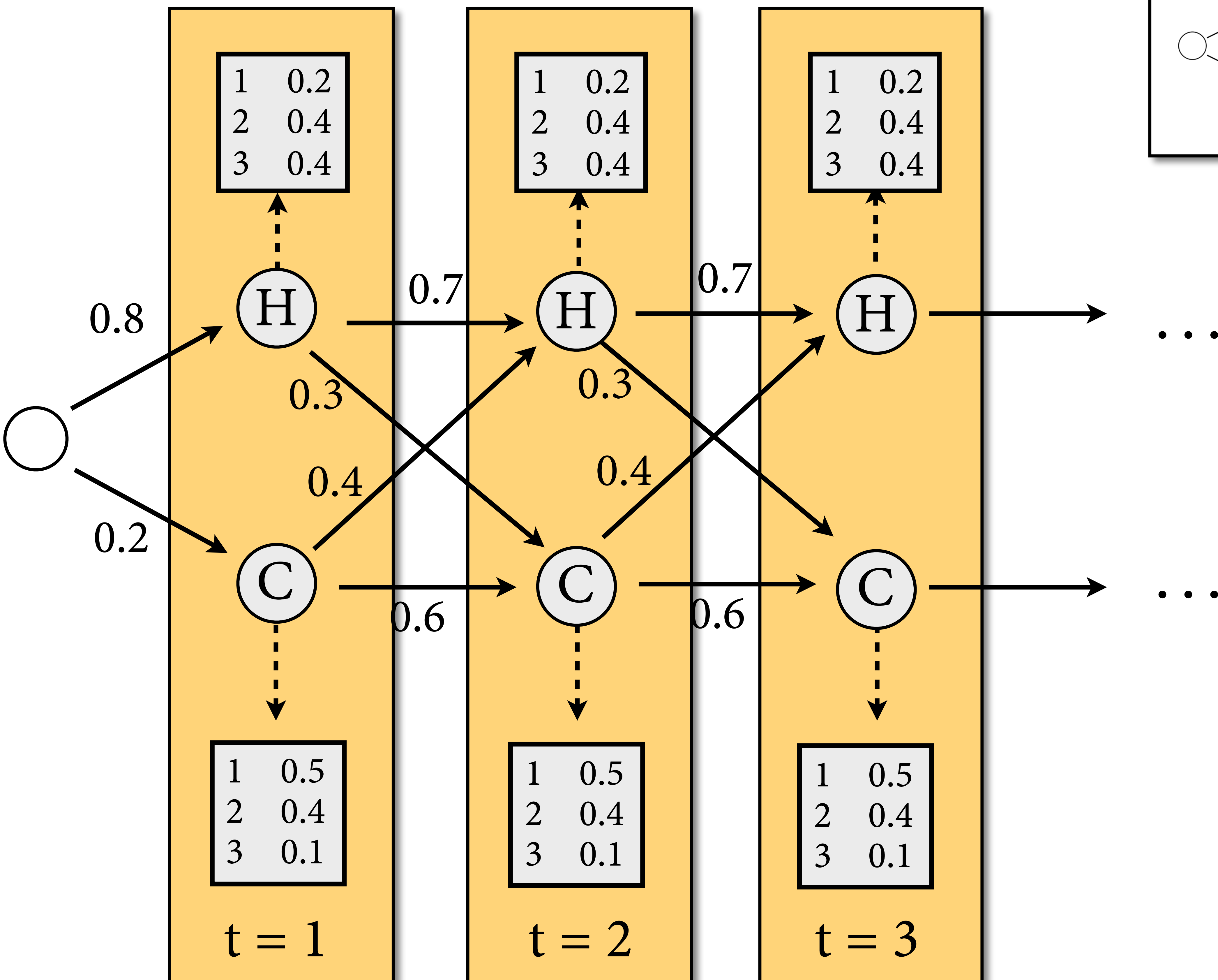
- Visualize this in *trellis:* unfolding of HMM
  - one column for each time point t, represents X_t
  - each column contains a copy of each state of HMM
  - edges from t to t+1 = transitions of HMM

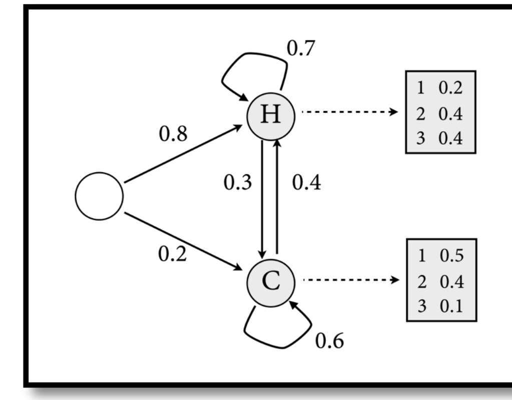
- Each path through trellis represents one state sequence.
  - So computation of P(w) = sum over all paths.

Ice cream trellis

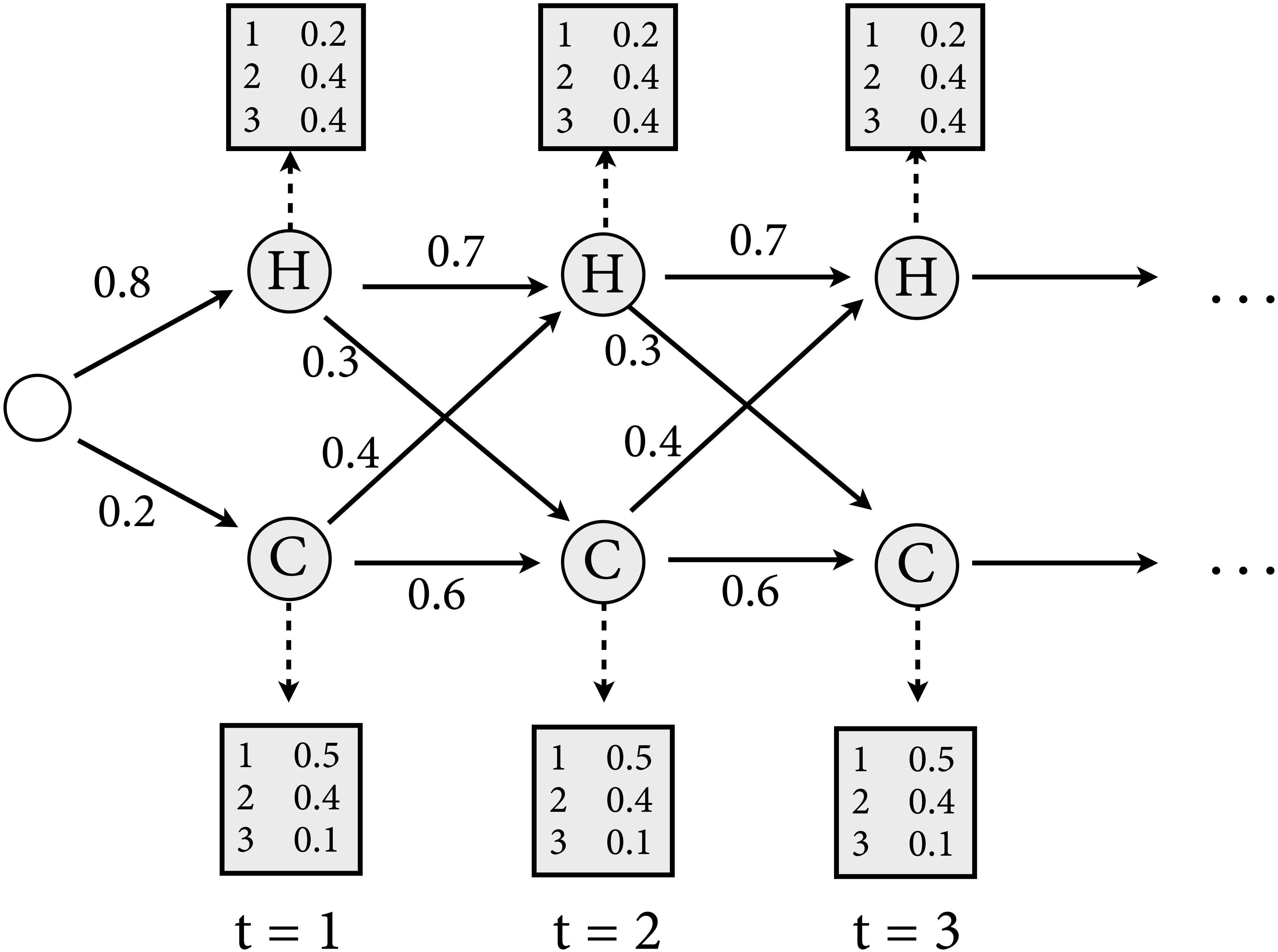


Ice cream trellis

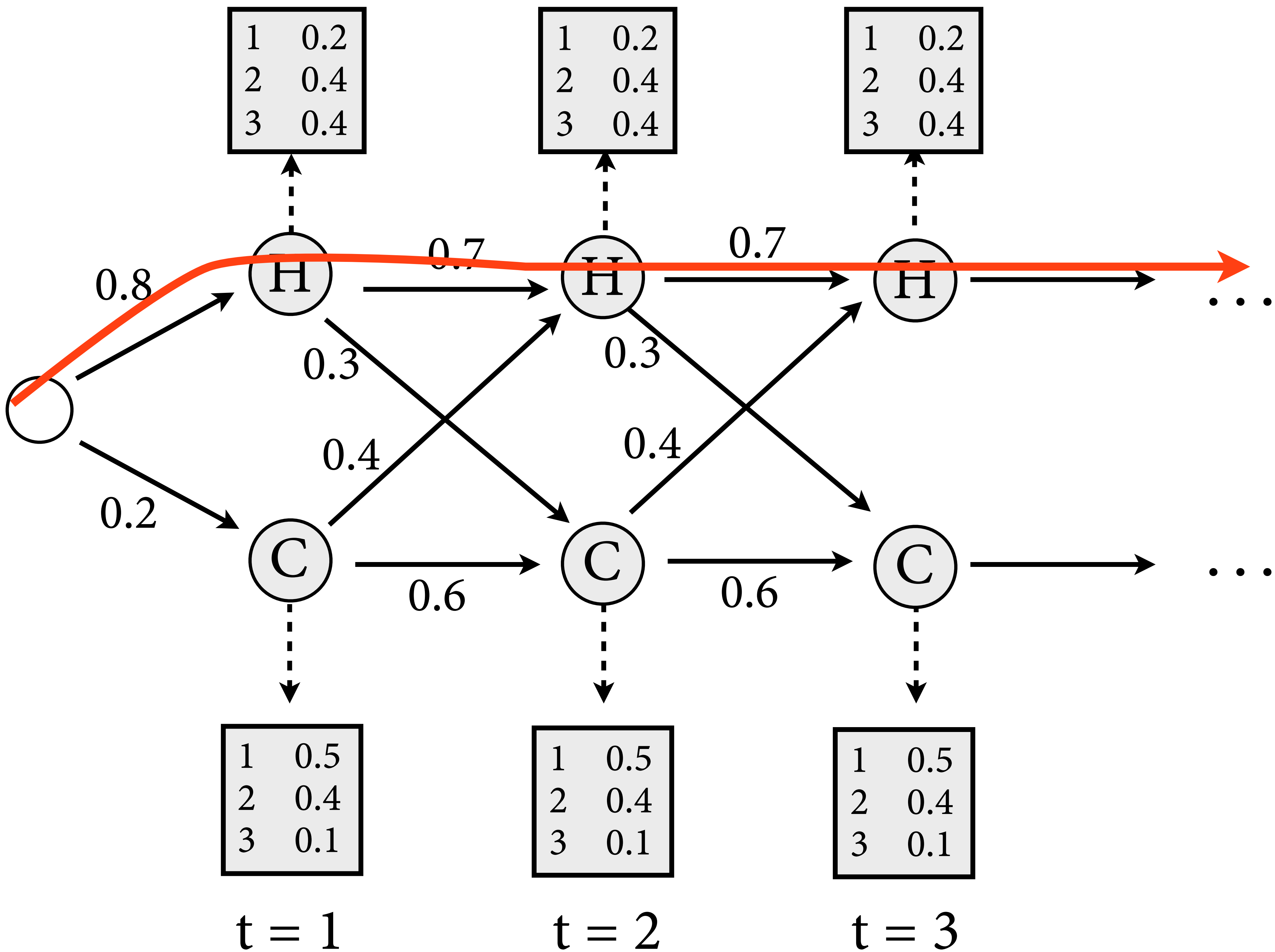




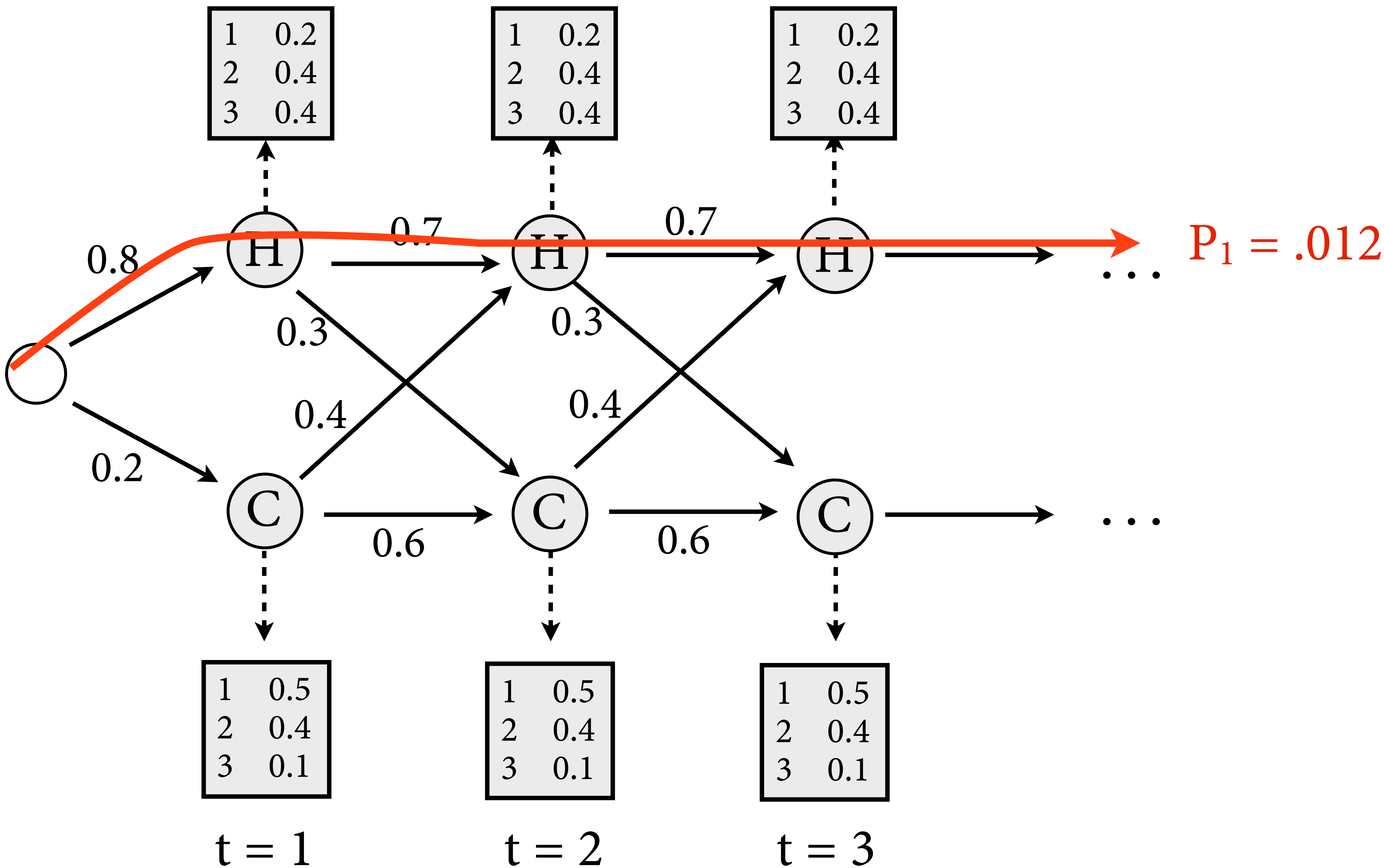
Ice cream trellis



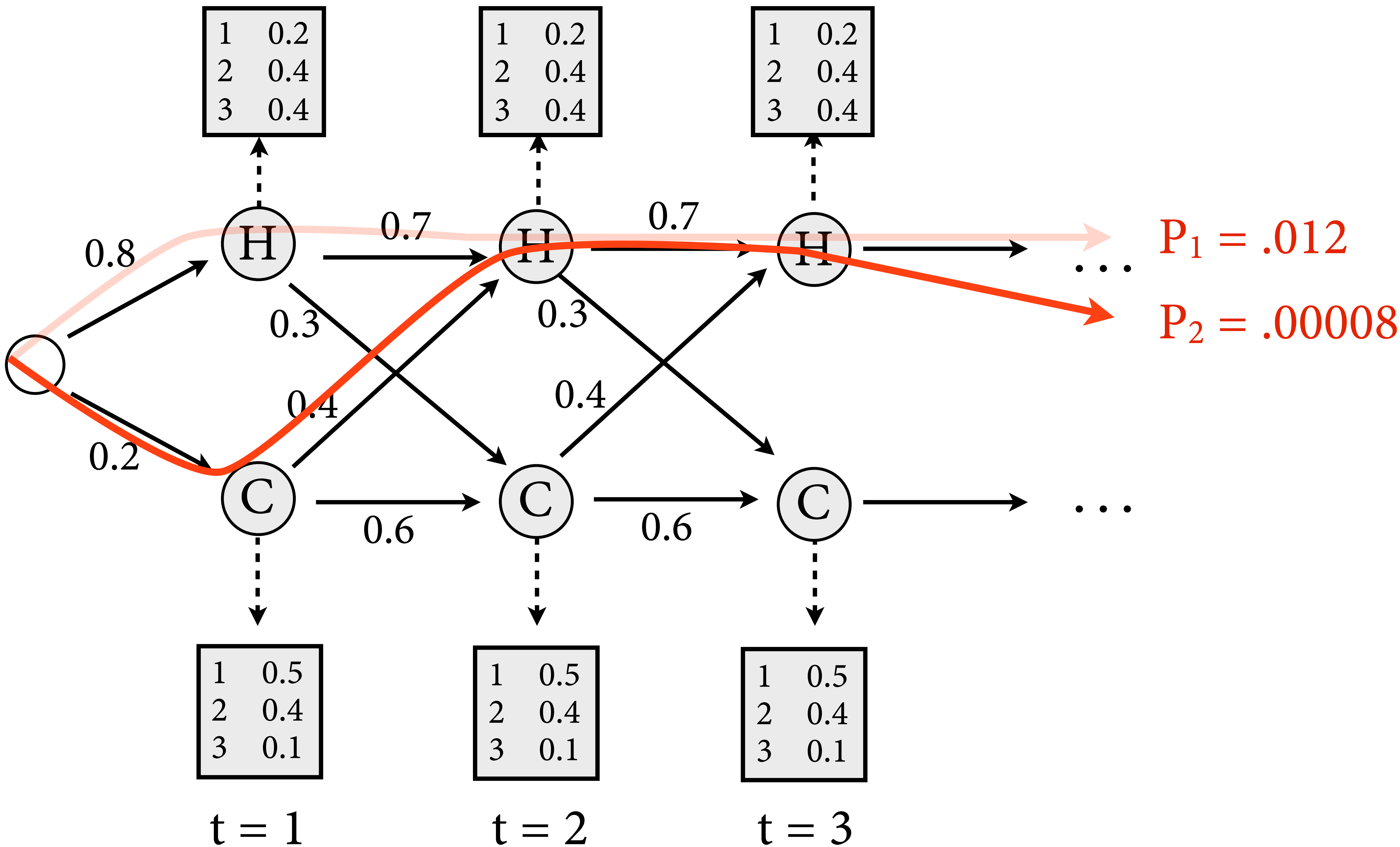
Ice cream trellis



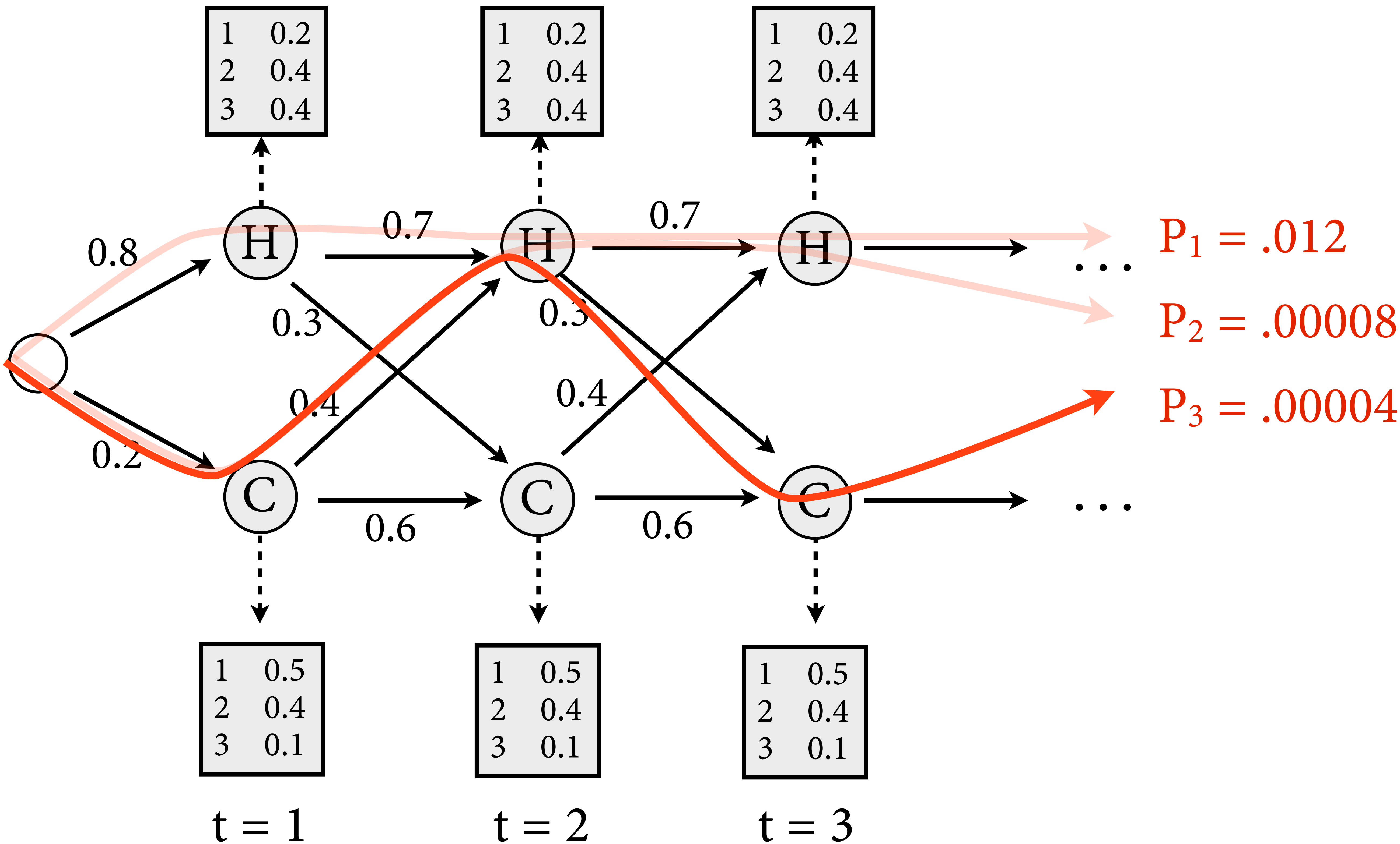
Ice cream trellis



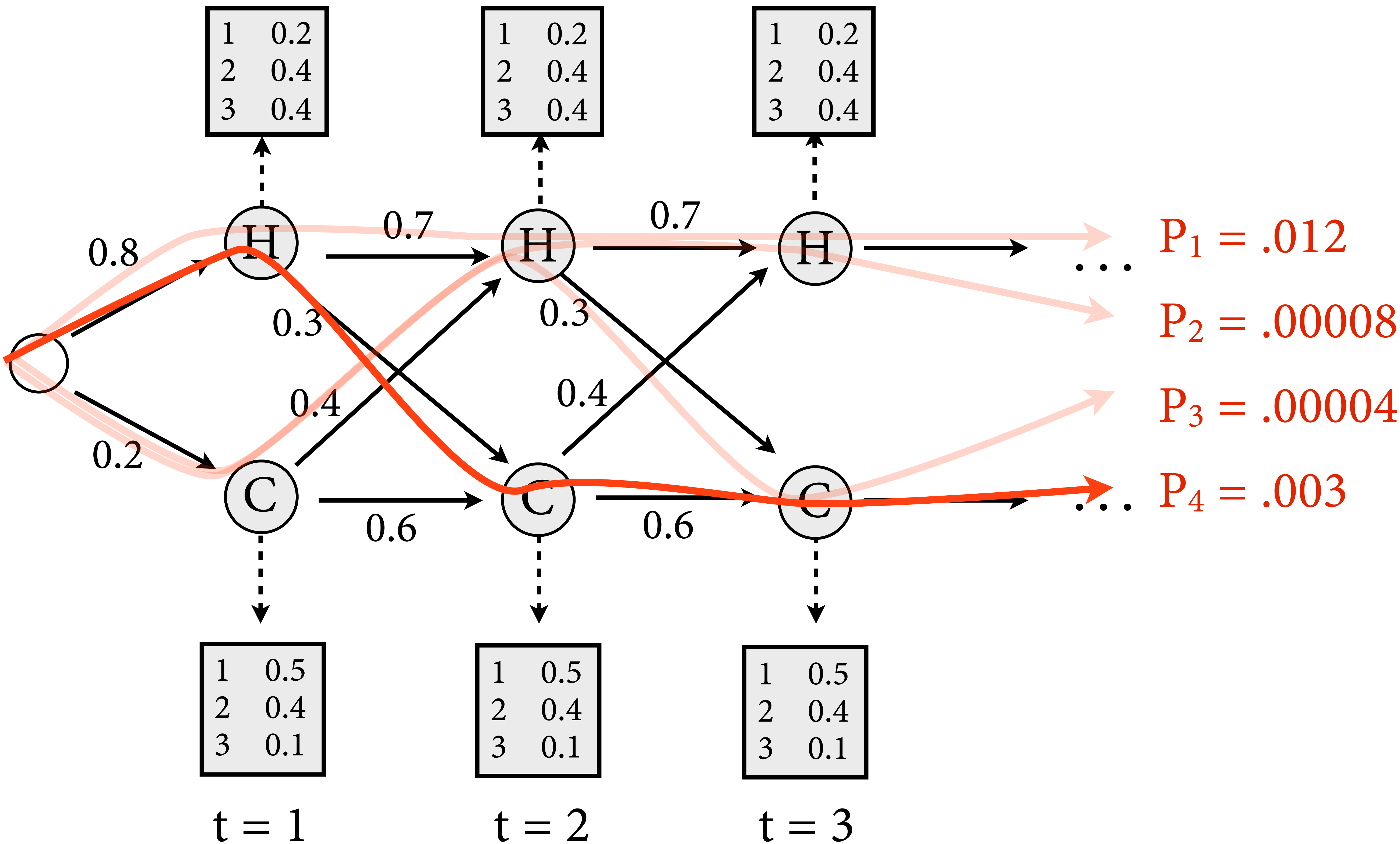
Ice cream trellis



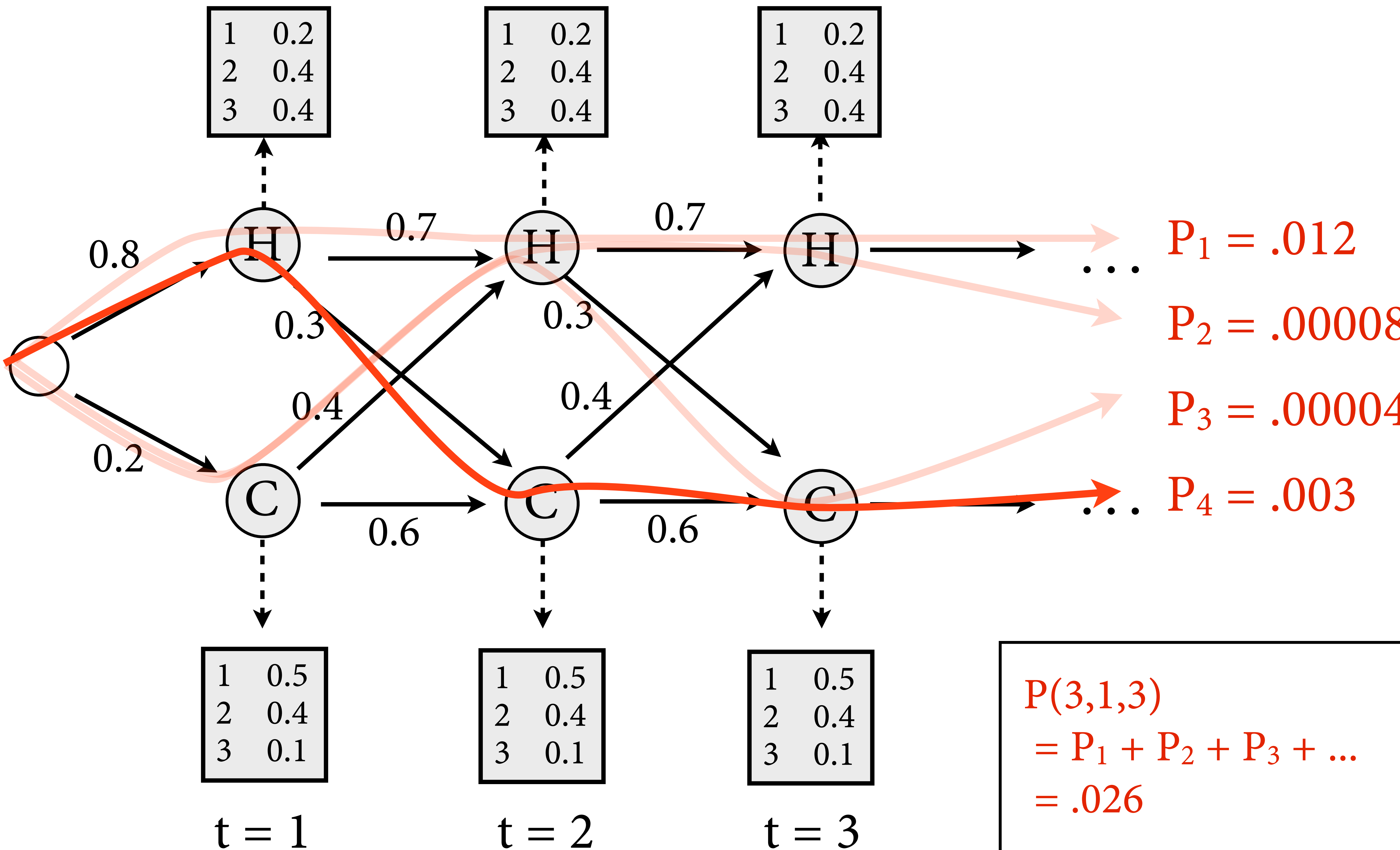
Ice cream trellis



Ice cream trellis



Ice cream trellis



$P(3,1,3)$
$= P_1 + P_2 + P_3 + \ldots$
$= .026$

The Forward Algorithm

- Naive algorithm computes intermediate results many times. Computation can be done faster.
- Key idea: *Forward probability* $\alpha_t(j)$ that HMM outputs $y_1, \ldots, y_t$ and then ends in $X_t = q_j$.

$$\begin{aligned} \alpha_t(j) &= P(y_1, \ldots, y_t, X_t = q_j) \\ &= \sum_{x_1, \ldots, x_{t-1}} P(y_1, \ldots, y_t, X_1 = x_1, \ldots, X_{t-1} = x_{t-1}, X_t = q_j) \end{aligned}$$

- From this, can compute easily

$$P(y_1, \ldots, y_T) = \sum_{q \in Q} \alpha_T(q)$$

The Forward Algorithm

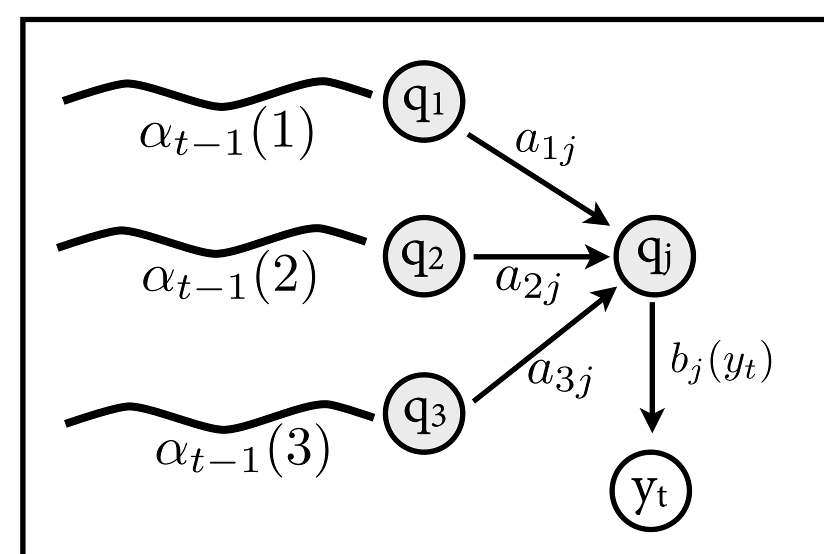
$$\alpha_t(j) = P(y_1, \dots, y_t, X_t = q_j)$$

- Base case, t = 1:

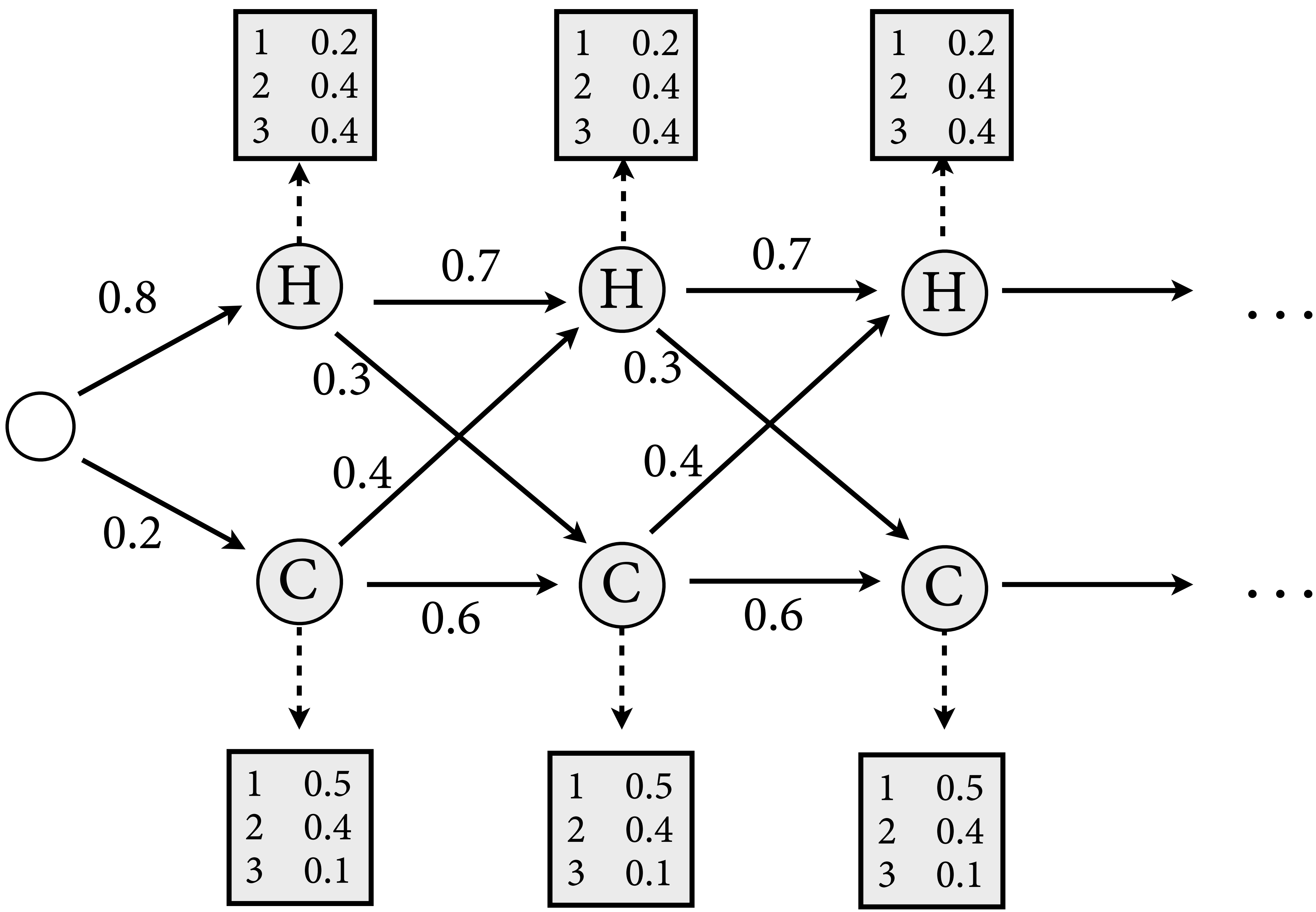
$$\alpha_1(j) = P(y_1, X_1 = q_j) = b_j(y_1) \cdot a_{0j}$$

- Inductive case, compute for t = 2, …, T:

$$\begin{aligned}\alpha_t(j) &= P(y_1, \dots, y_t, X_t = q_j) \\ &= \sum_{i=1}^{N} P(y_1, \dots, y_{t-1}, X_{t-1} = q_i) \cdot P(X_t = q_j \mid X_{t-1} = q_i) \cdot P(y_t \mid X_t = q_j) \\ &= \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)\end{aligned}$$



P(3,1,3) with Forward

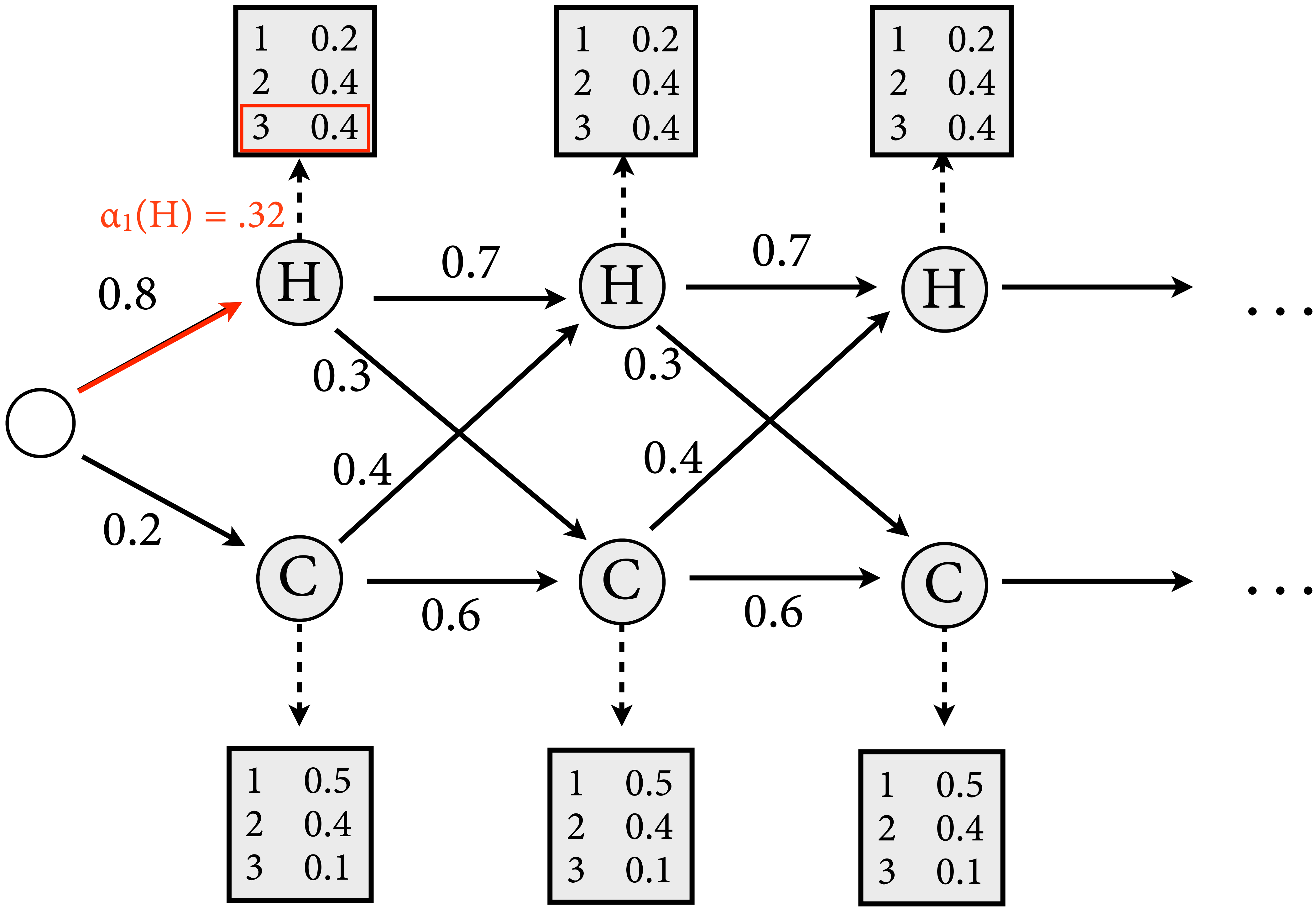


$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$

$\alpha_1(j) = b_j(y_1) \cdot a_{0j}$

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

P(3,1,3) with Forward

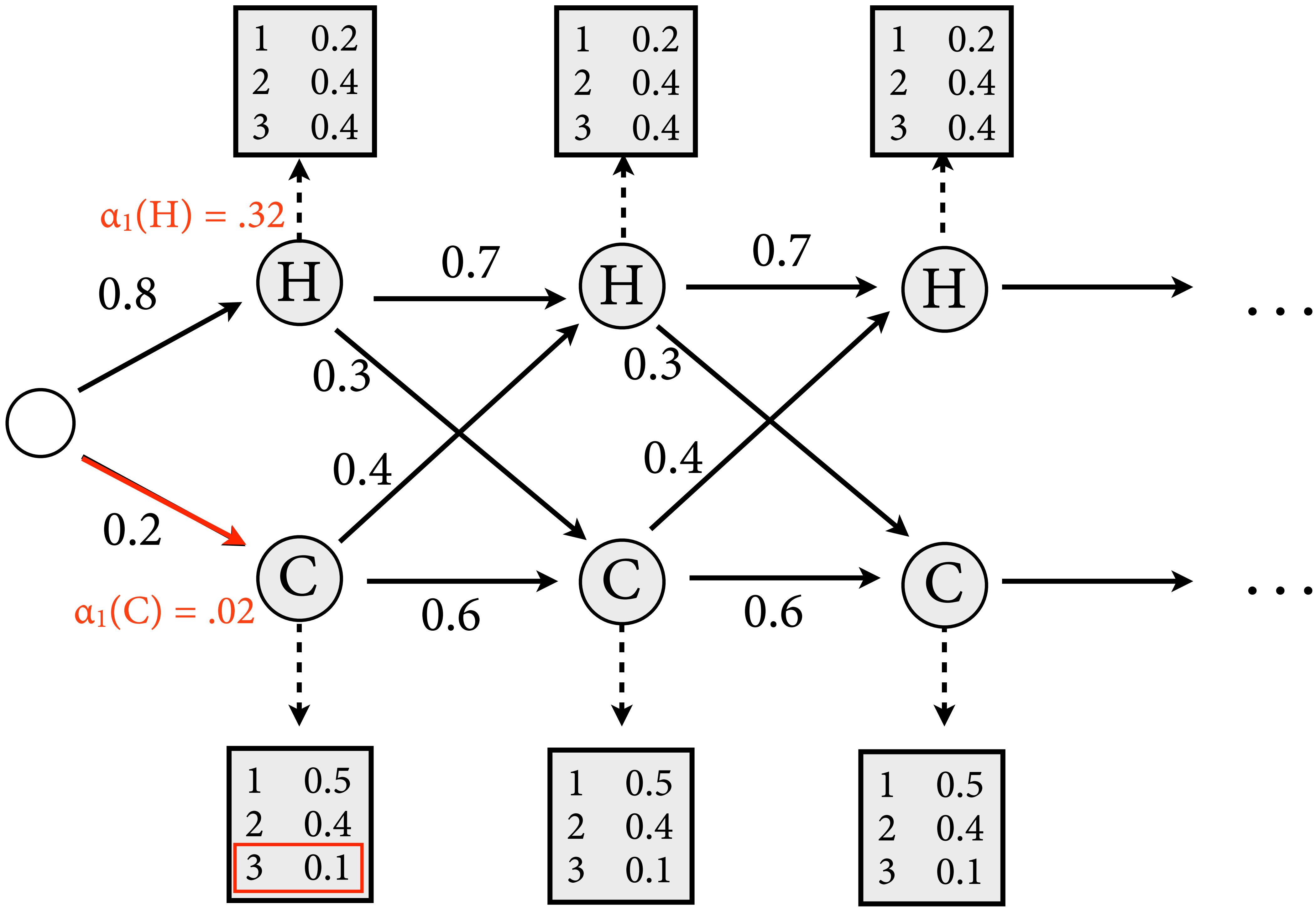


$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$

$\alpha_1(j) = b_j(y_1) \cdot a_{0j}$

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

P(3,1,3) with Forward

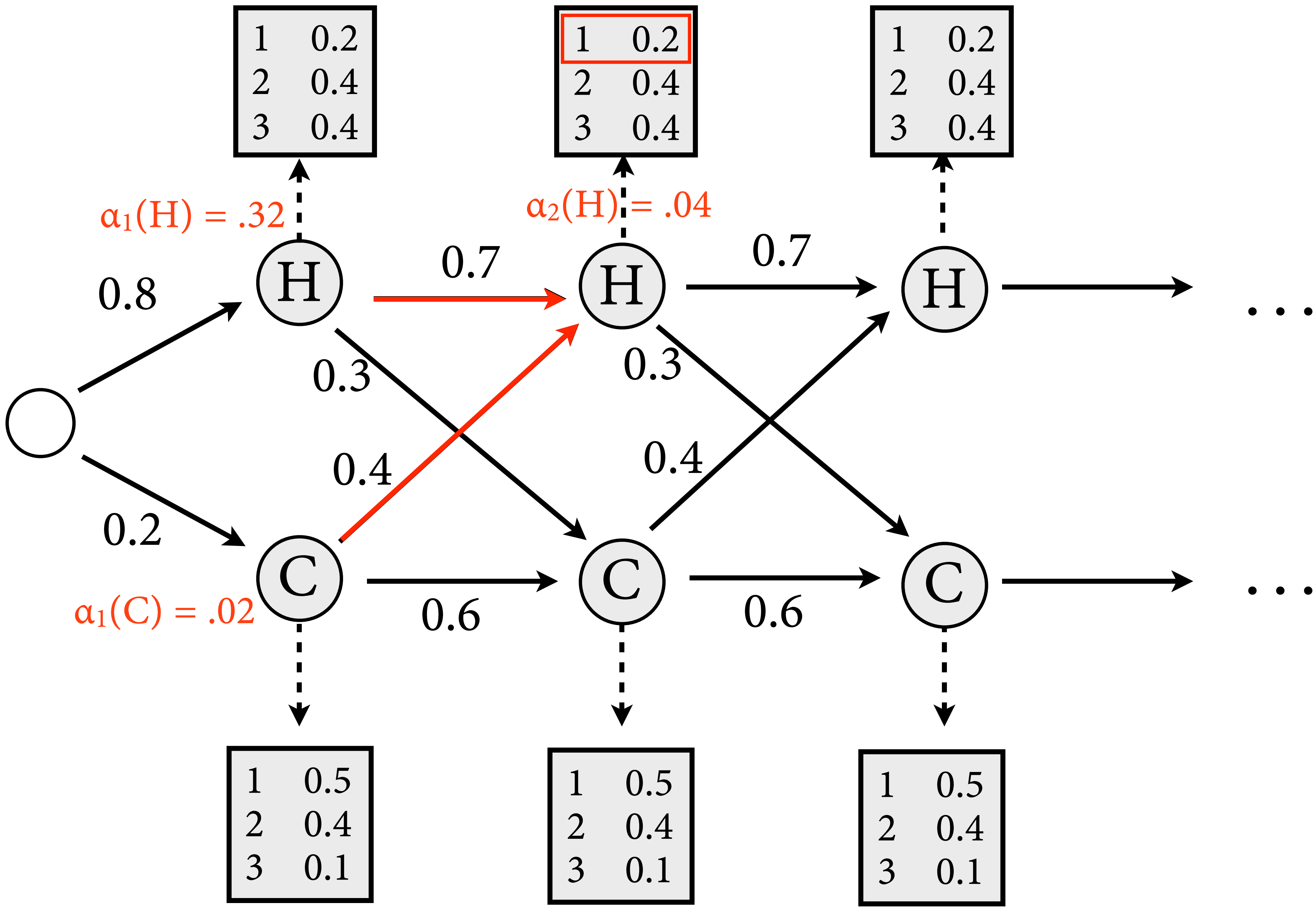


$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$

$\alpha_1(j) = b_j(y_1) \cdot a_{0j}$

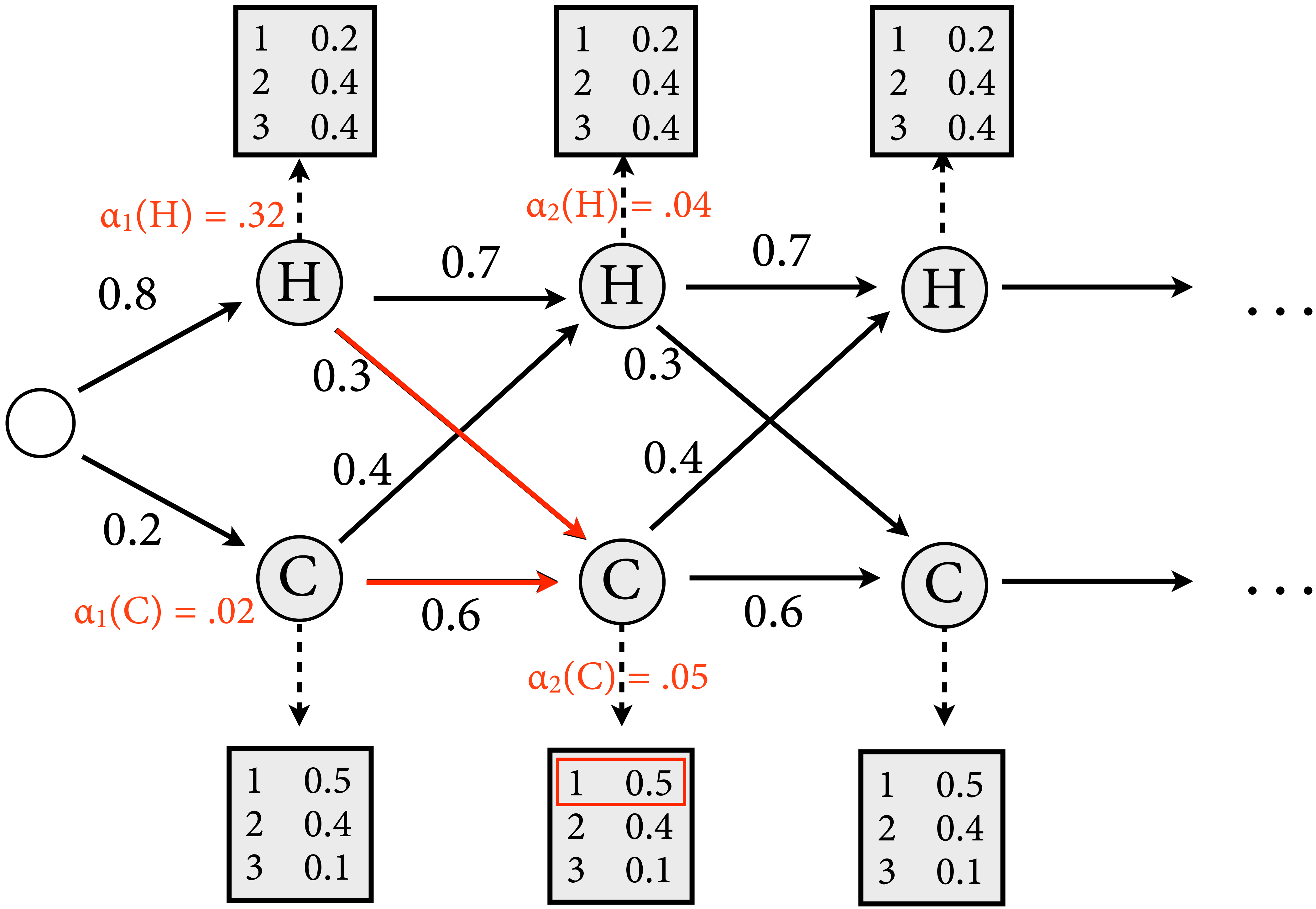
$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$

P(3,1,3) with Forward



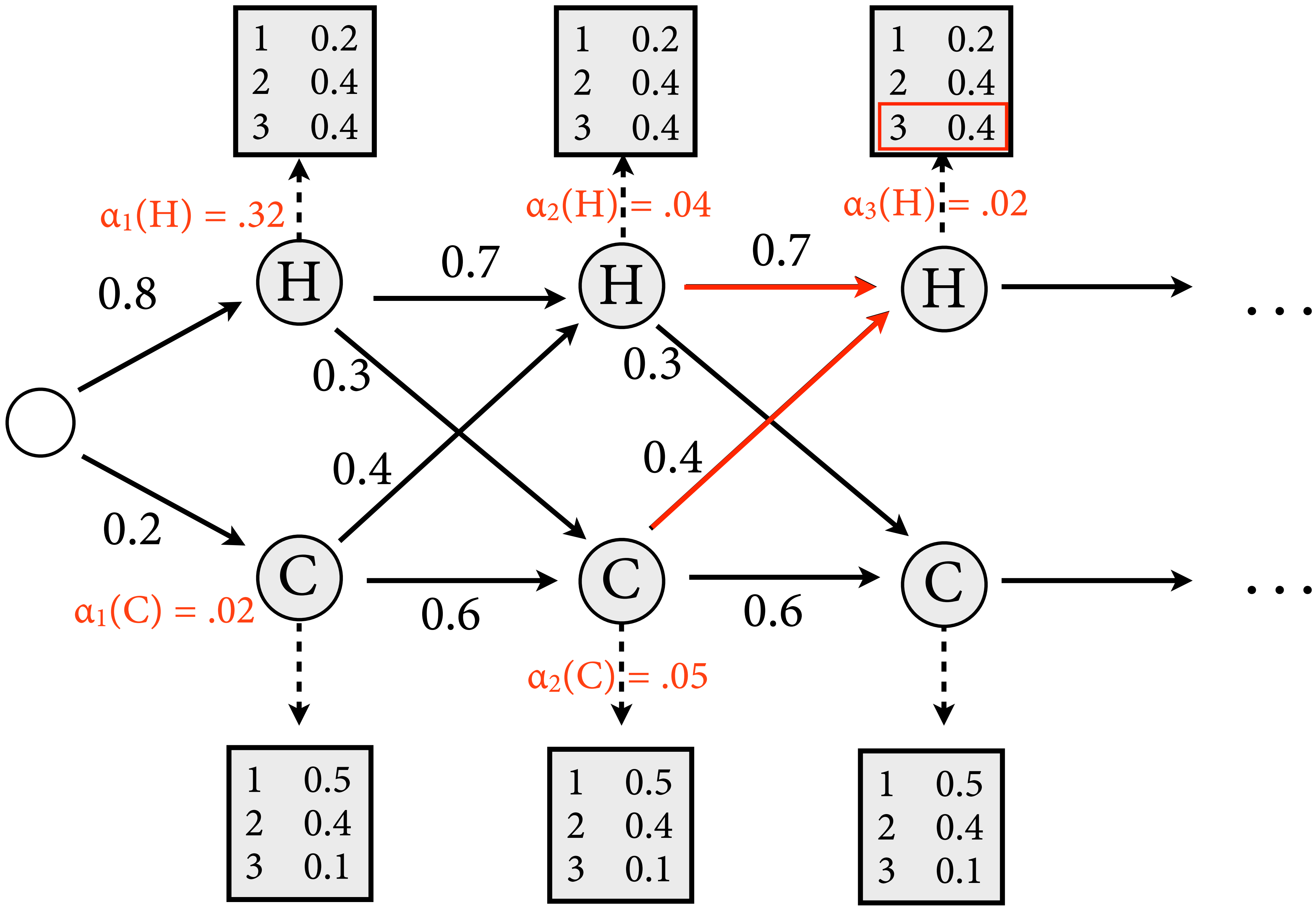
$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$ $\alpha_1(j) = b_j(y_1) \cdot a_{0j}$ $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$

P(3,1,3) with Forward



$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$ $\alpha_1(j) = b_j(y_1) \cdot a_{0j}$ $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$

P(3,1,3) with Forward

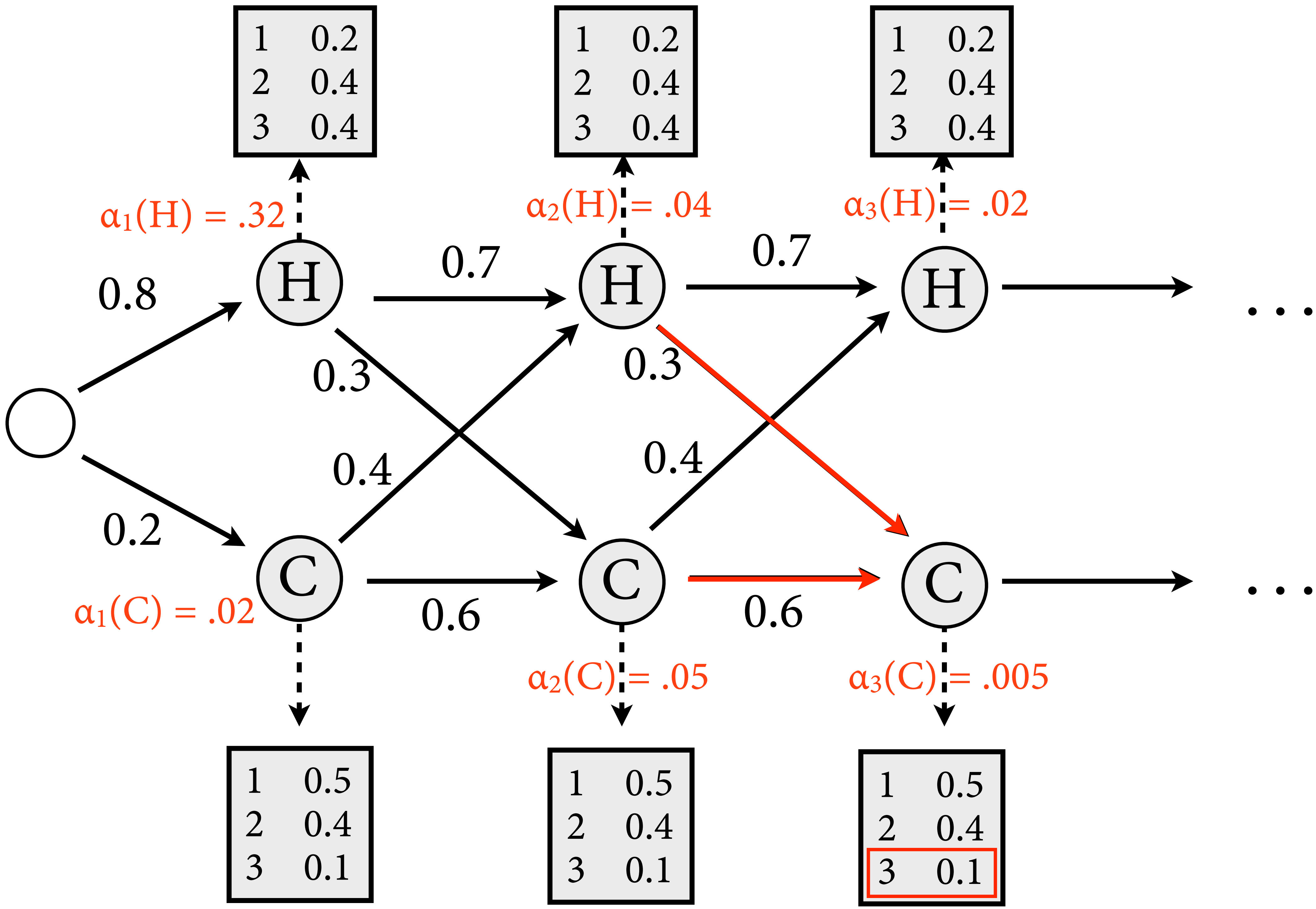


$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$

$\alpha_1(j) = b_j(y_1) \cdot a_{0j}$

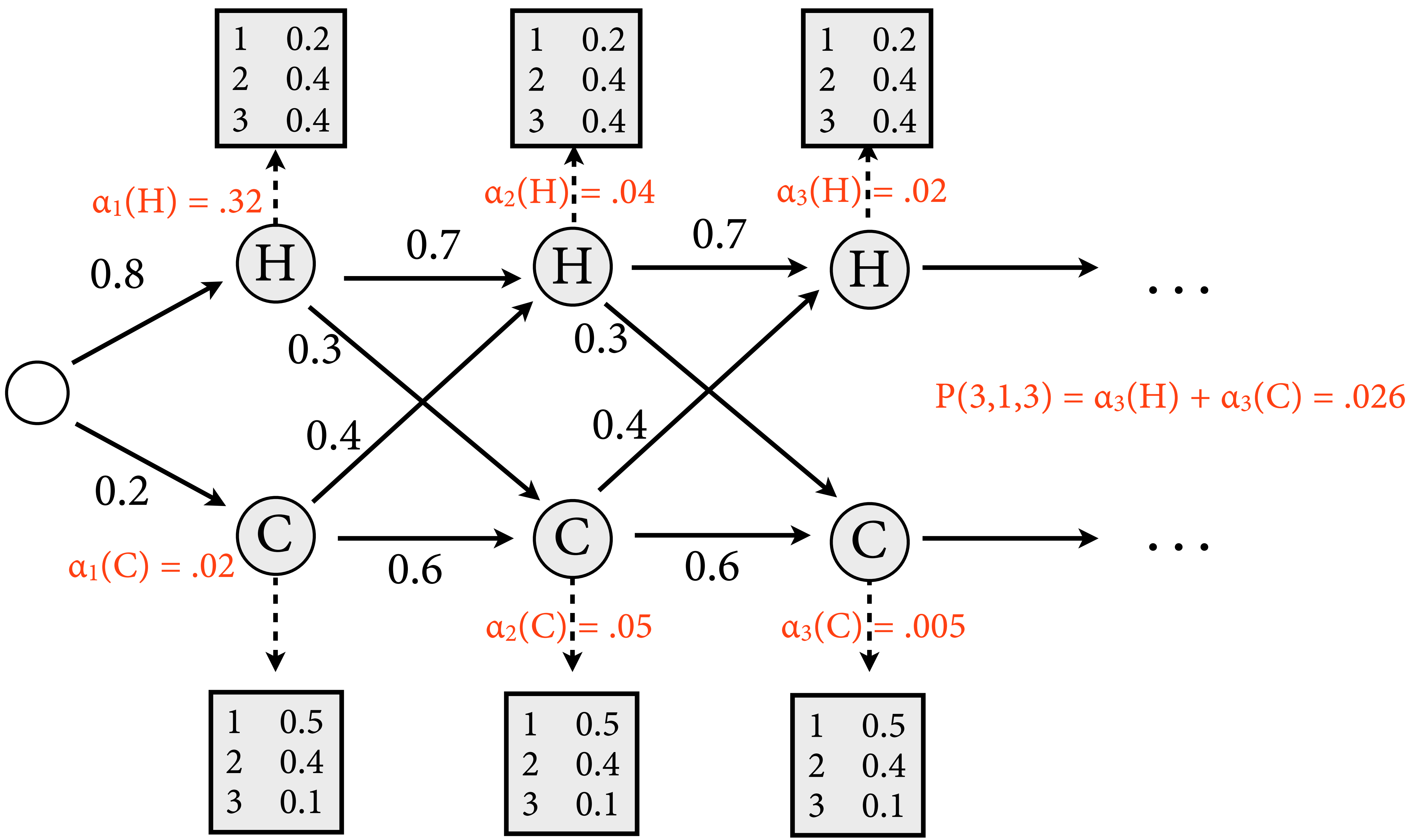
$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$

P(3,1,3) with Forward



$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$ $\alpha_1(j) = b_j(y_1) \cdot a_{0j}$ $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$

P(3,1,3) with Forward



$$\alpha_t(j) = P(y_1, \ldots, y_t, X_t = q_j)$$

$$\alpha_1(j) = b_j(y_1) \cdot a_{0j}$$

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

Question 2: Tagging

- Given observations $y_1, \ldots, y_T$, what is the most probable sequence $x_1, \ldots, x_T$ of hidden states?

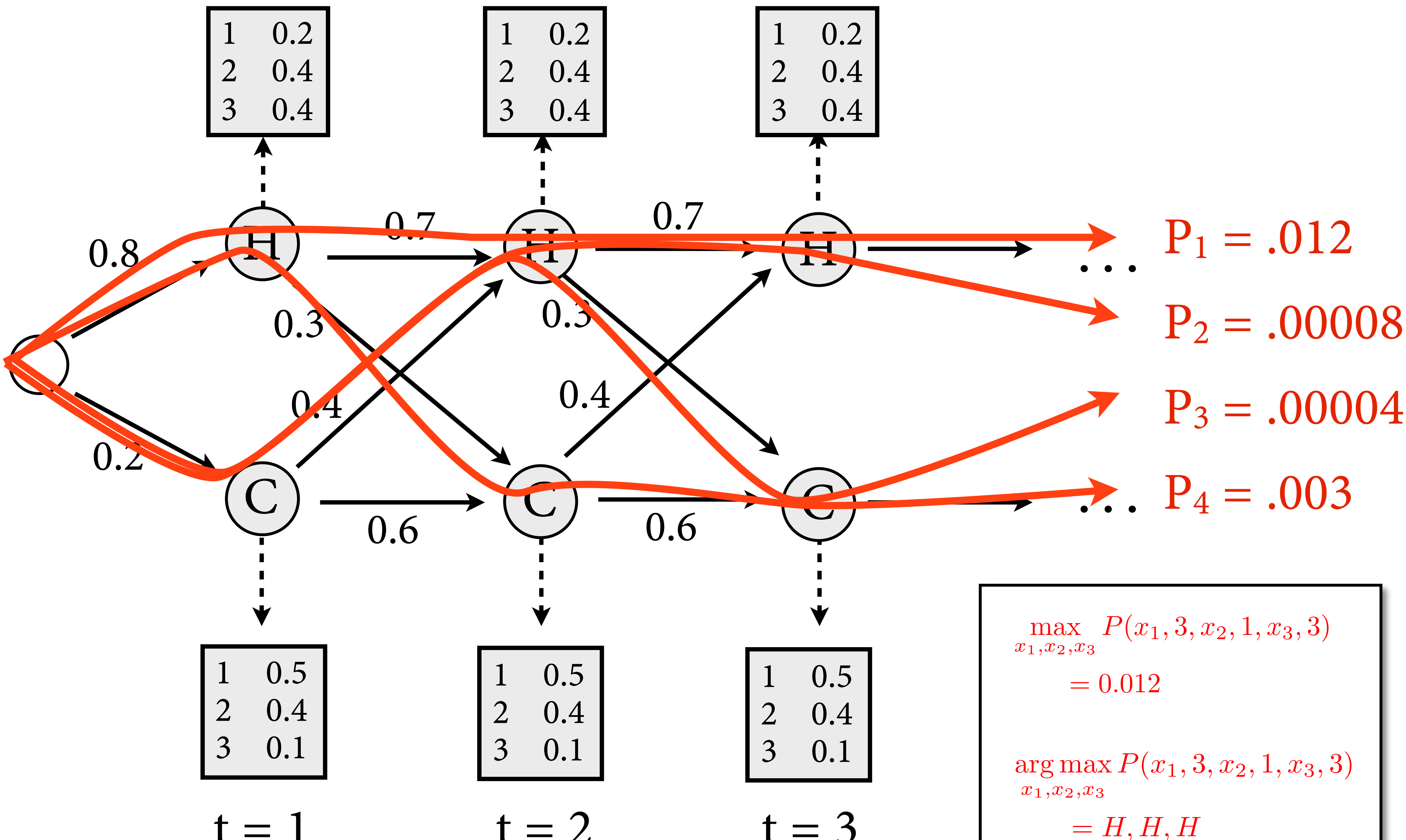
- Maximum probability:

$$\max_{x_1,\ldots,x_T} P(x_1,\ldots,x_T \mid y_1,\ldots,y_T)$$

- We are primarily interested in arg max:

$$\begin{aligned}
&\arg\max_{x_1,\ldots,x_T} P(x_1,\ldots,x_T \mid y_1,\ldots,y_T) \\
&= \arg\max_{x_1,\ldots,x_T} \frac{P(x_1,\ldots,x_T,y_1,\ldots,y_T)}{P(y_1,\ldots,y_T)} \\
&= \arg\max_{x_1,\ldots,x_T} P(x_1,\ldots,x_T,y_1,\ldots,y_T)
\end{aligned}$$

Naive solution



$$\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = 0.012$$

$$\arg\max_{x_1,x_2,x_3} P(x_1, 3, x_2, 1, x_3, 3) = H, H, H$$

Parallel to Likelihood

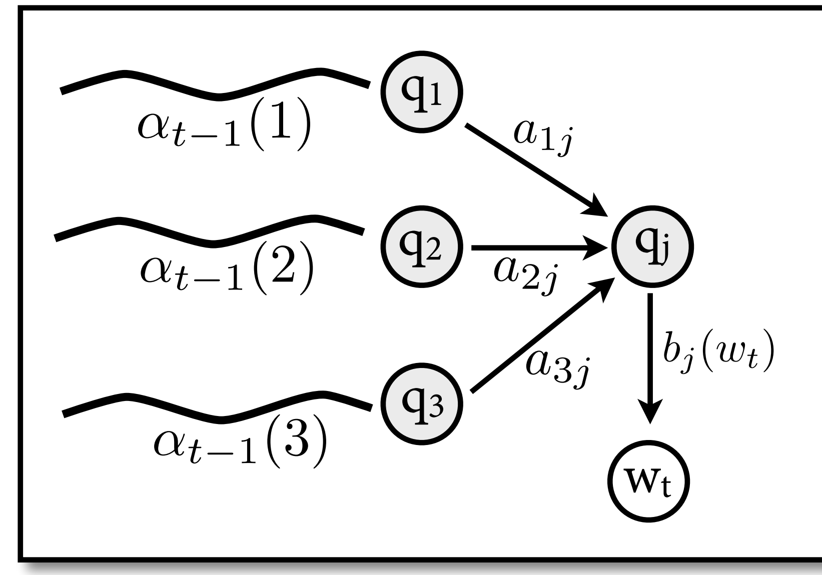
Likelihood: P(y) Forward Algorithm	Tagging: argmax P(x,y) Viterbi Algorithm
$\sum_{x_1,\ldots,x_T} P(x_1,\ldots,x_T,y_1,\ldots,y_T)$	$\arg\max_{x_1,\ldots,x_T} P(x_1,\ldots,x_T,y_1,\ldots,y_T)$
$\alpha_t(j) = \sum_{x_1,\ldots,x_{t-1}} P(y_1,\ldots,y_t,x_1,\ldots,x_{t-1},X_t=q_j)$	$V_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1,\ldots,y_t,x_1,\ldots,x_{t-1},X_t=q_j)$
$P(y) = \sum_{q\in Q} \alpha_T(q)$	$\max_x P(x,y) = \max_{q\in Q} V_T(q)$

The Viterbi Algorithm

$$V_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1,\ldots,y_t,x_1,\ldots,x_{t-1},X_t=q_j)$$

- Base case, t = 1:

$$V_1(j) = b_j(y_1) \cdot a_{0j}$$

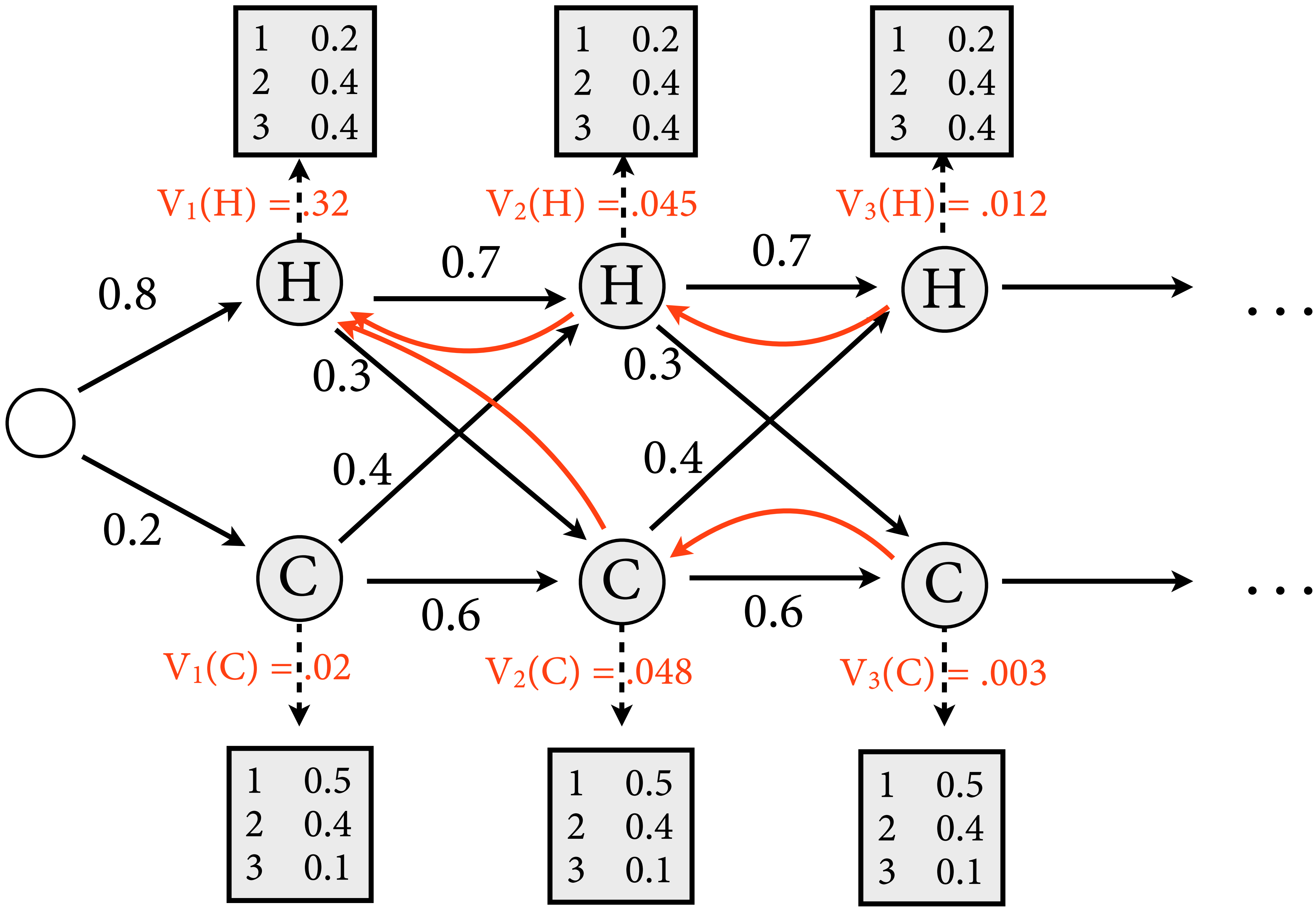


- Inductive case, for t = 2, …, T:

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

- For each (t,j), remember the i for which maximum was achieved in *backpointer* $bp_t(j)$.
  Use bp to compute best tag sequence from T backwards to 1.

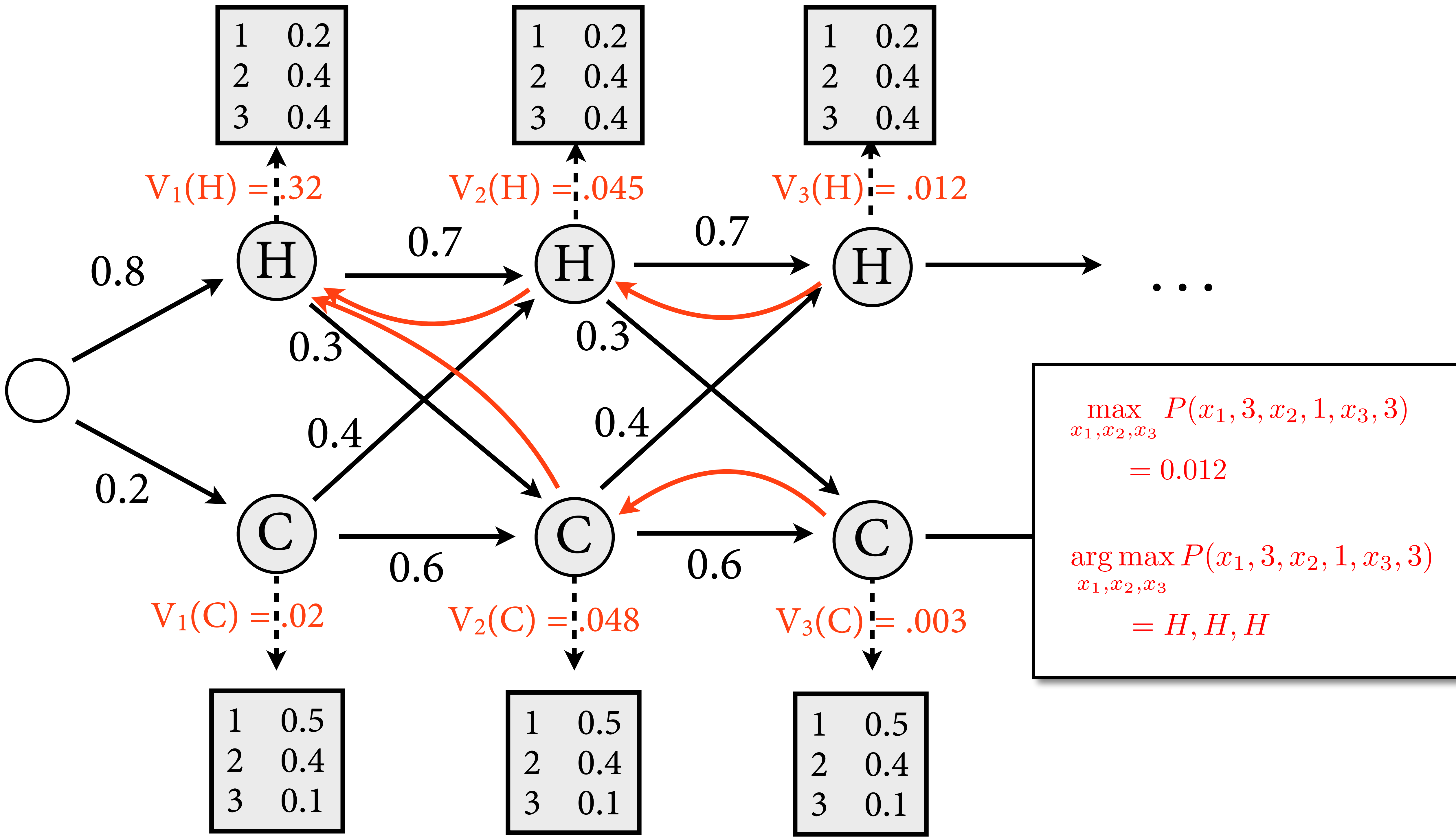
Viterbi Algorithm: Example



$$V_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1,\ldots,y_t,x_1,\ldots,x_{t-1},X_t = q_j)$$

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

Viterbi Algorithm: Example



$$V_t(j) = \max_{x_1,\ldots,x_{t-1}} P(y_1, \ldots, y_t, x_1, \ldots, x_{t-1}, X_t = q_j)$$

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

Runtime

- Forward and Viterbi have the same runtime, dominated by inductive step:

$$V_t(j) = \max_{i=1}^{N} V_{t-1}(i) \cdot a_{ij} \cdot b_j(y_t)$$

- Compute $N \cdot T$ values for $V_t(j)$. Each computation step requires iteration over N predecessor states.

- Total runtime is $O(N^2 \cdot T)$, i.e.
  - linear in sentence length
  - quadratic in size of tag set

Summary

- Hidden Markov Models popular model for POS tagging (and other applications, see later).
- Two coupled random processes:
  - bigram model for hidden states
  - model for producing observable output from state
- Efficient algorithms for common problems:
  - Likelihood computation: Forward algorithm
  - Best state sequence: Viterbi algorithm.