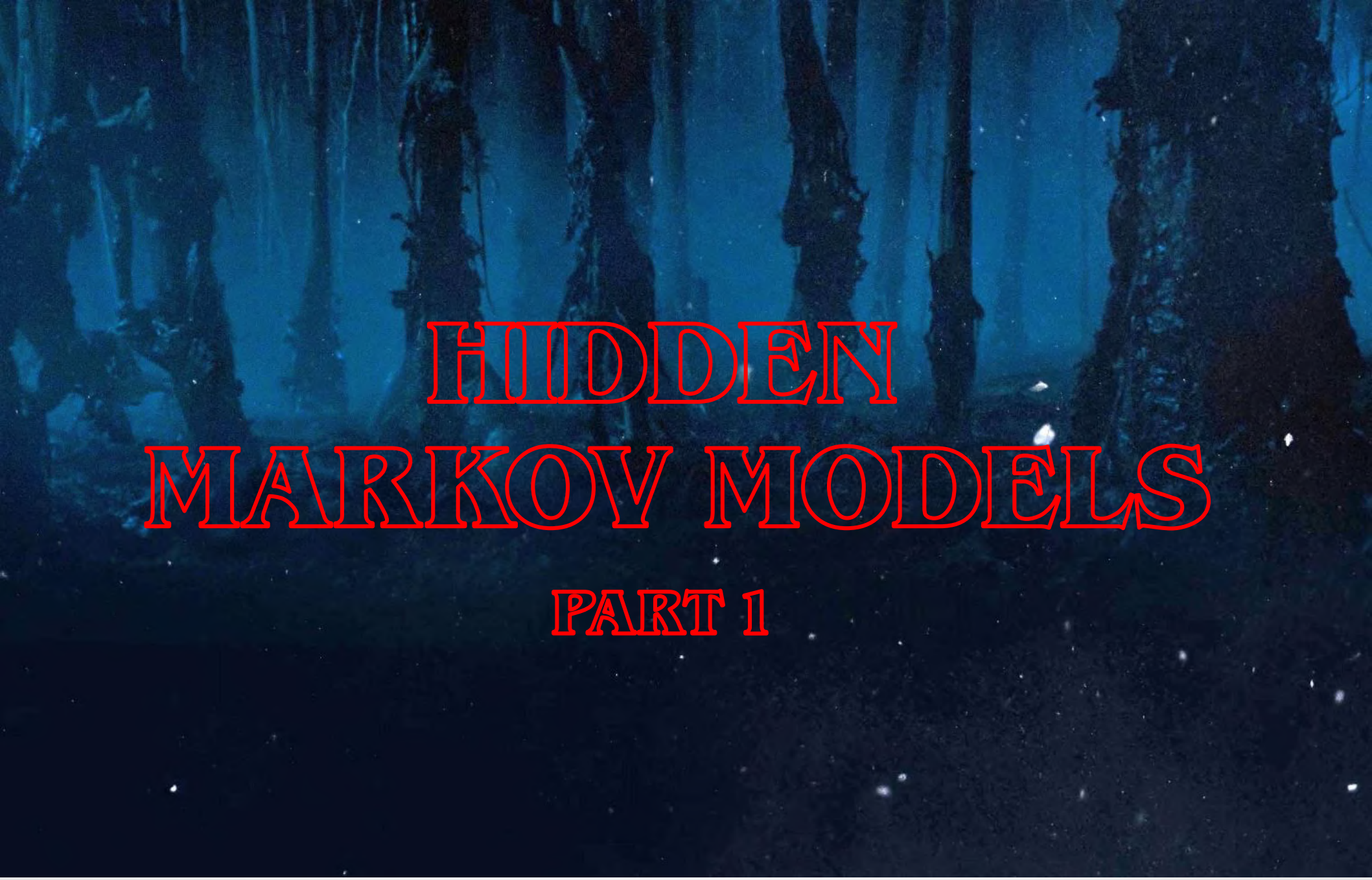
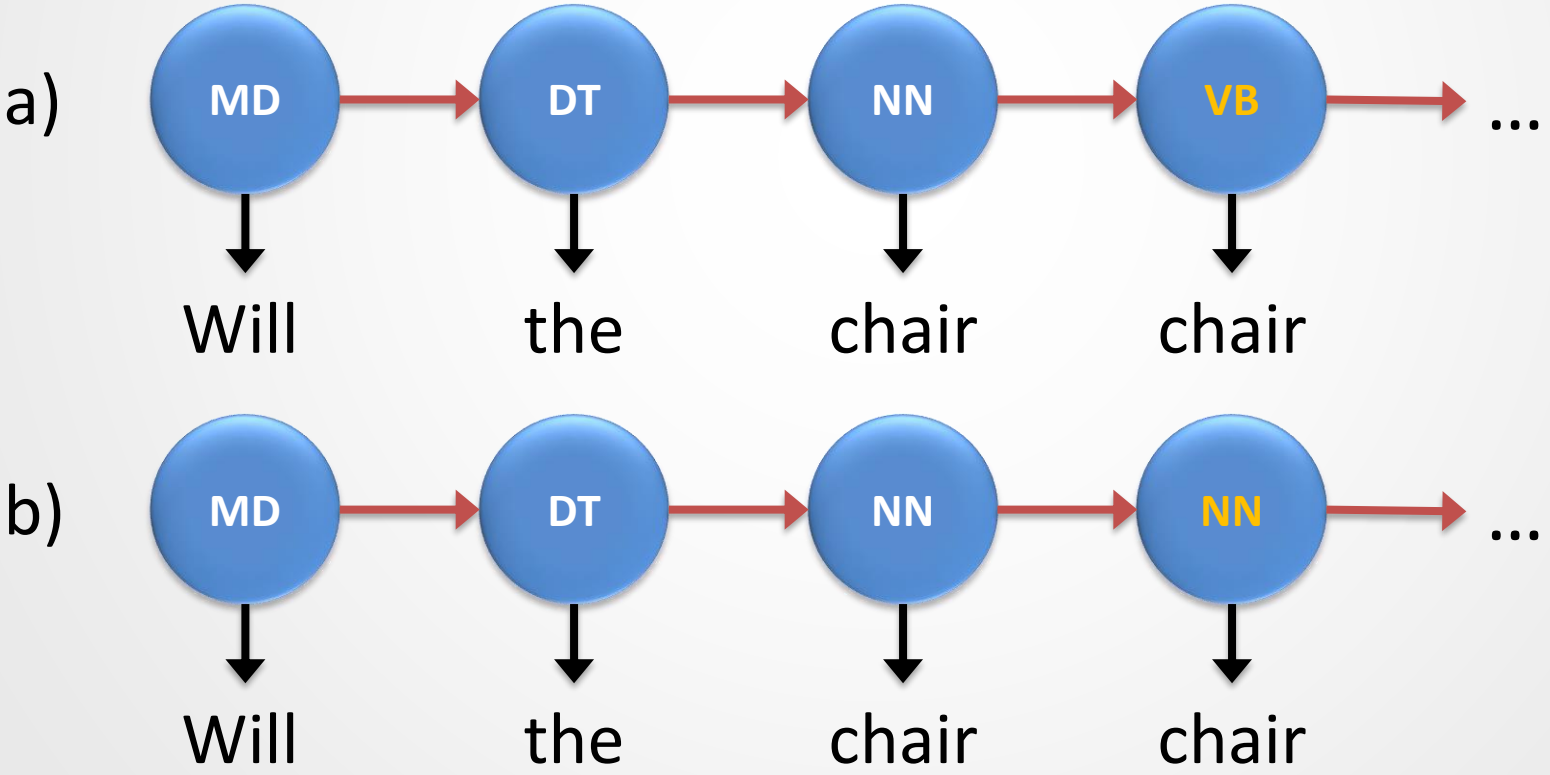


CSC401/2511 – Natural Language Computing – Spring 2019
Lecture 5 Frank Rudzicz and Chloé Pou-Prom
University of Toronto

Revisiting PoS tagging

- *Will/MD the/DT* ***chair/NN*** ***chair/??*** *the/DT meeting/NN from/IN that/DT* ***chair/NN****?*



Now

- (Hidden) Markov models.
  - Using them.
  - Training them.
  - Loving them.

Observable Markov model

- We've seen this type of model:
  - e.g., consider the 7-word vocabulary:

$$\{upside, down, promise, friend, midnight, monster, halloween\}$$

  - What is the probability of the **sequence**

$$upside, down, upside, down, monster?$$

  - Assuming a **bigram** model (i.e., **1st-order Markov**),

$$P(upside|<s>)P(down|upside)P(upside|down) \cdot P(down|upside)P(monster|down)$$

Observable Markov model

- This can be conceptualized graphically.
- We start with N **states**, $s_1, s_2, \ldots, s_N$ that represent unique observations in the world.
- Here, $N = 7$ and each state represents one of the words we can observe.

Observable Markov model

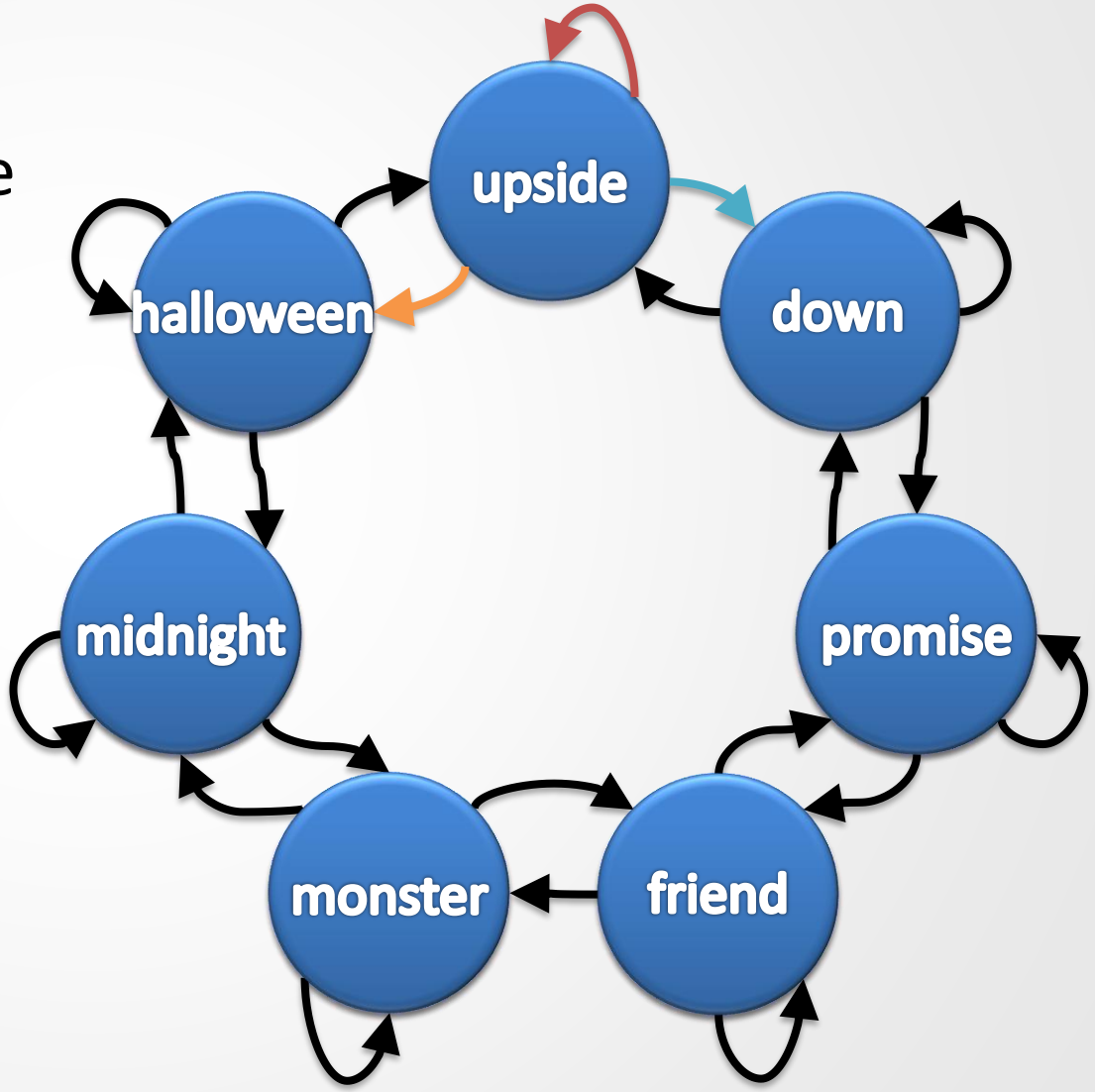
- We have discrete **timesteps**, $t = 0, t = 1, \ldots$
- On the t^{th} timestep the system is in exactly one of the available states, $\boldsymbol{q_t}$.
  - $q_t \in \{s_1, s_2, \ldots, s_N\}$
- We could start in any state. The probability of starting with a particular state s is $P(q_0 = s) = \boldsymbol{\pi(s)}$

Observable Markov model

- At each step we must move to a state with some probability.

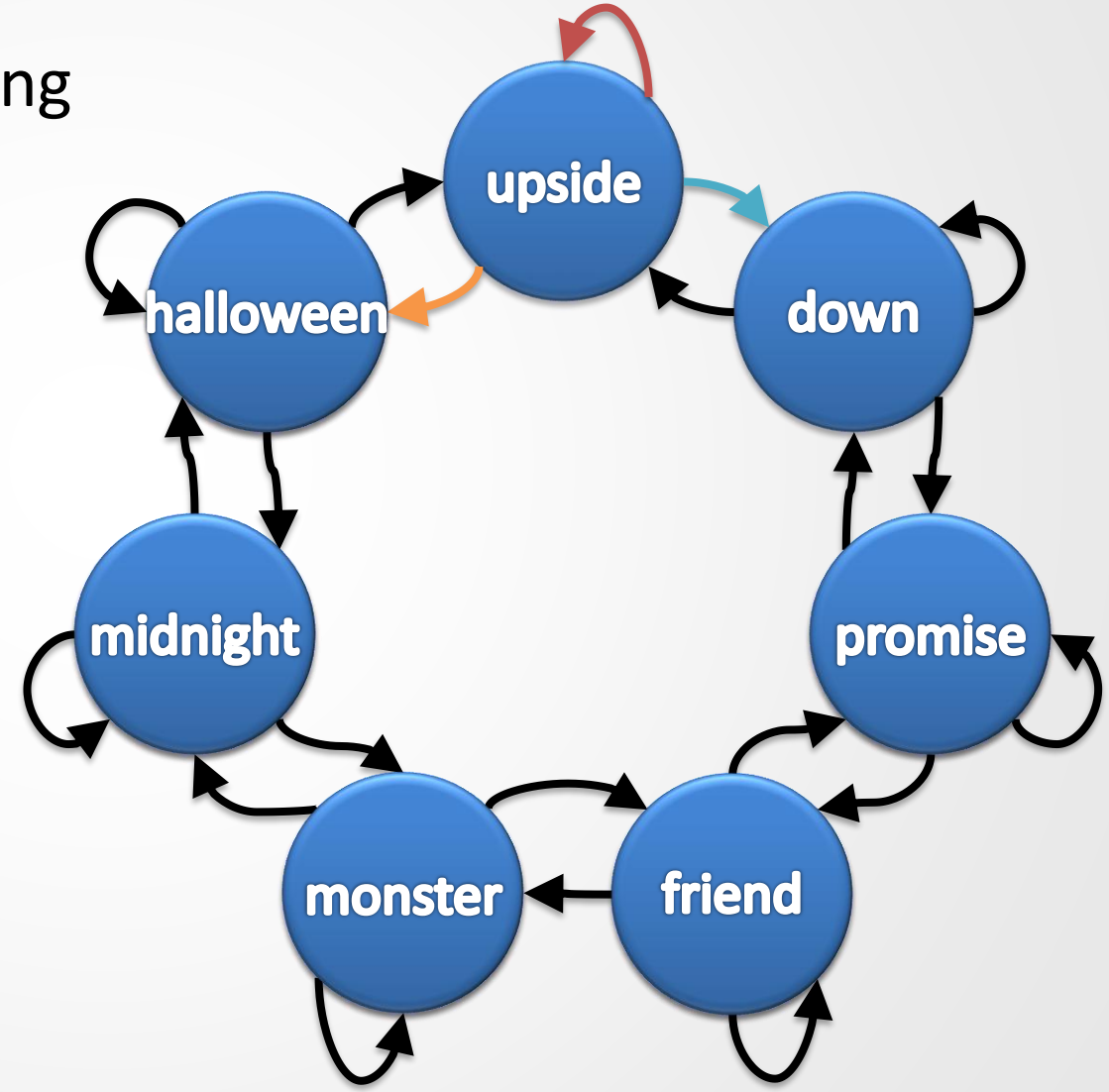
- Here, an arrow from q_t to q_{t+1} represents $P(q_{t+1}|q_t)$

- $P(upside|upside)$
- $P(halloween|upside)$
- $P(down|upside)$
- $P(monster|down) = 0$



Observable Markov model

- Probabilities on all outgoing arcs must sum to 1.
- $P(upside|upside) + P(halloween|upside) + P(down|upside) = 1$
- $P(upside|down) + P(down|down) + P(promise|down) = 1$
- …



A multivariate system

- What if the probabilities of observing words depended ***only*** on some ***other*** variable, like ***mood***?

word	P(word)
upside	0.1
down	0.05
promise	0.05
friend	0.6
monster	0.05
midnight	0.1
halloween	0.05

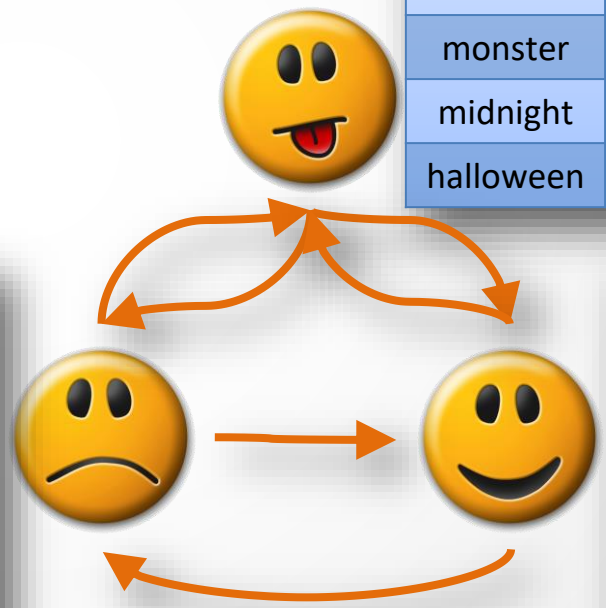
word	P(word)
upside	0.25
down	0.25
promise	0.05
friend	0.3
monster	0.05
midnight	0.09
halloween	0.01

word	P(word)
upside	0.3
down	0
promise	0
friend	0.2
monster	0.05
midnight	0.05
halloween	0.4

A multivariate system

- What if that variable **changes** over time?
  - e.g., I'm **happy** one second and **disgusted** the next.
- Here, **state** ≡ mood
  observation ≡ word.

word	P(word)
upside	0.1
down	0.05
promise	0.05
friend	0.6
monster	0.05
midnight	0.1
halloween	0.05



word	P(word)
upside	0.25
down	0.25
promise	0.05
friend	0.3
monster	0.05
midnight	0.09
halloween	0.01

word	P(word)
upside	0.3
down	0
promise	0
friend	0.2
monster	0.05
midnight	0.05
halloween	0.4

Observable multivariate systems

- Imagine you have access to my emotional state somehow.
- All your data are completely **observable** at every timestep.
- E.g.,

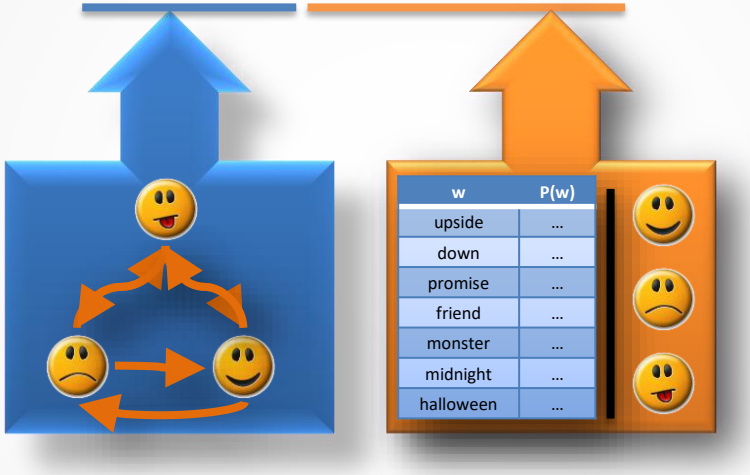
t	0	1	2	…
state	🙂	🙂	😛	…
word	*midnight*	*friend*	*upside*	…

≡

$\langle midnight, friend, upside \rangle, \langle 🙂, 🙂, 😛 \rangle$

Observable multivariate systems

- What is the probability of a sequence of words and states?
  - $P(w_{0:t}, q_{0:t}) = P(q_{0:t})P(w_{0:t}|q_{0:t}) \approx \prod_{i=0}^{t} P(q_i|q_{i-1})P(w_i|q_i)$

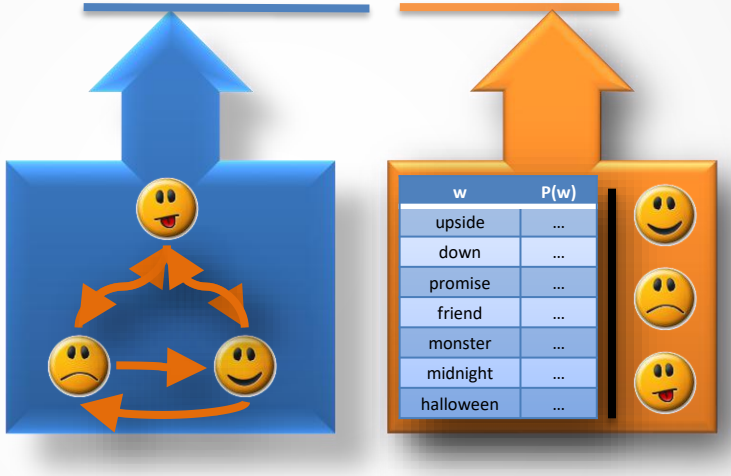


- e.g.,

$$P(\langle friend, upside \rangle, \langle \text{☺} \; \text{😛} \rangle) = P(q_0 = \text{☺}) \cdot P(friend|\text{☺}) \cdot P(\text{😛}|\text{☺}) \cdot P(upside|\text{😛})$$

Observable multivariate systems

- **Q**: How do you **learn** these probabilities?
  - $P(w_{0:t}, q_{0:t}) \approx \prod_{i=0}^{t} P(q_i|q_{i-1})P(w_i|q_i)$

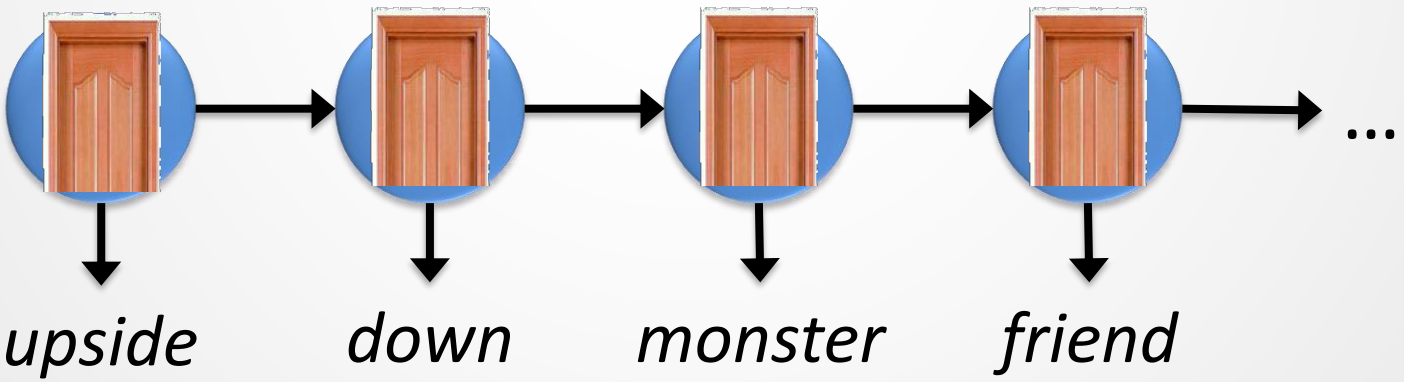


- **A**: When all data are observed, basically the same as before.
  - $P(q_i|q_{i-1}) = \frac{P(q_{i-1}q_i)}{P(q_{i-1})}$ is learned with MLE from training data.
  - $P(w_i|q_i) = \frac{P(w_i,q_i)}{P(q_i)}$ is also learned with MLE from training data.

Hidden variables

- Q: What if you **don't** know the **states** during *testing*?
  - e.g., compute $P(\langle upside, down, monster, friend \rangle)$

- Q: What if you **don't** know the **states** during *training*?



Examples of hidden phenomena

- We want to represent **surface** (i.e., **observable**) phenomena as the **output** of **hidden** underlying systems.
  - e.g.,
    - **Words** are the outputs of hidden **parts-of-speech**,
    - **French phrases** are the outputs of hidden **English phrases**,
    - **Speech sounds** are the outputs of hidden **phonemes**.
  - in other fields,
    - **Encrypted symbols** are the outputs of hidden **messages**,
    - **Genes** are the outputs of hidden **functional relationships**,
    - **Weather** is the output of hidden **climate conditions**,
    - **Stock prices** are the outputs of hidden **market conditions**,
    - …

Definition of an HMM

- A **hidden Markov model** (HMM) is specified by the 5-tuple $\{S, W, \Pi, A, B\}$:
  - $S = \{s_1, \ldots, s_N\}$: set of **states** (e.g., moods)
  - $W = \{w_1, \ldots, w_K\}$: output **alphabet** (e.g., words)

θ
  - $\Pi = \{\pi_1, \ldots, \pi_N\}$: **initial** state probabilities
  - $A = \{a_{ij}\}, i, j \in S$: state **transition** probabilities
  - $B = b_i(w), i \in S, w \in W$: state **output** probabilities

yielding

  - $Q = \{q_0, \ldots, q_T\}, q_i \in S$: state sequence
  - $\mathcal{O} = \{o_0, \ldots, o_T\}, o_i \in W$: output sequence

A hidden Markov production process

- An HMM is a **representation** of a process in the world.
  - We can *synthesize* data, as in Shannon's game.
- This is how an HMM **generates** new sequences:

- $t := 0$
- **Start** in state $q_0 = s_i$ with probability π_i
- **Emit** observation symbol $o_0 = w_k$ with probability $b_i(o_0)$
- **While** (**not** forever)
  - **Go** from state $q_t = s_i$ to state $q_{t+1} = s_j$ with probability a_{ij}
  - **Emit** observation symbol $o_{t+1} = w_k$ with probability $b_j(o_{t+1})$
  - $t := t + 1$

Fundamental tasks for HMMs

1. Given a **model** with particular parameters $\theta = \langle \Pi, A, B \rangle$, how do we efficiently compute the likelihood of a *particular* **observation sequence**, $P(\mathcal{O}; \theta)$?

We previously computed the probabilities of word sequences using *N*-grams.

The probability of a particular sequence is usually useful as a means to some other end.

Fundamental tasks for HMMs

2. Given an **observation sequence** $\mathcal{O}$ and a **model** θ, how do we choose a **state sequence** $Q = \{q_0, \ldots, q_T\}$ that *best explains* the observations?

This is the task of **inference** – i.e., guessing at the best explanation of unknown ('latent') variables given our model.

This is often an important part of **classification**.

Fundamental tasks for HMMs

3. Given a large **observation sequence** $\mathcal{O}$, how do we choose the *best parameters* $\theta = \langle \Pi, A, B \rangle$ that explain the data $\mathcal{O}$?

This is the task of **training**.

As before, we want our parameters to be set so that the available training data is maximally likely,
But doing so will involve guessing unseen information.

Task 1: Computing $P(\mathcal{O}; \theta)$

- We've seen the probability of a joint sequence of observations and states:

$$P(\mathcal{O}, Q; \theta) = P(\mathcal{O}|Q; \theta)P(Q; \theta)$$
$$= \pi_{q_0} b_{q_0}(o_0) a_{q_0 q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \ldots$$

- To get the probability of our observations **without** seeing the state, we must **sum over all possible state sequences**:

$$P(\mathcal{O}; \theta) = \sum_Q P(\mathcal{O}|Q; \theta)P(Q; \theta).$$

Computing $P(\mathcal{O}; \theta)$ naïvely

- To get the total probability of our observations, we could *directly* **sum over all possible state sequences**:

$$P(\mathcal{O}; \theta) = \sum_Q P(\mathcal{O}|Q; \theta)P(Q; \theta).$$

- For observations of length T, each state sequence involves $2T$ multiplications (1 for each state transition, 1 for each observation, 1 for the start state, minus 1).
- There are up to N^T possible state sequences of length T given N states.

$$\therefore \sim(1 + T + T - 1) \cdot N^T \text{ multiplications}$$ 🙁

Computing $P(\mathcal{O}; \theta)$ cleverly

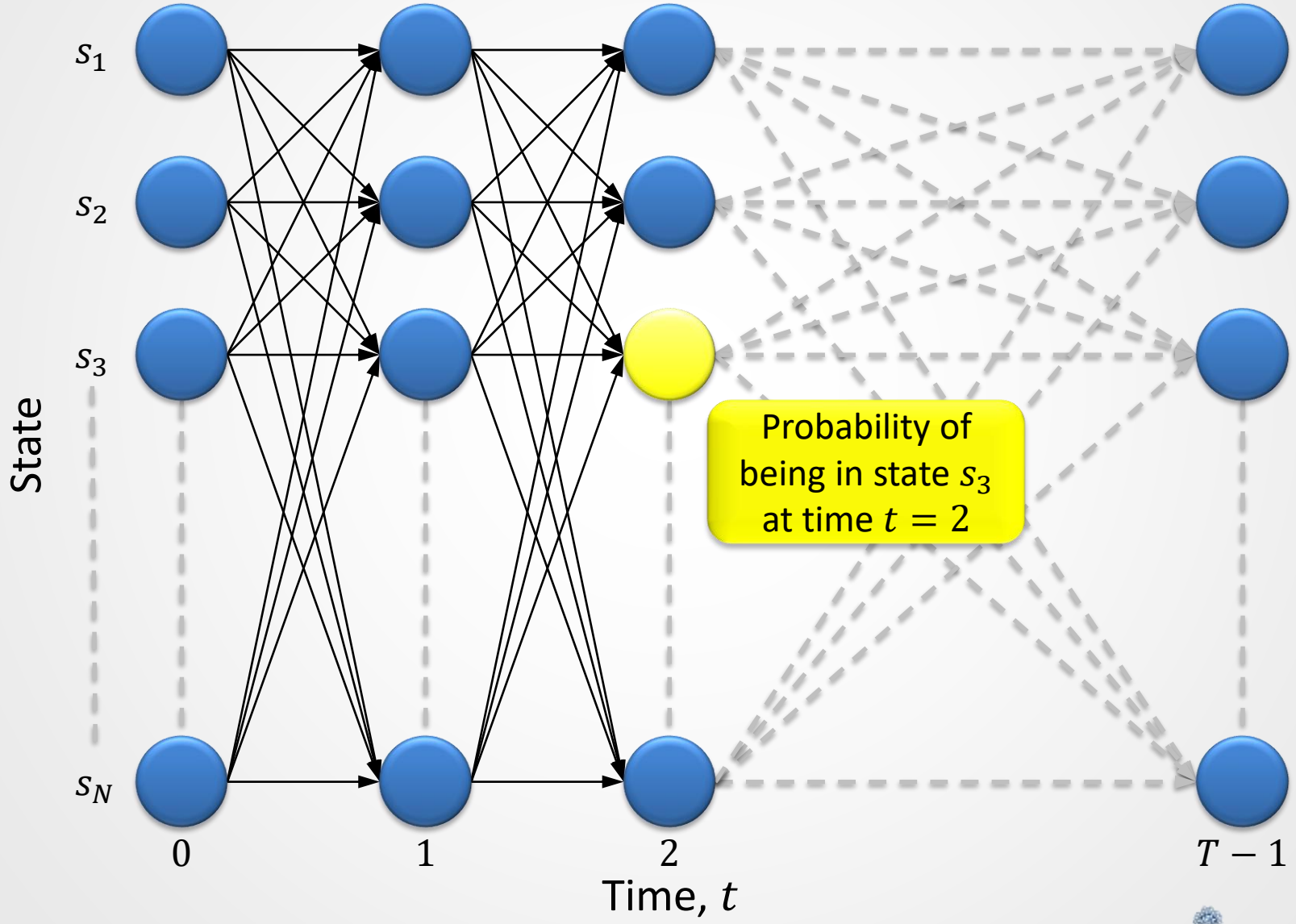
- To avoid this complexity, we use **dynamic programming**; we **remember**, rather than **recompute**, partial results.

- We make a **trellis** which is an array of **states vs. time**.
  - The element at (i, t) is $\boldsymbol{\alpha_i(t)}$

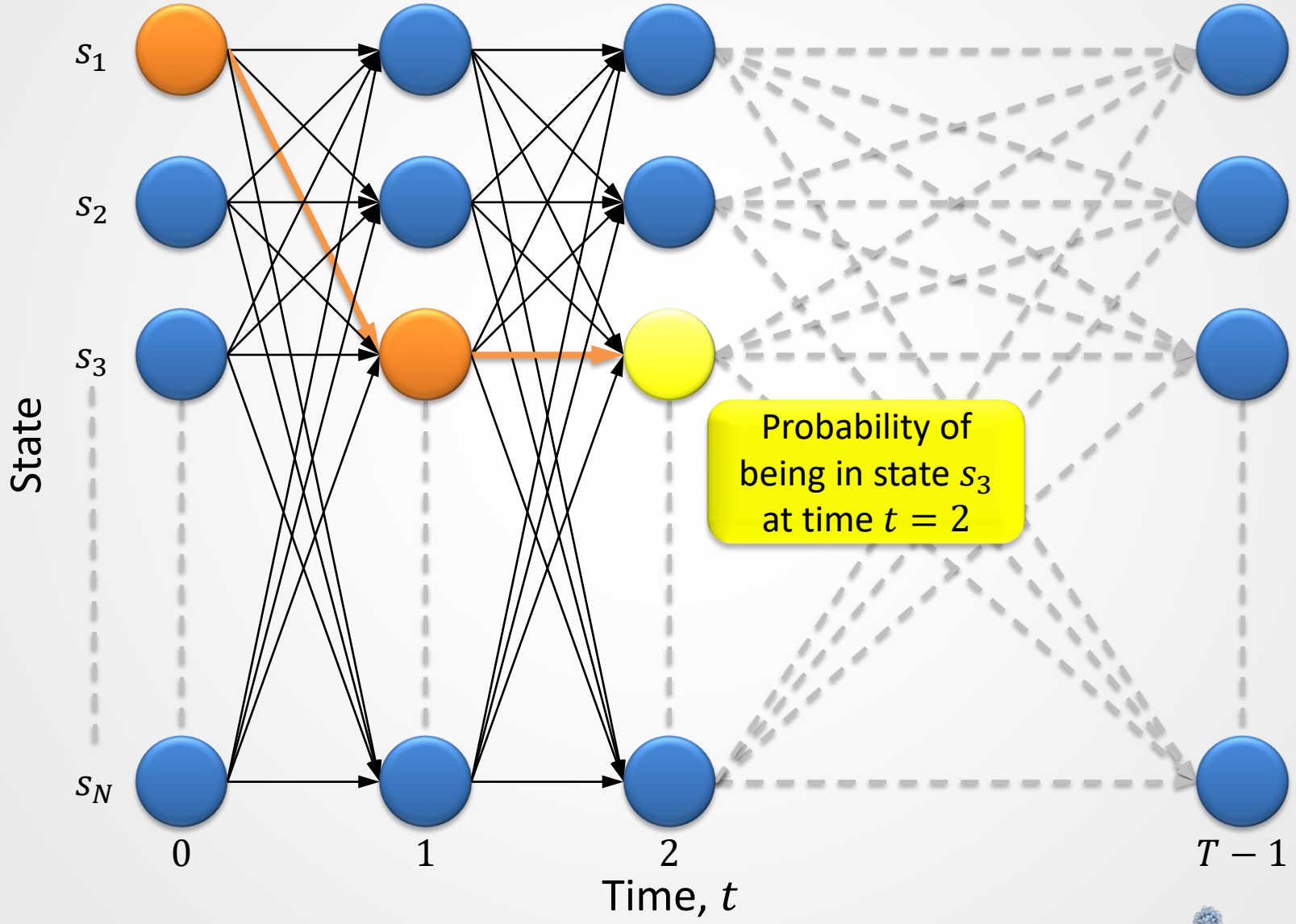
    the probability of being in state i at time t *after seeing all previous observations*:

    $P(\mathcal{o}_{o:t-1}, q_t = s_i; \theta)$

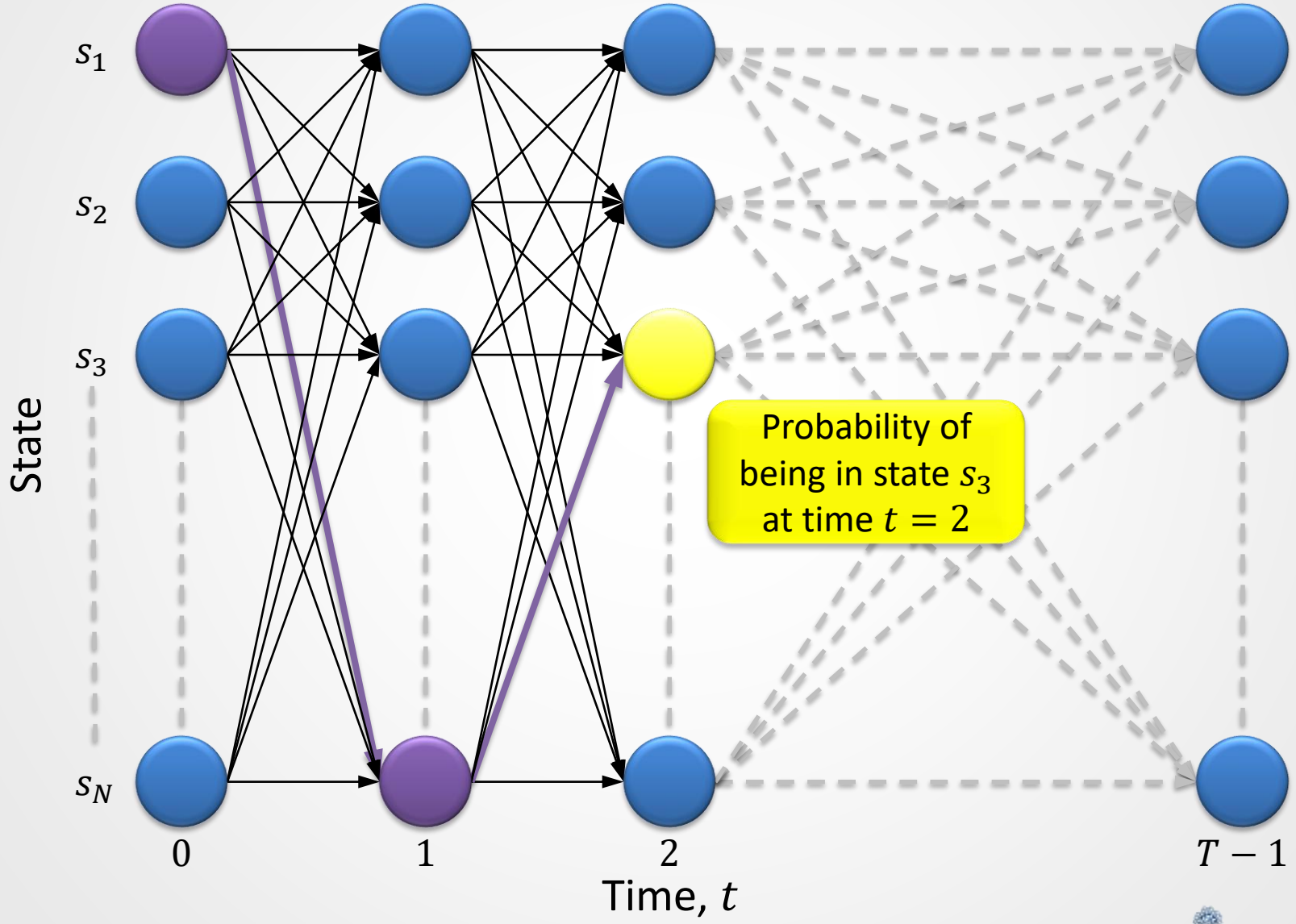
Trellis


s_1
s_2
s_3
s_N
State
0
1
2
$T-1$
Time, t
Probability of being in state s_3 at time $t = 2$

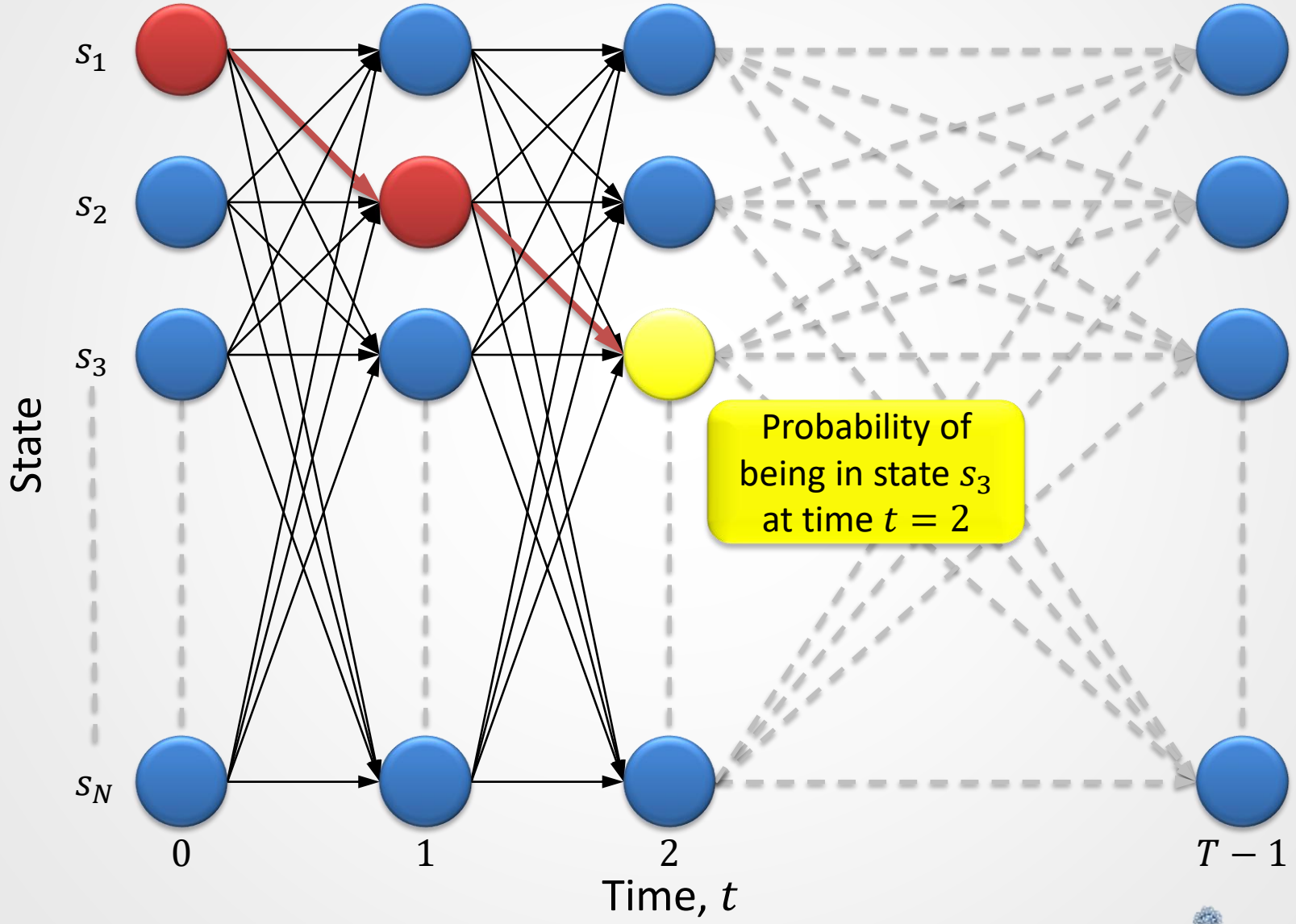
Alternative paths through the trellis



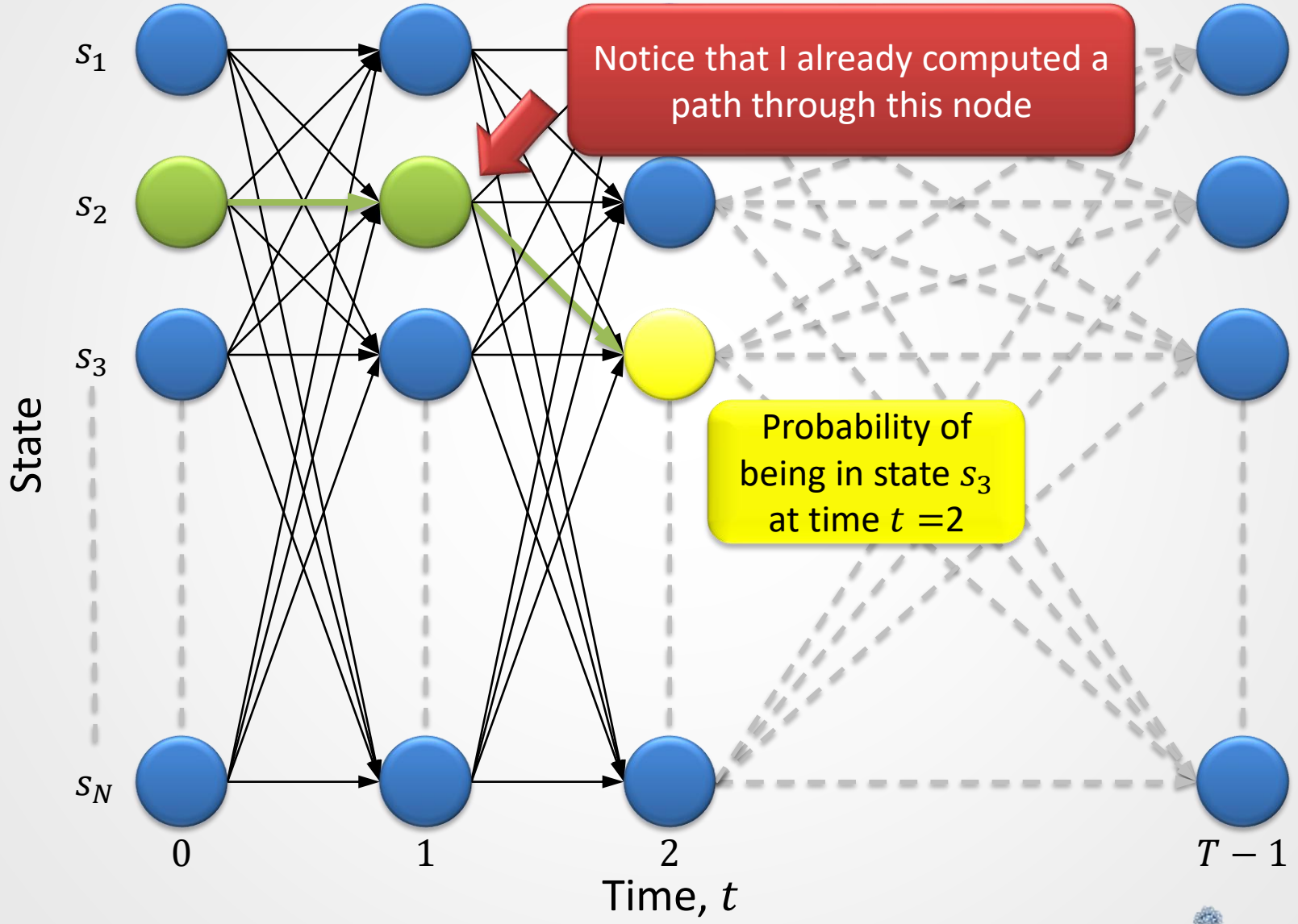
Alternative paths through the trellis



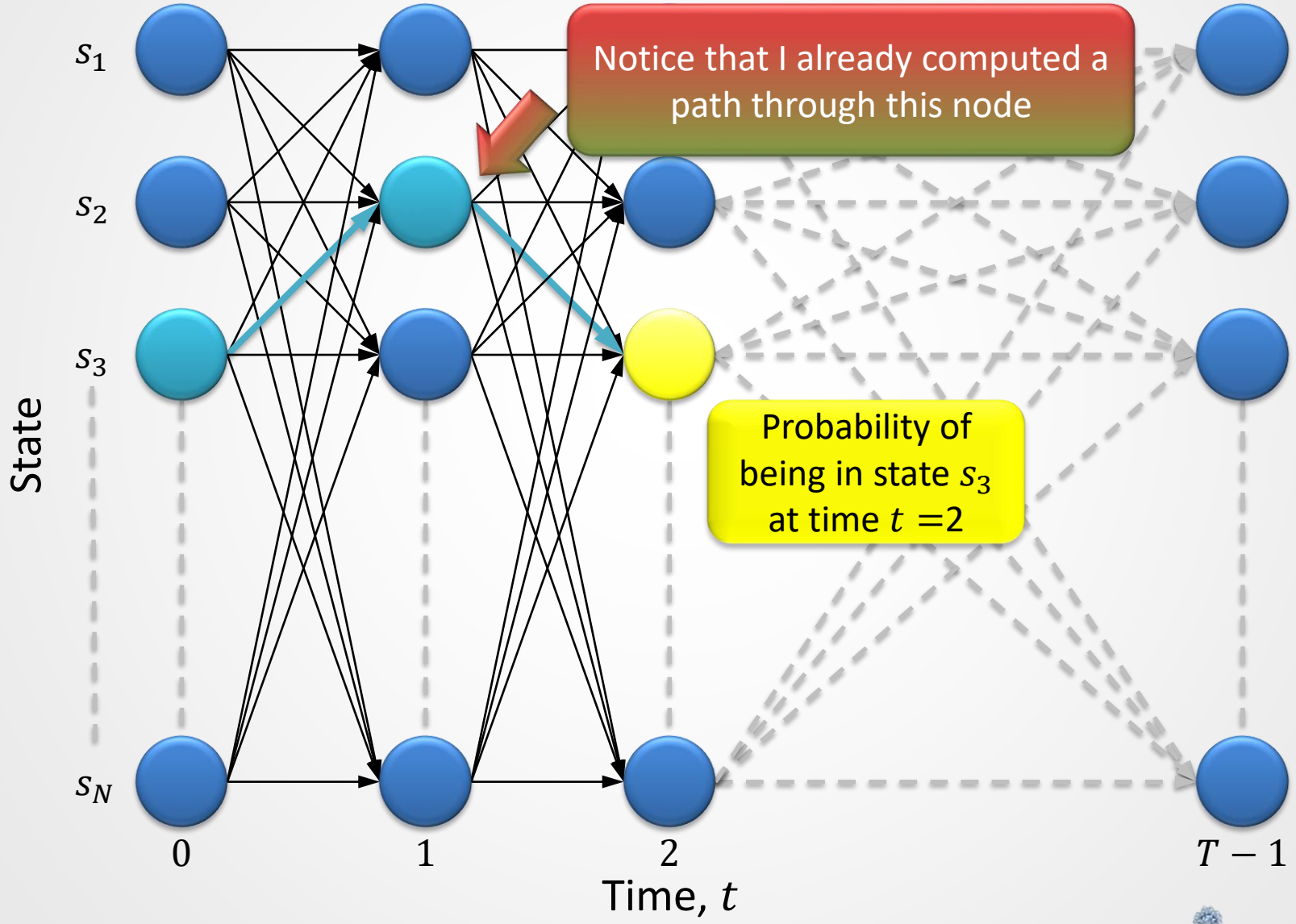
Alternative paths through the trellis



Alternative paths through the trellis

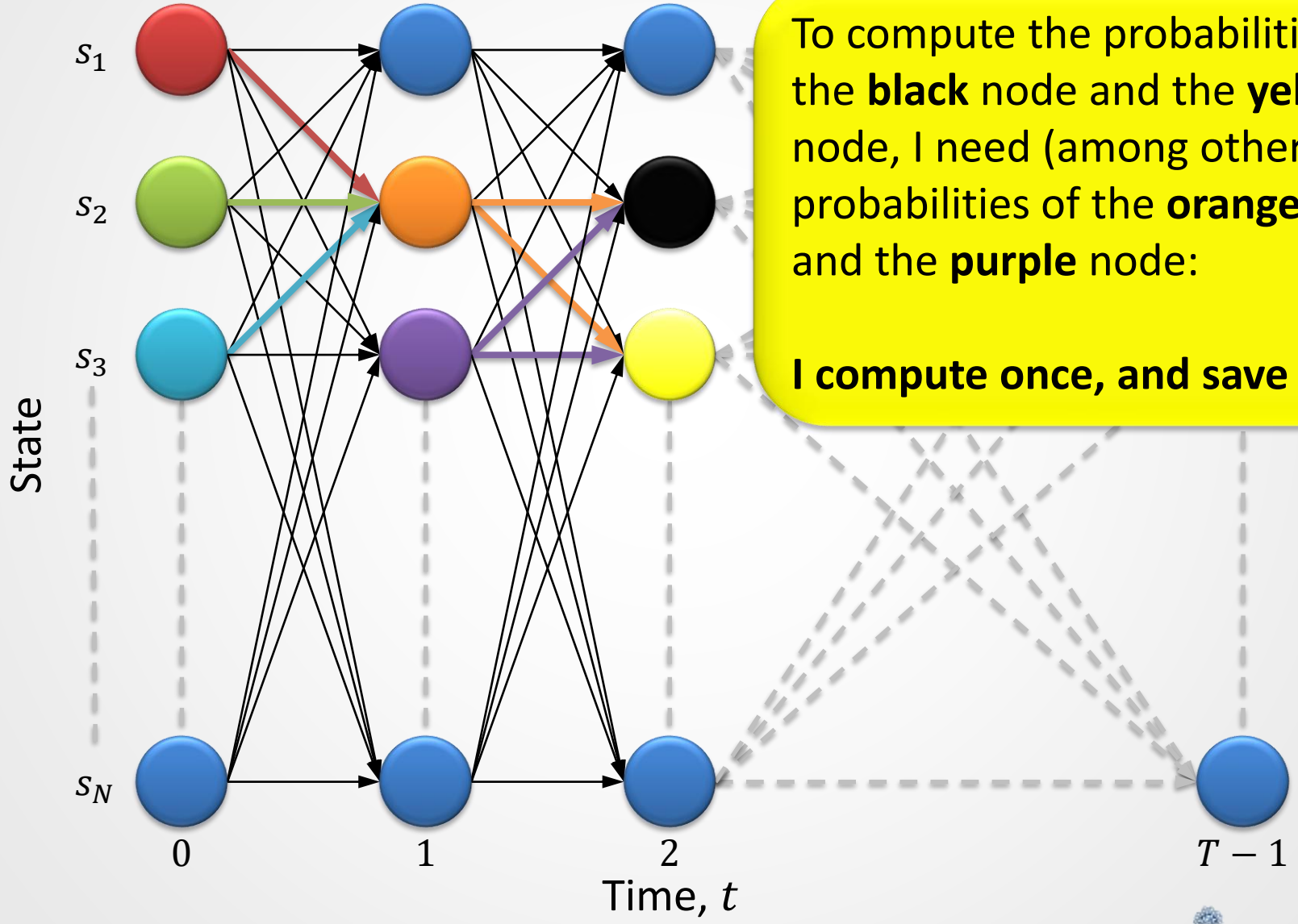


Alternative paths through the trellis



AND SO ON…

Trellis



To compute the probabilities of the **black** node and the **yellow** node, I need (among others) the probabilities of the **orange** node and the **purple** node:

I compute once, and save them.

The Forward procedure

- To compute

$$\alpha_i(t) = P(\mathcal{O}_{0:t}, q_t = s_i; \theta)$$

  we can compute $\alpha_j(t-1)$ for possible *previous* states s_j, then use our knowledge of a_{ji} and $b_i(\mathcal{O}_t)$

- We compute the trellis **left-to-right** (because of the convention of time) and **top-to-bottom** ('just because').

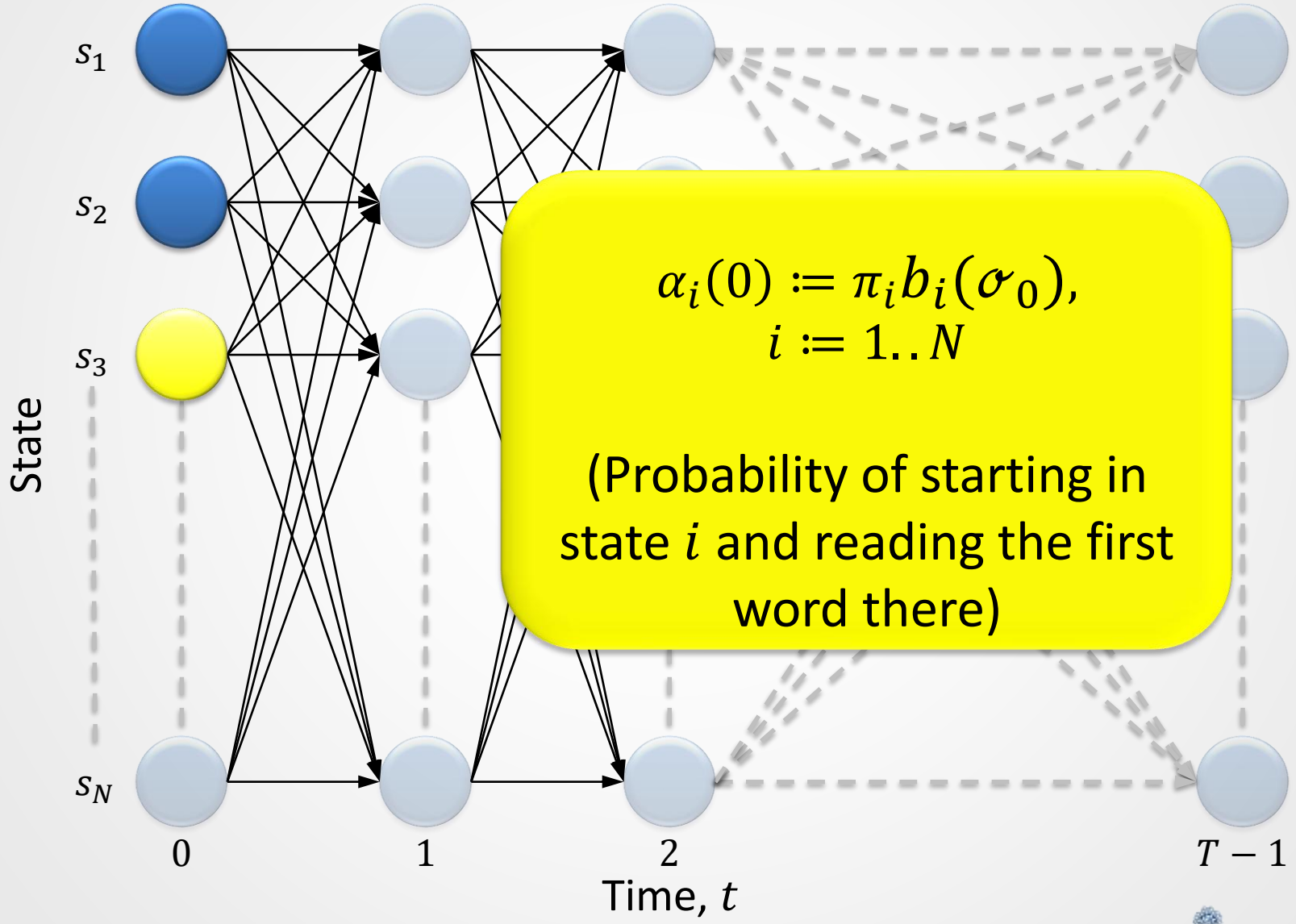
- **Remember**: $\mathcal{O}_t$ is fixed and known.
  $\alpha_i(t)$ is agnostic of the future.

The Forward procedure

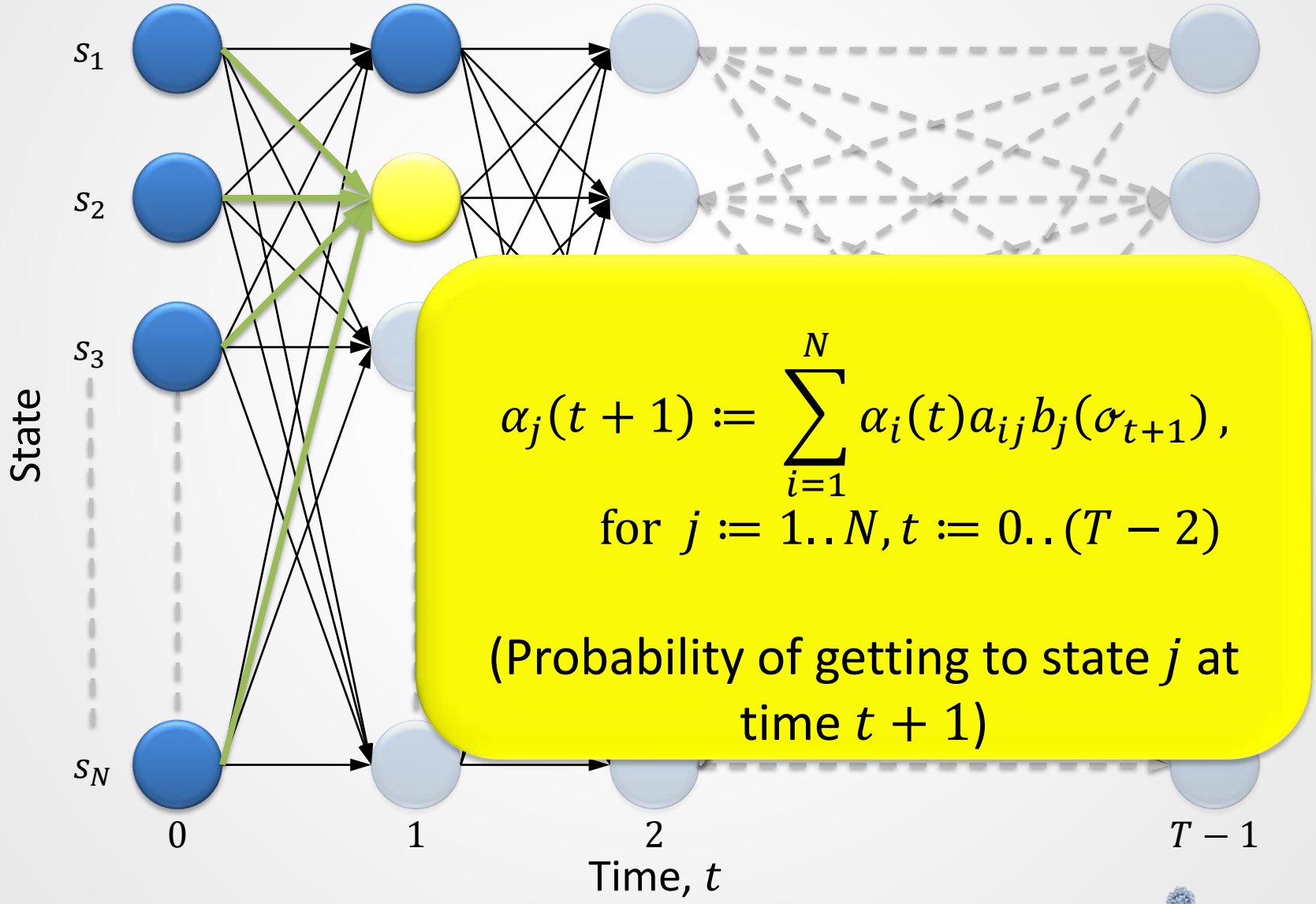
- The trellis is computed **left-to-right** and **top-to-bottom**.

- There are three steps in this procedure:
  - **Initialization**: Compute the nodes in the ***first*** *column* of the trellis ($t = 0$).
  - **Induction**: Iteratively compute the nodes in the *rest* of the trellis ($1 \leq t < T$).
  - **Conclusion**: Sum over the nodes in the ***last*** *column* of the trellis ($t = T - 1$).

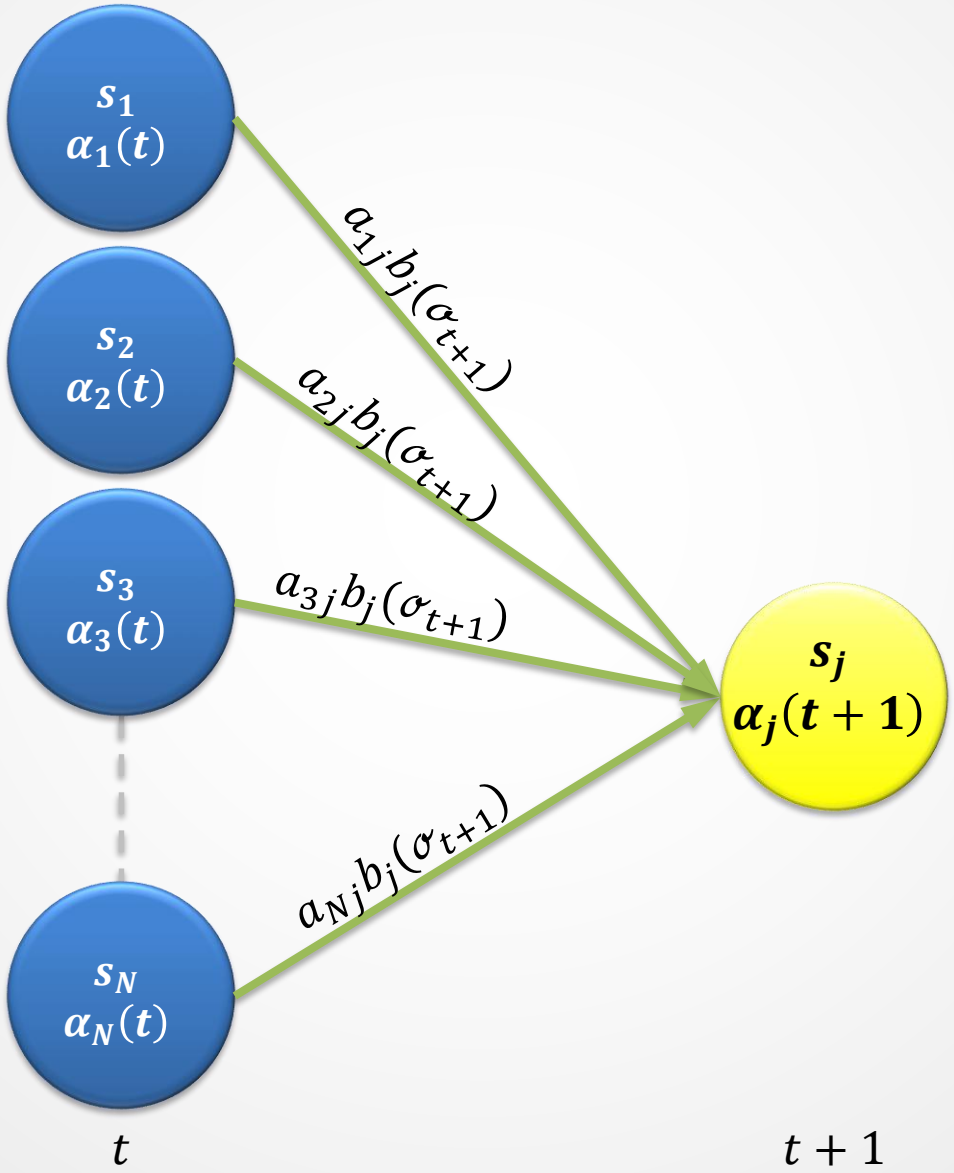
Initialization of Forward procedure



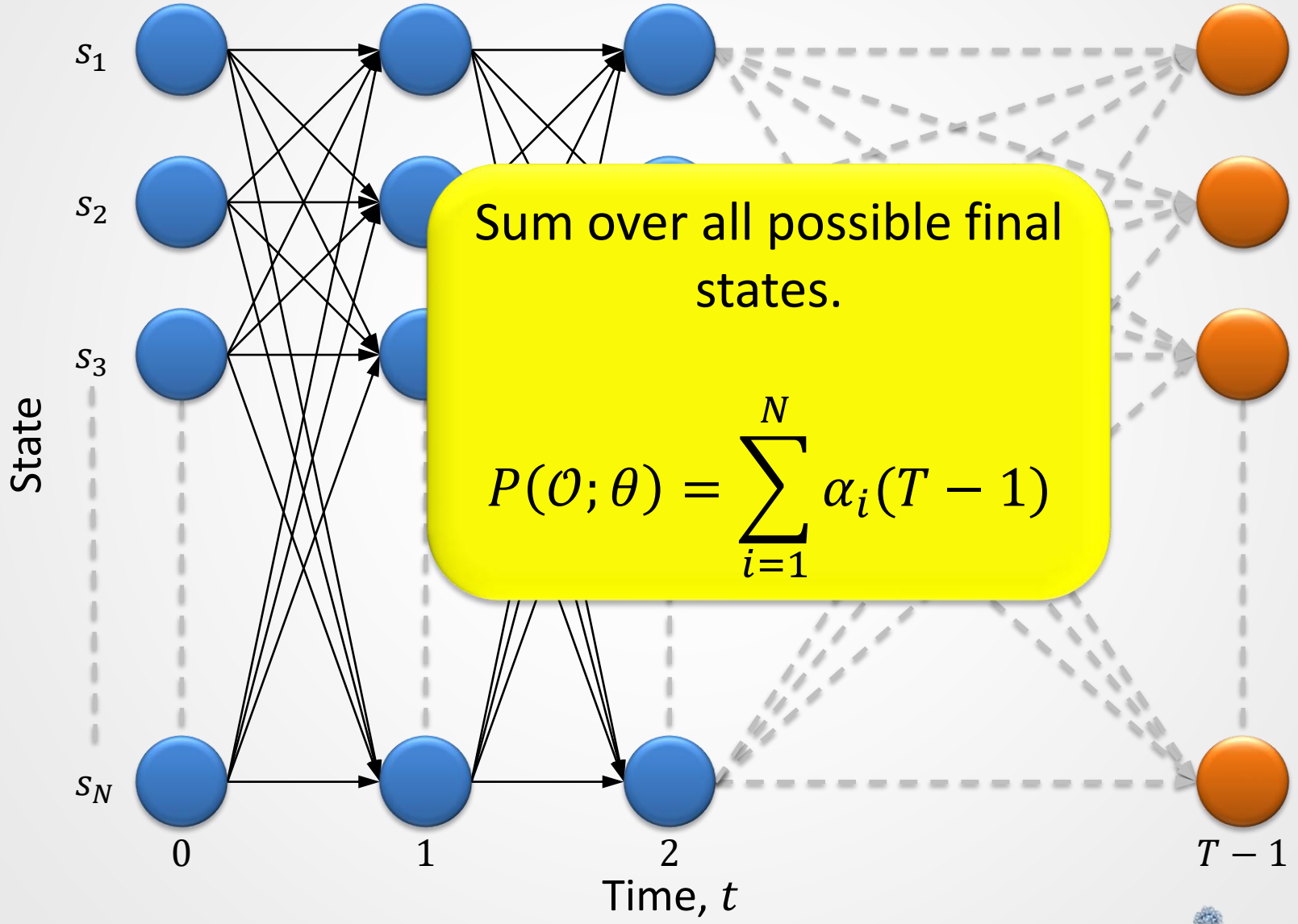
Induction of Forward procedure



Induction of Forward procedure

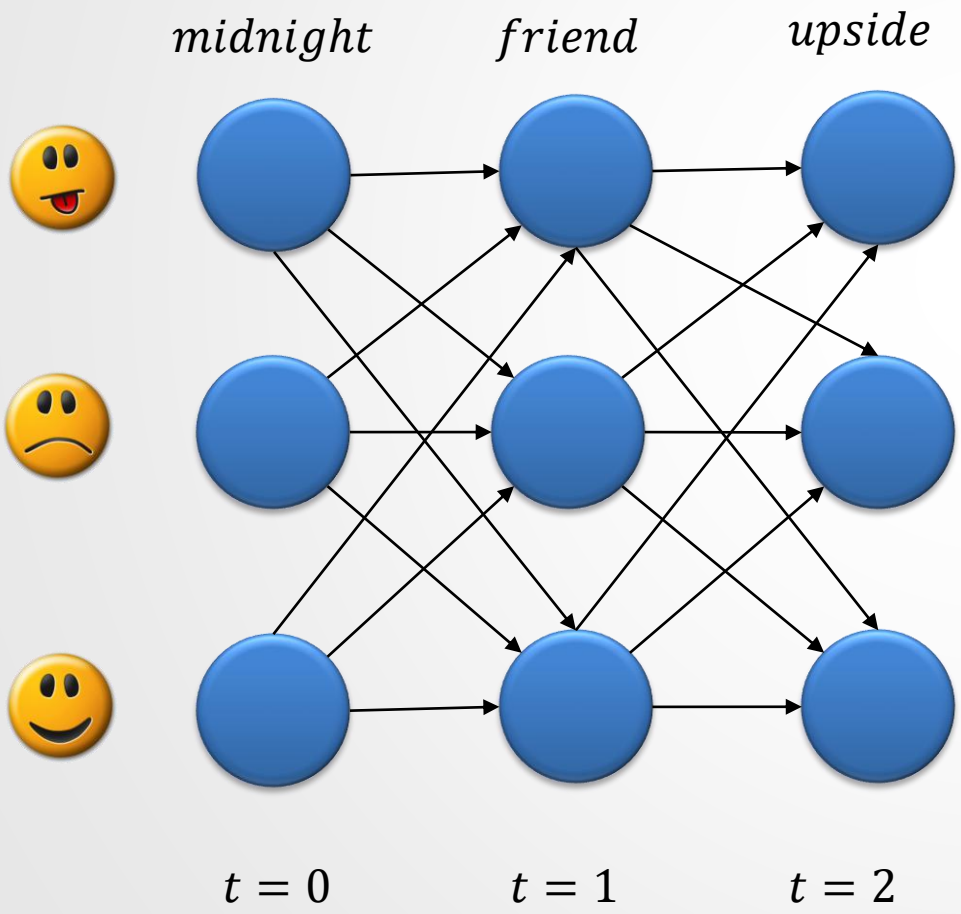


Conclusion of Forward procedure



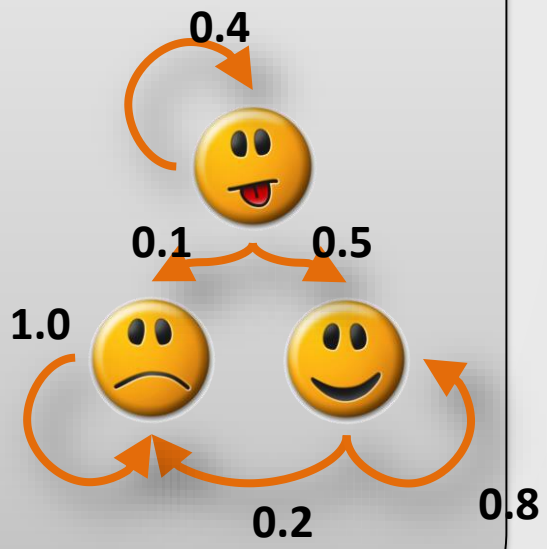
The Forward procedure - Example

- Let's compute $\mathrm{P}(\langle midnight, friend, upside \rangle)$



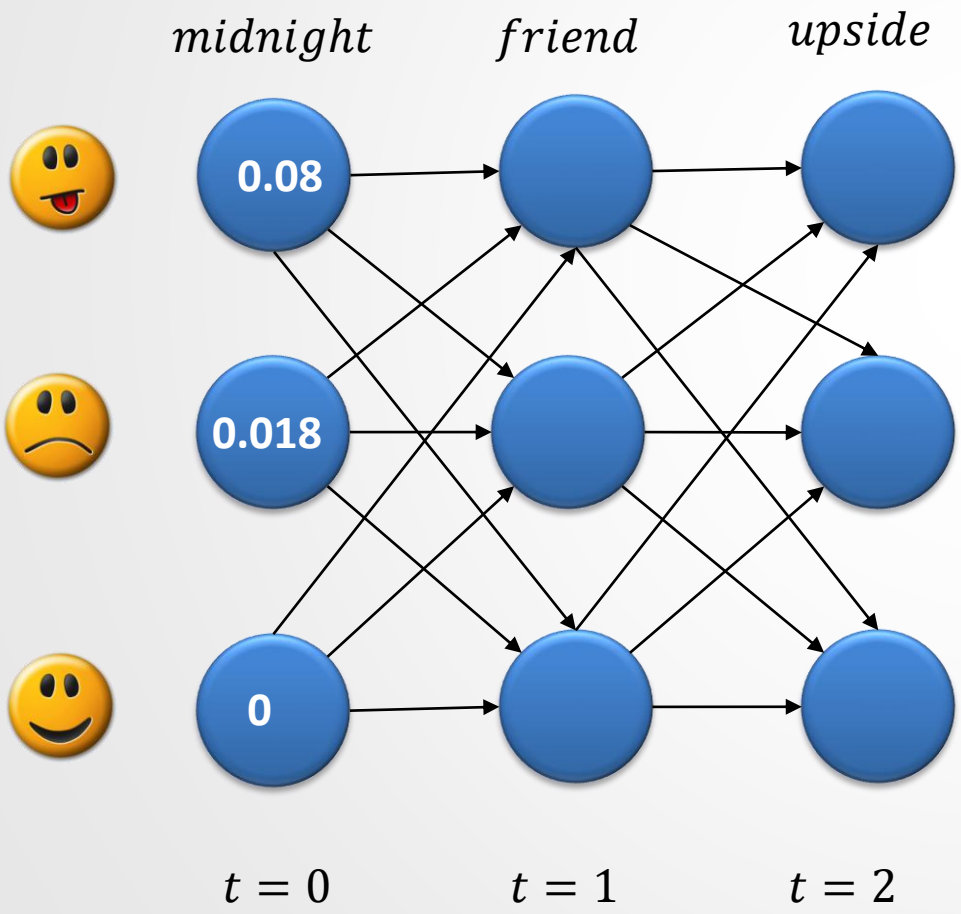
We need initial state probabilities $\boldsymbol{\Pi}$ and transition probabilities $\boldsymbol{\alpha_{ij}}$

	$\boldsymbol{\Pi}$ = 0.80
	$\boldsymbol{\Pi}$ = 0.20
	$\boldsymbol{\Pi}$ = 0



The Forward procedure - Example

- Let's compute $\mathrm{P}(\langle midnight, friend, upside \rangle)$



Initialization

Compute the probability of starting in state i and reading the first word there

$$\boldsymbol{\alpha_i(0) \coloneqq \pi_i b_i(o_0)}$$

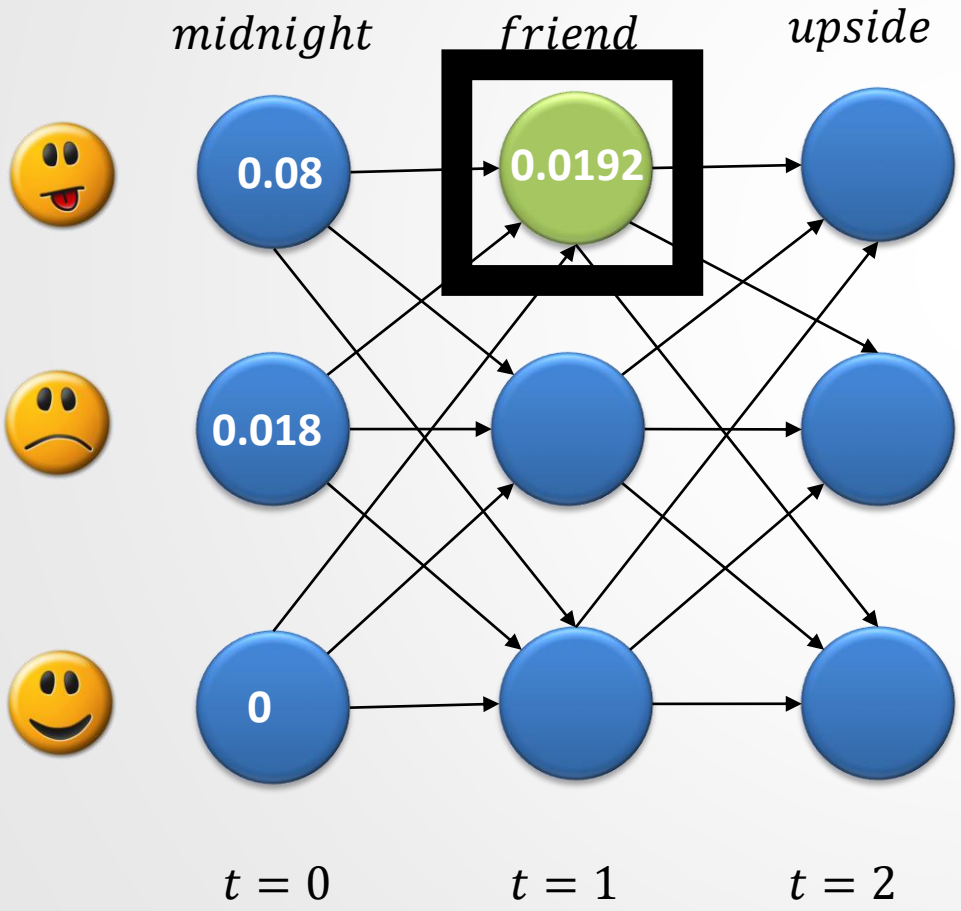
$$\boldsymbol{\alpha(0) = 0.80 \times 0.10 = 0.08}$$

$$\boldsymbol{\alpha(0) = 0.20 \times 0.09 = 0.018}$$

$$\boldsymbol{\alpha(0) = 0 \times 0.05 = 0}$$

The Forward procedure - Example

- Let's compute $\mathrm{P}(\langle midnight, friend, upside \rangle)$



Induction

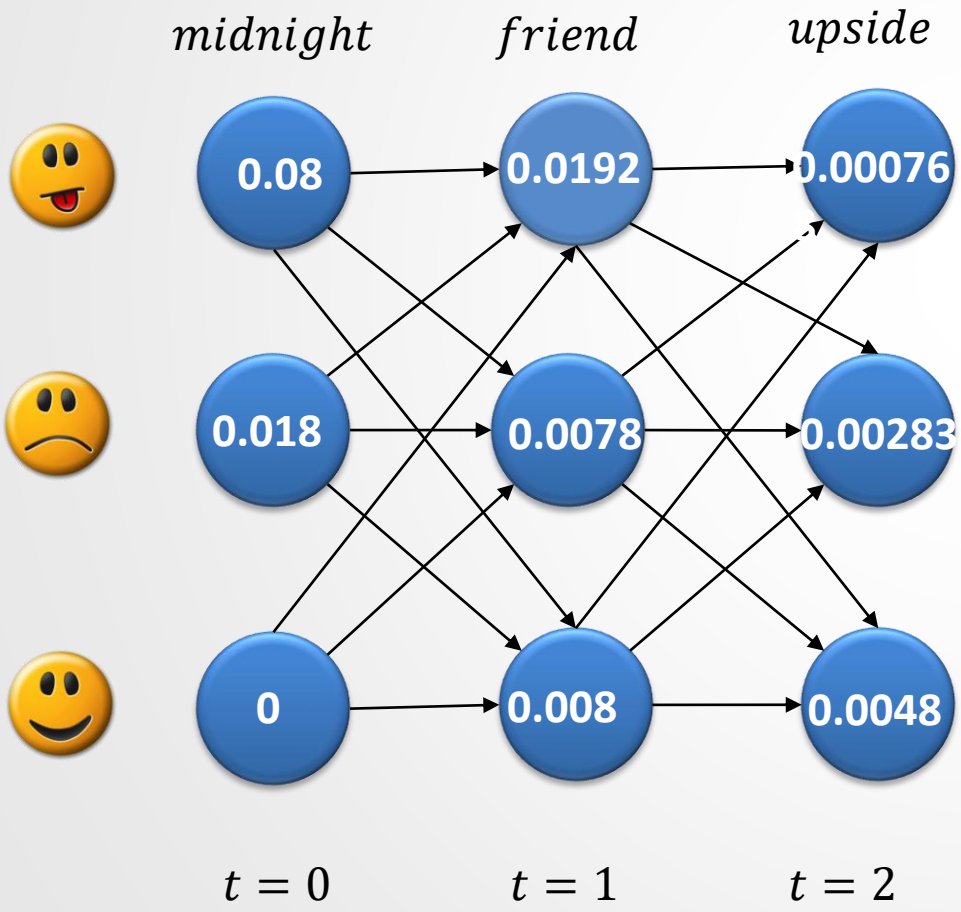
Iteratively compute the rest of the nodes in the trellis; i.e., the probability of getting to state *j* at time *t+1*

$$\alpha_j(t+1) := \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_j(\sigma_{t+1})$$

$$\begin{aligned}\boldsymbol{\alpha(t+1)} &= \mathbf{0.08(0.4)(0.6)} \\ &+ \mathbf{0.018(0)(0.6)} \\ &+ \mathbf{0(0)(0.6)} \\ &= \mathbf{0.0192}\end{aligned}$$

The Forward procedure - Example

- Let's compute $\mathrm{P}(\langle midnight, friend, upside \rangle)$



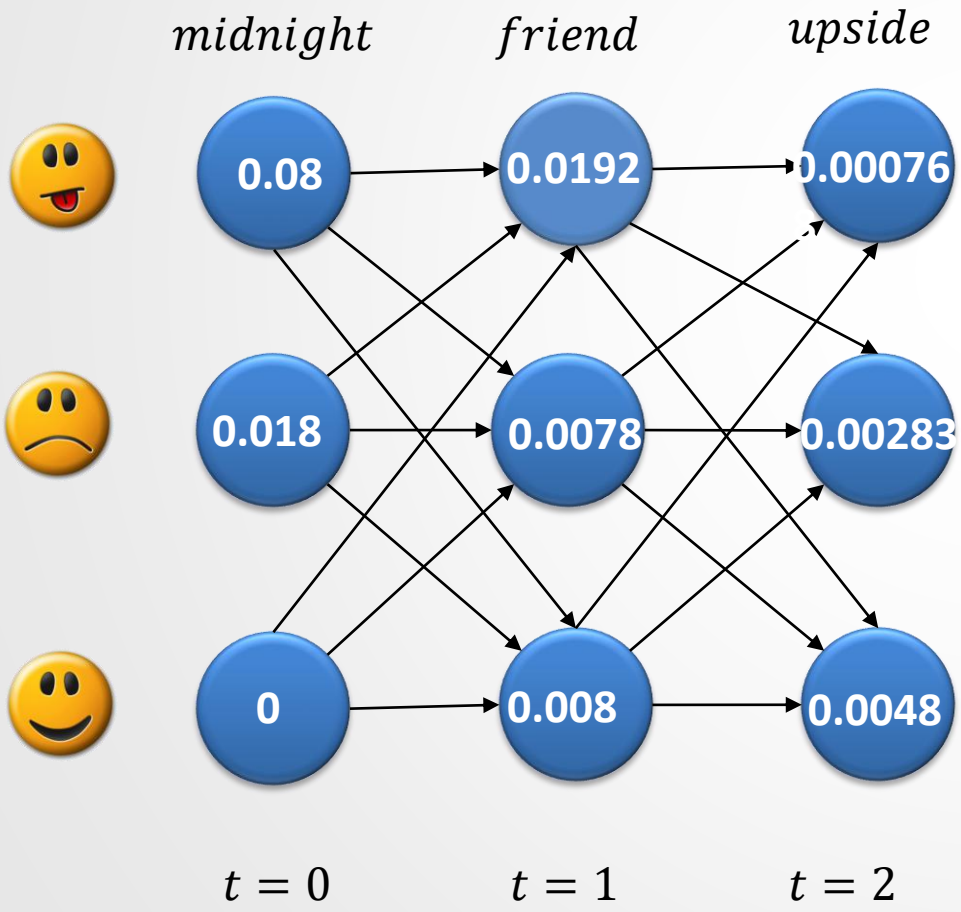
Induction

Iteratively compute the rest of the nodes in the trellis; i.e., the probability of getting to state *j* at time *t+1*

$$\alpha_j(t+1) := \sum_{i=1}^{N} \alpha_i(t) a_{ij} b_j(\sigma_{t+1})$$

The Forward procedure - Example

- Let's compute $\mathrm{P}(\langle midnight, friend, upside \rangle)$



Conclusion

Sum over all possible final states

$$\boldsymbol{P(\mathcal{O};\theta) = \sum_{i=1}^{N} \alpha_i(T-1)}$$

$$P(\mathcal{O};\theta) = 0.00076 + 0.00283 + 0.0048 = \mathbf{0.00839}$$

The Forward procedure

- The naïve approach needed $(2T) \cdot N^T$ multiplications.
- The Forward procedure (using **dynamic programming**) needs only $2N^2T$ multiplications. 🙂
- The Forward procedure gives us $P(\mathcal{O}; \theta)$.
- Clearly, but less intuitively, we can also compute the trellis from back-to-front, i.e., *backwards in time*…

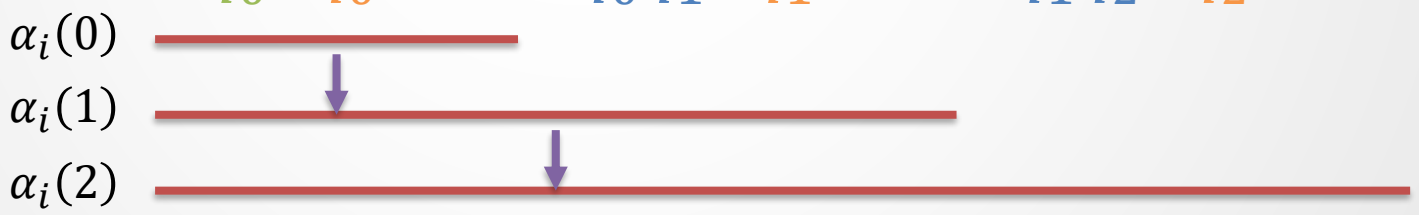
Remember the point

- The point was to compute the equivalent of

$$P(\mathcal{O};\theta) = \sum_{Q} P(\mathcal{O}, Q;\theta)$$

where

$$P(\mathcal{O}, Q;\theta) = P(\mathcal{O}|Q;\theta)P(Q;\theta)$$
$$= \pi_{q_0} b_{q_0}(o_0) a_{q_0 q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \ldots$$



The Forward algorithm stores all possible 1-state sequences (from the start), to store all possible 2-state sequences (from the start), to store all possible 3-state sequences (from the start)…

Remember the point

- But, we can compute these factors in reverse

$$P(\mathcal{O}, Q; \theta) = P(\mathcal{O}|Q; \theta)P(Q; \theta)$$

$$= \pi_{q_0} \dots b_{q_{T-2}}(\sigma_{T-2}) a_{q_{T-2}q_{T-1}} b_{q_{T-1}}(\sigma_{T-1}) a_{q_{T-1}q_T} b_{q_T}(\sigma_T)$$

We can still deal with sequences that evolve *forward* in time, but simply **store temporary results in reverse**…

The Backward procedure

- In the $(i,t)^{th}$ node of the **trellis**, we store

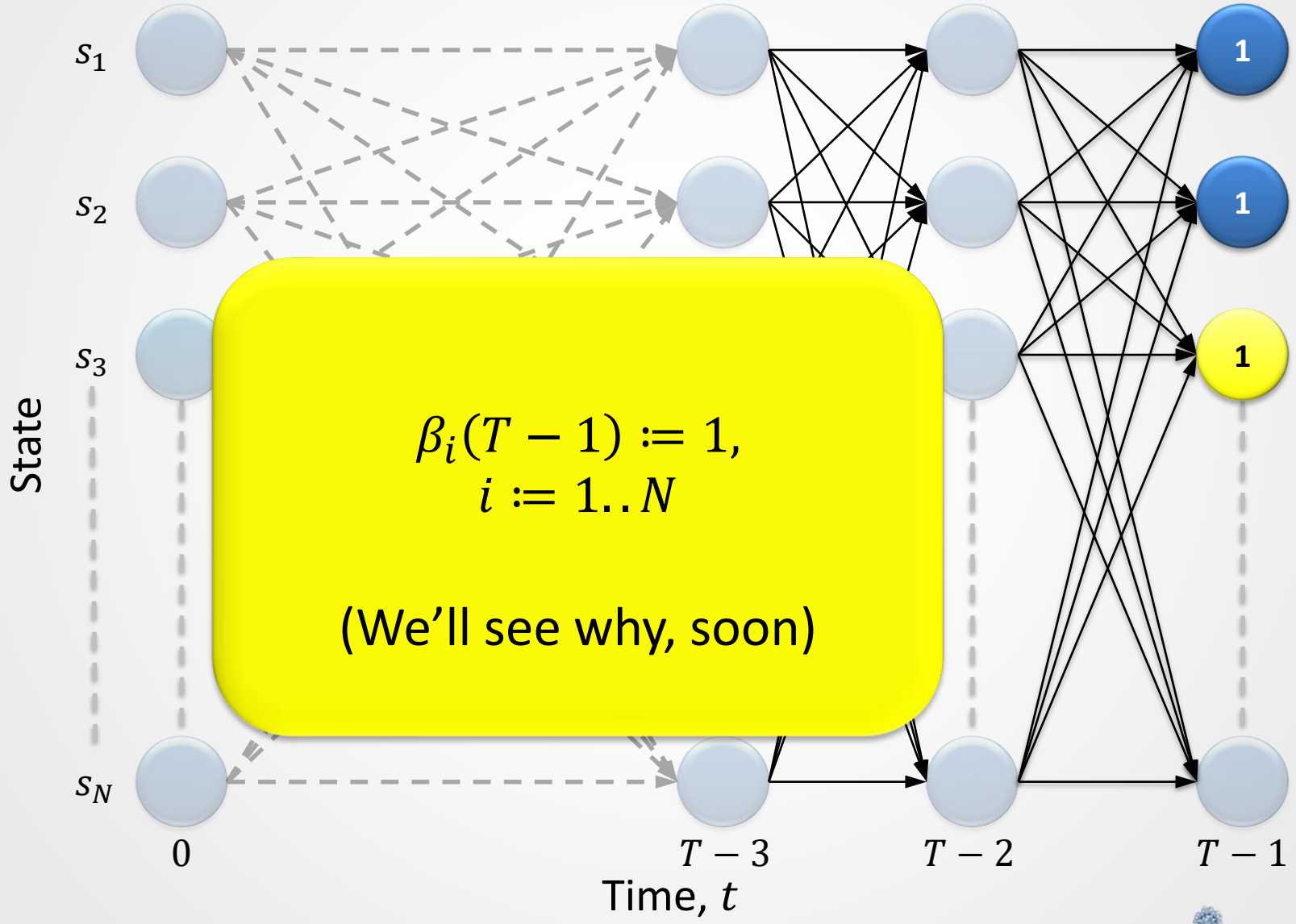
$$\beta_i(t) = P(\sigma_{t+1:T} | q_t = s_i; \theta)$$

  which is computed by summing probabilities on **outgoing** arcs **from** that node.

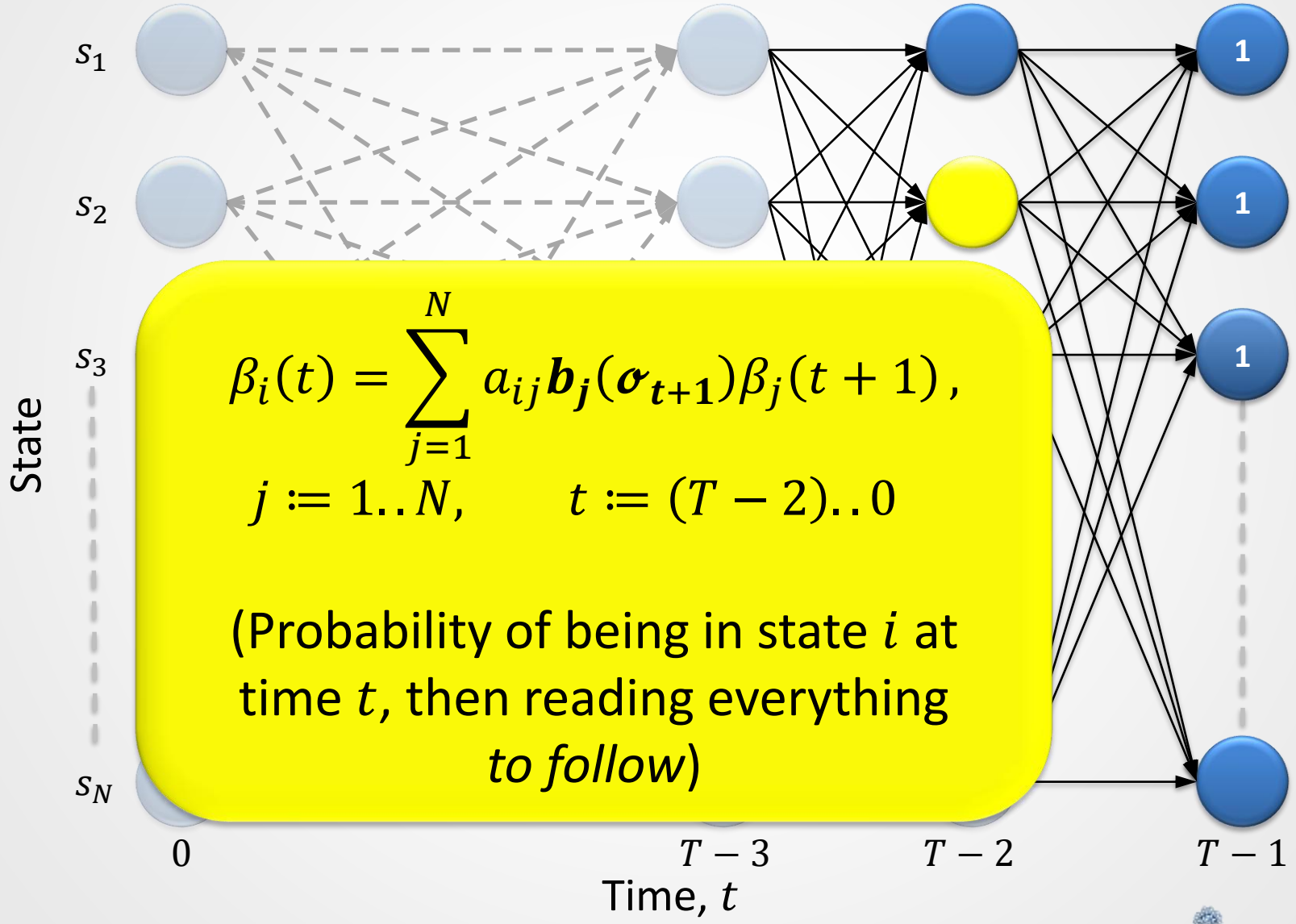
$\beta_i(t)$ is the probability of starting in state i at time t then observing everything that comes thereafter.

- The trellis is computed **right-to-left** and **top-to-bottom**.

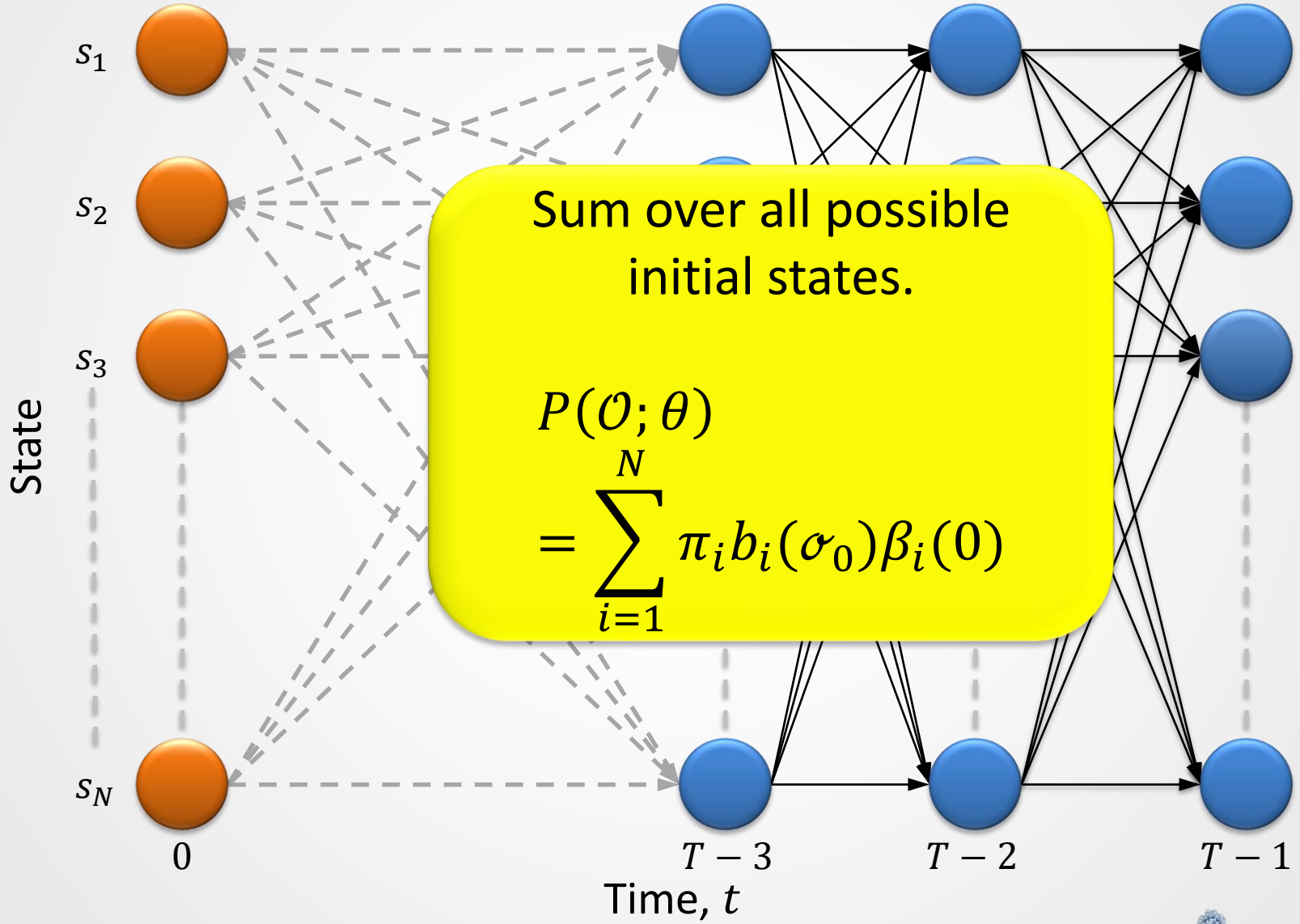
Step 1: Backward initialization



Step 2: Backward induction



Step 3: Backward conclusion



The Backward procedure

- Initialization

$$\beta_i(T-1) = 1, \qquad i := 1..N$$

- Induction

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_j(t+1), \qquad i := 1..N, \quad t := T-1..0$$

- Conclusion

$$P(O;\theta) = \sum_{i=1}^{N} \pi_i b_i(o_0) \beta_i(0)$$

The Backward procedure – so what?

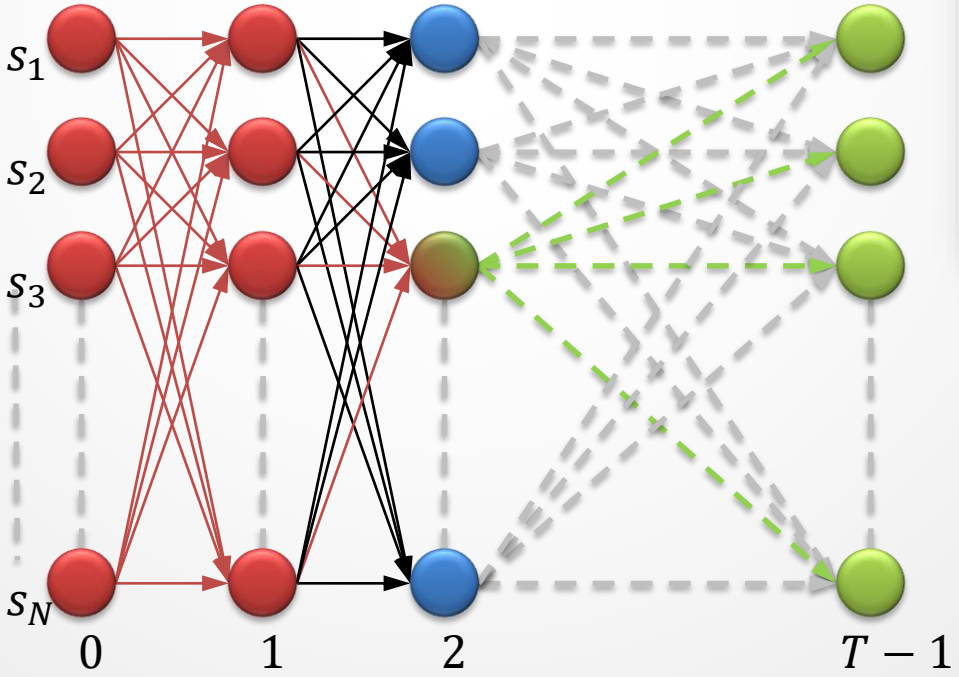
- The **combination** of Forward and Backward procedures will be vital for solving parameter re-estimation, i.e., **training**.

- Generally, we can **combine** α and β at any point in time to represent the probability of an **entire** observation sequence…

Combining α and β

$$P(\mathcal{O}, q_t = i; \theta) = \alpha_i(t)\beta_i(t)$$

$$\therefore P(\mathcal{O}; \theta) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$$



This requires the current word to be incorporated by $\alpha_i(t)$, but **not** $\beta_i(t)$.

This isn't merely for fun – it will soon become useful…

Reading

- (optional) Manning & Schütze: Section 9.2—9.4.1
  - Note that they use another formulation…

- Rabiner, L. (1990) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: *Readings in speech recognition*. Morgan Kaufmann.
  (posted on course website)

- Optional software:
  - Hidden Markov Model Toolkit (http://htk.eng.cam.ac.uk/)
  - Sci-kit's HMM (http://scikit-learn.sourceforge.net/stable/modules/hmm.html)