9: Viterbi Algorithm for HMM Decoding

Machine Learning and Real-world Data

Andreas Vlachos
(slides adapted from Simone Teufel)

Department of Computer Science and Technology
University of Cambridge

Last session: estimating parameters of an HMM

- The dishonest casino, dice edition.
- Two hidden states: L (loaded dice), F (fair dice).
- You don't know which dice is currently in use. You can only observe the numbers that are thrown.
- You estimated transition and emission probabilities (Problem 1 from last time).
- We are now turning to Problem 4.
- We want the HMM to find out when the fair dice was out, and when the loaded dice was out.
- We need to write a decoder.

Decoding: finding the most likely path

- Definition of decoding: Finding the most likely hidden state sequence X that explains the observation O given the HMM parameters $\mu = (A, B)$.

$$\begin{aligned}\hat{X} &= \operatorname*{argmax}_X P(X, O|\mu) \\ &= \operatorname*{argmax}_X P(O|X, B)P(X|A) \\ &= \operatorname*{argmax}_{X_1 \ldots X_T} \prod_{t=1}^{T} P(O_t|X_t, B)P(X_t|X_{t-1}, A)\end{aligned}$$

- Search space of possible state sequences X is O(N^T); too large for brute force search.

Viterbi is a Dynamic Programming Application

(Reminder from Algorithms course)
We can use Dynamic Programming if two conditions apply:

- Optimal substructure property
  - An optimal state sequence $X_1 \dots X_j \dots X_T$ contains inside it the sequence $X_1 \dots X_j$, which is also optimal
- Overlapping subsolutions property
  - If both X_t and X_u are on the optimal path, with $u > t$, then the calculation of the probability for being in state X_t is part of each of the many calculations for being in state X_u.

Viterbi is a Dynamic Programming Application

(Reminder from Algorithms course)
We can use Dynamic Programming if two conditions apply:

- Optimal substructure property
  - An optimal state sequence $X_1 \dots X_j \dots X_T$ contains inside it the sequence $X_1 \dots X_j$, which is also optimal
- Overlapping subsolutions property
  - If both X_t and X_u are on the optimal path, with $u > t$, then the calculation of the probability for being in state X_t is part of each of the many calculations for being in state X_u.

The intuition behind Viterbi

- Here's how we can save ourselves a lot of time.
- Because of the Limited Horizon of the HMM, we don't need to keep a complete record of how we arrived at a certain state.
- For the first-order HMM, we only need to record one previous step.
- Just do the calculation of the probability of reaching each state once for each time step (variable δ).
- Then memoise this probability in a Dynamic Programming table
- This reduces our effort to $O(N^2T)$.
- This is for the first order HMM, which only has a memory of one previous state.

Viterbi: main data structure

- Memoisation is done using a *trellis*.
- A trellis is equivalent to a Dynamic Programming table.
- The trellis is $(N+2) \times (T+2)$ in size, with states j as rows and time steps t as columns.
- Each cell j, t records the Viterbi probability $\delta_j(t)$, the probability of the most likely path that ends in state s_j at time t:

$$\delta_j(t) = \max_{1 \leq i \leq N} [\delta_i(t-1)\, a_{ij}\, b_j(O_t)]$$

- This probability is calculated by maximising over the best ways of going to s_j for each s_i.
- a_{ij}: the transition probability from s_i to s_j
- $b_j(O_t)$: the probability of emitting O_t from destination state s_j

Viterbi algorithm, initialisation

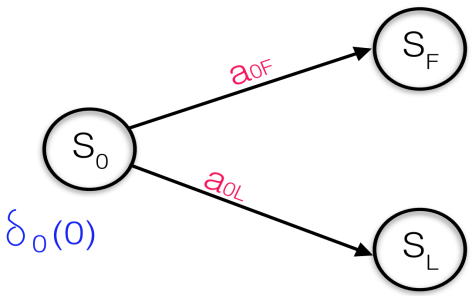
Note: the probability of a state starting the sequence at $t = 0$ is just the probability of it emitting the first symbol.

Viterbi algorithm, initialisation

Viterbi algorithm, initialisation

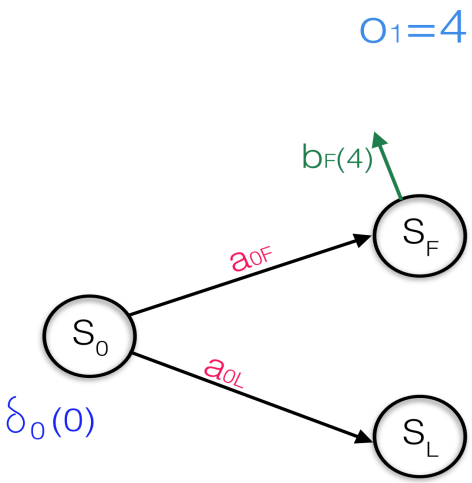
Viterbi algorithm, initialisation

Viterbi algorithm, main step



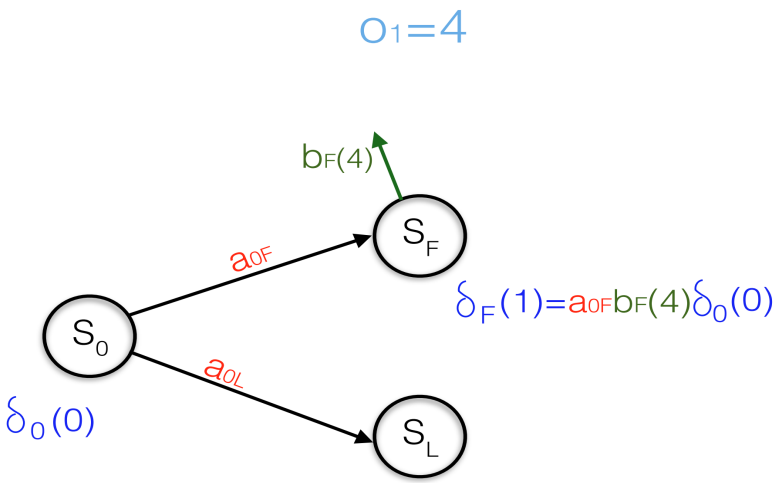
$X_1 = F$ $X = L$ $X = L$

Viterbi algorithm, main step: observation is 4



$X_1 = F$ $X = L$ $X = L$

Viterbi algorithm, main step: observation is 4



$X_1 = F$ $X = L$ $X = L$

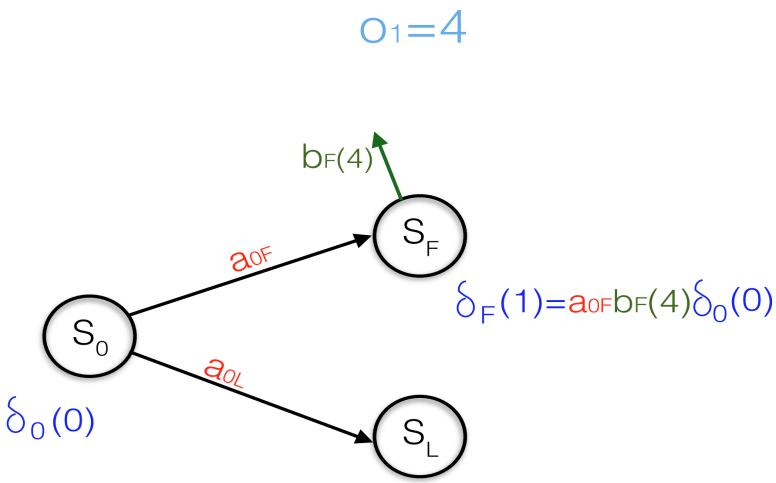
Viterbi algorithm, main step, ψ

- $\psi_j(t)$ is a helper variable that stores the $t-1$ state index i on the highest probability path.

$$\psi_j(t) = \underset{1 \leq i \leq N}{\operatorname{argmax}} [\delta_i(t-1)\, a_{ij}\, b_j(O_t)]$$

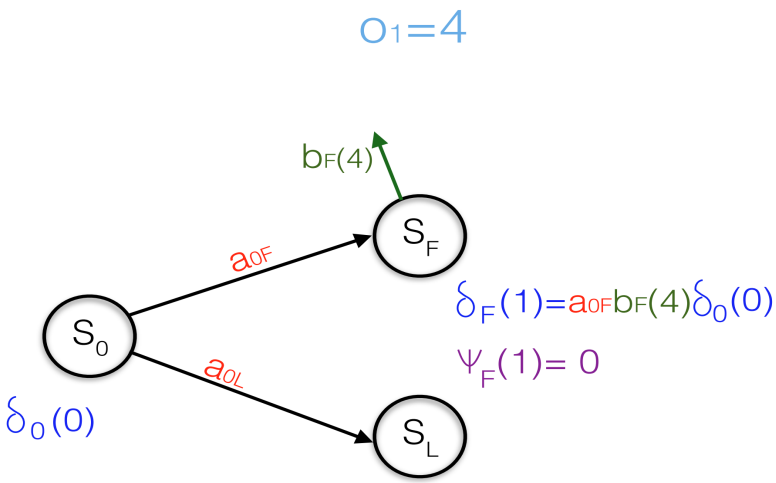
- In the backtracing phase, we will use ψ to find the previous cell/state in the best path.

Viterbi algorithm, main step: observation is 4



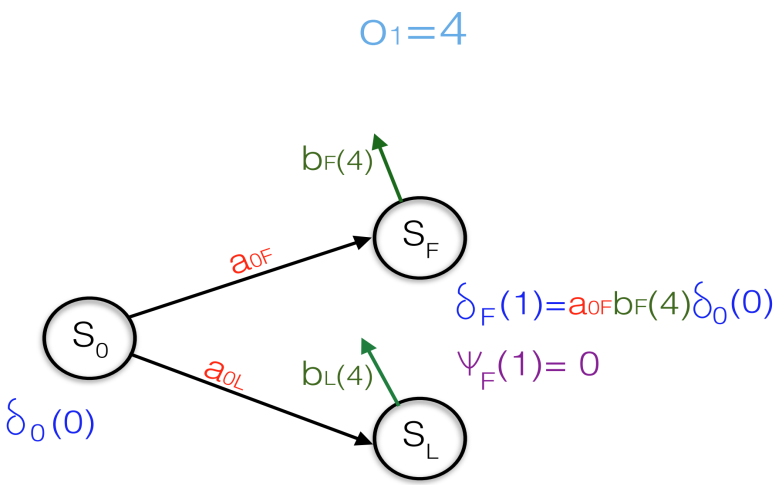
$X_1 = F$ $X = L$ $X = L$

Viterbi algorithm, main step: observation is 4



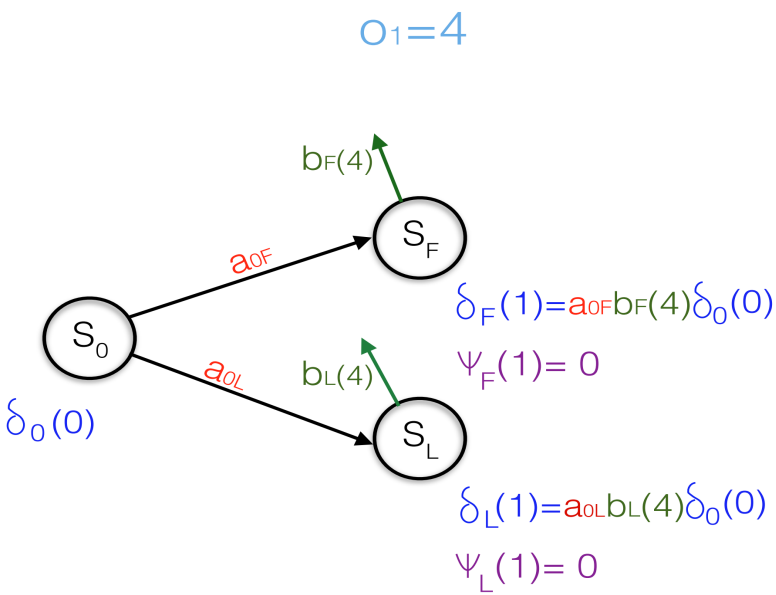
$X_1 = F$ $X = L$ $X = L$

Viterbi algorithm, main step: observation is 4

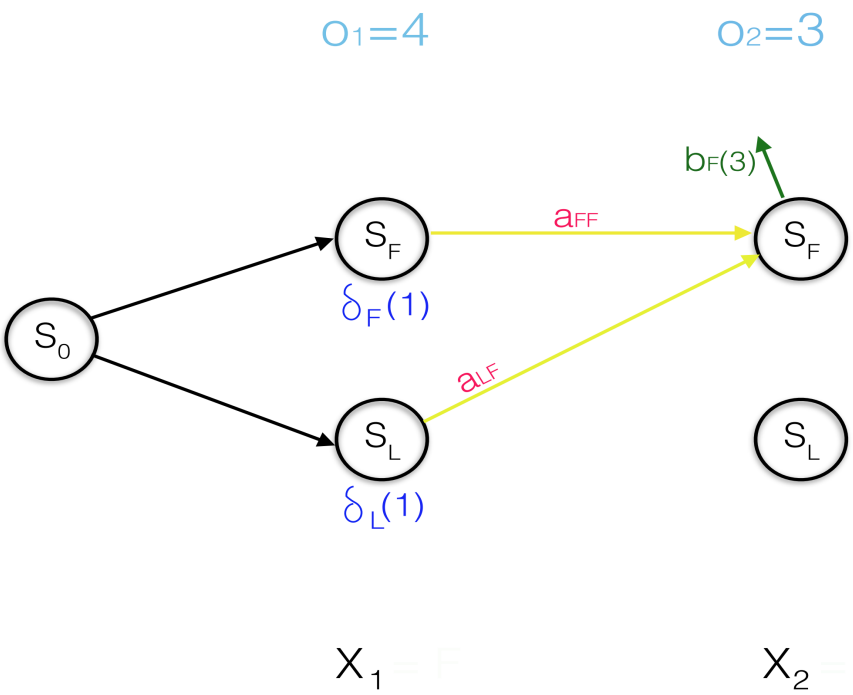


$X_1 = F$ $X = L$ $X = L$

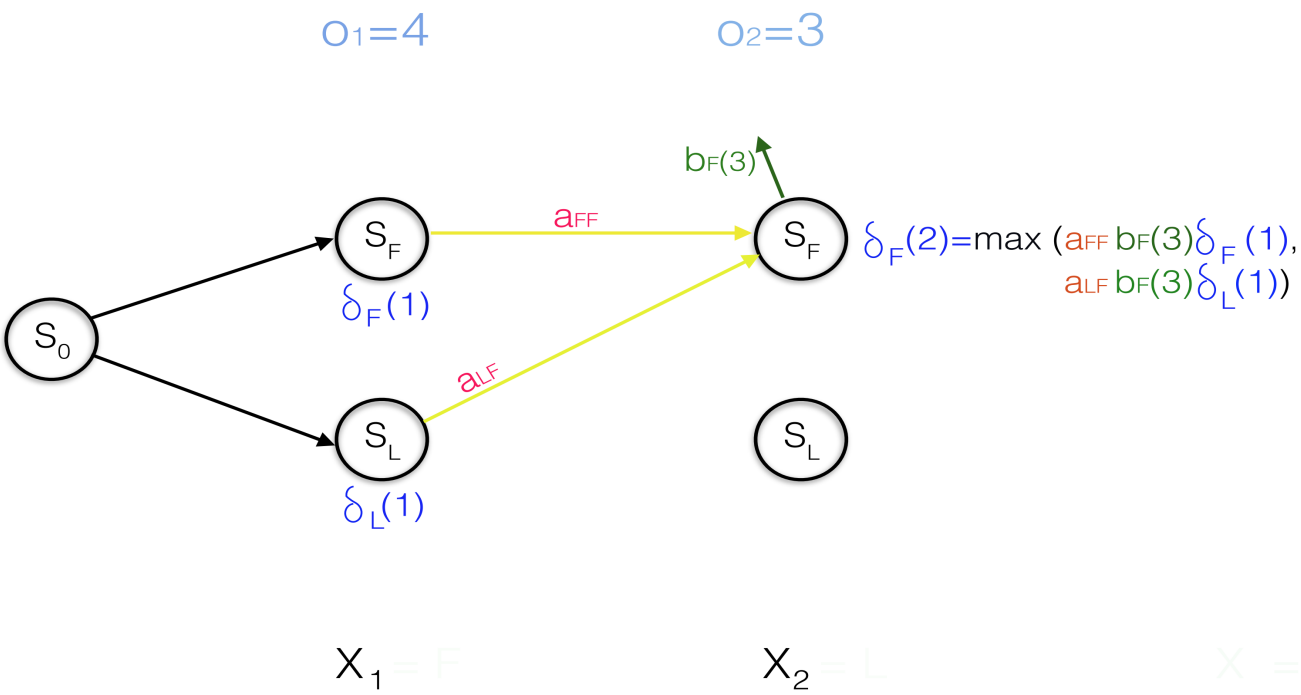
Viterbi algorithm, main step: observation is 4



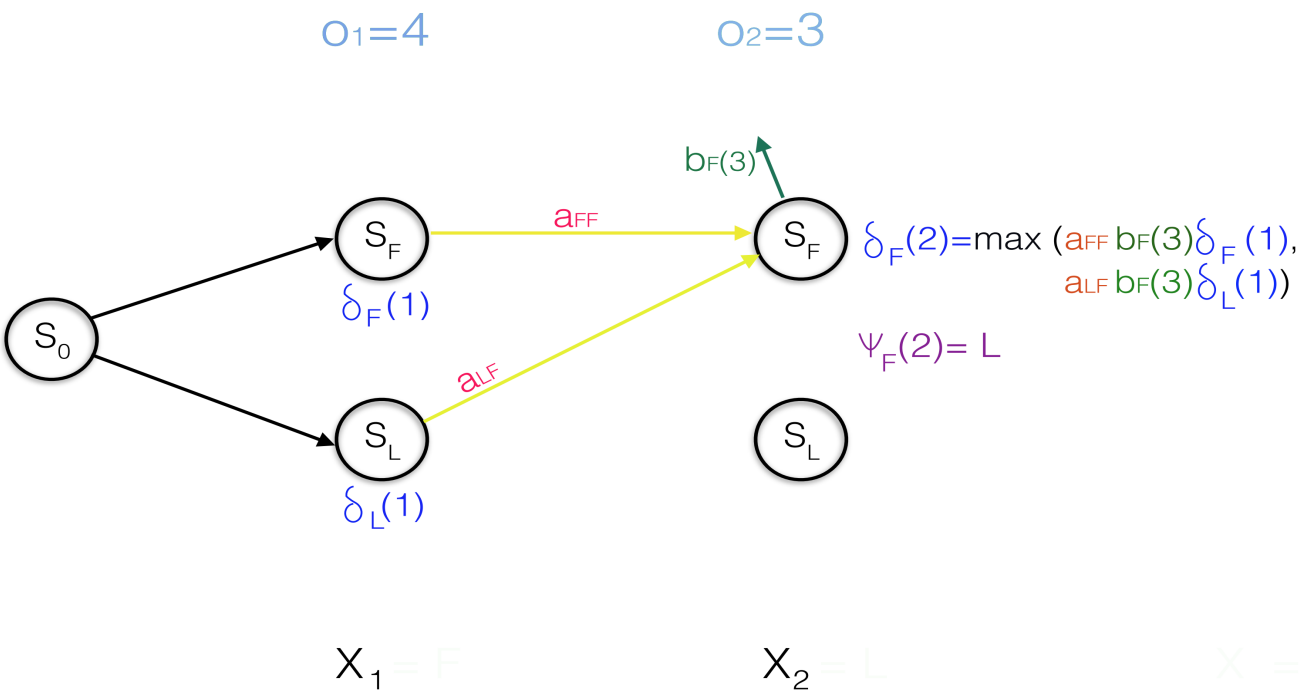
Viterbi algorithm, main step: observation is 3



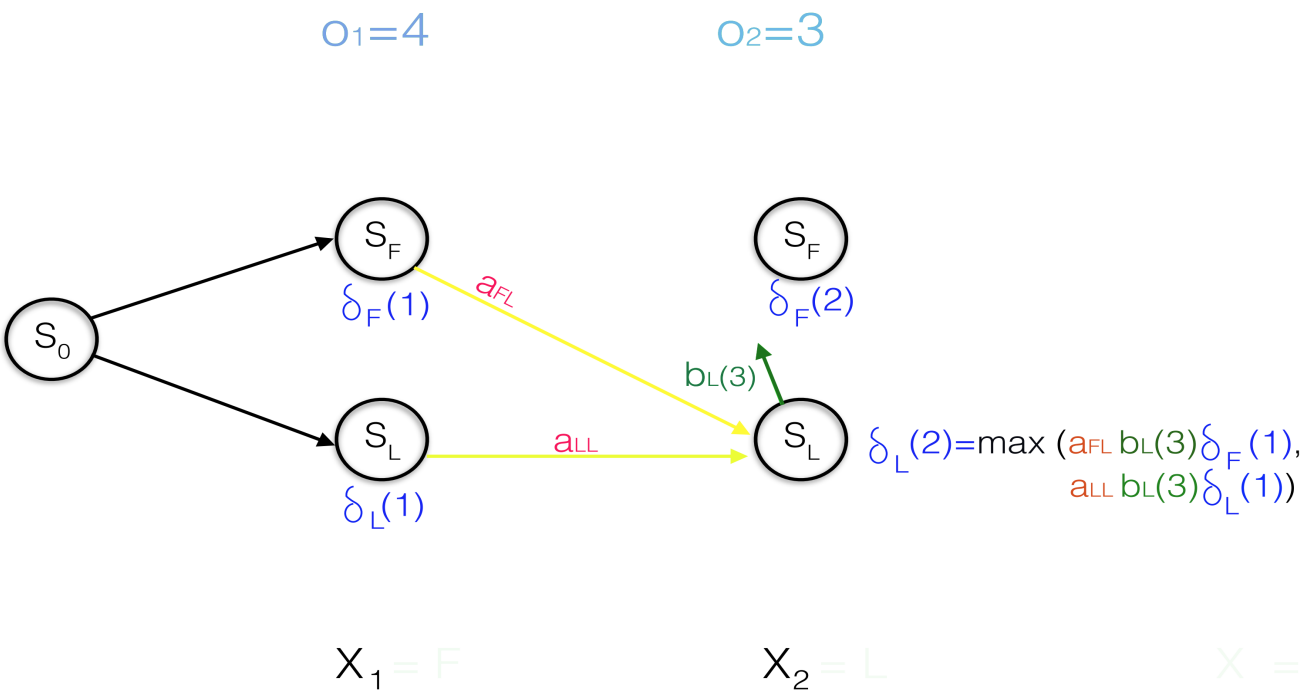
Viterbi algorithm, main step: observation is 3



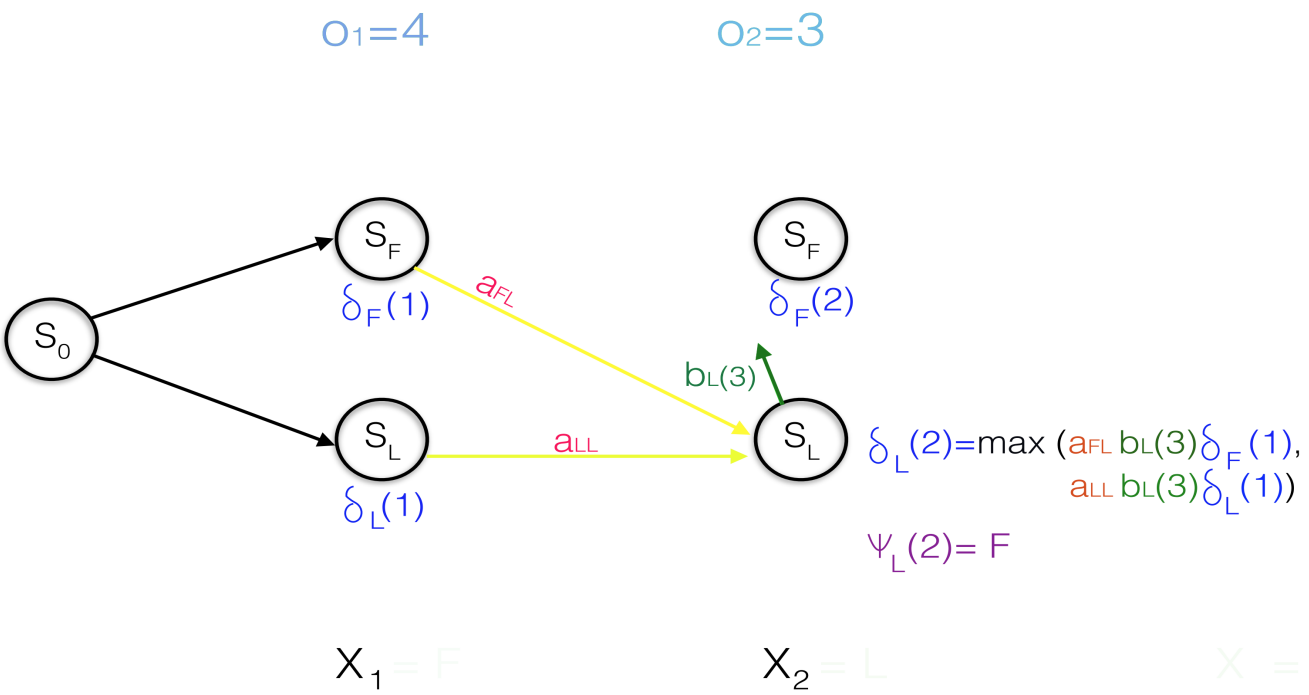
Viterbi algorithm, main step: observation is 3



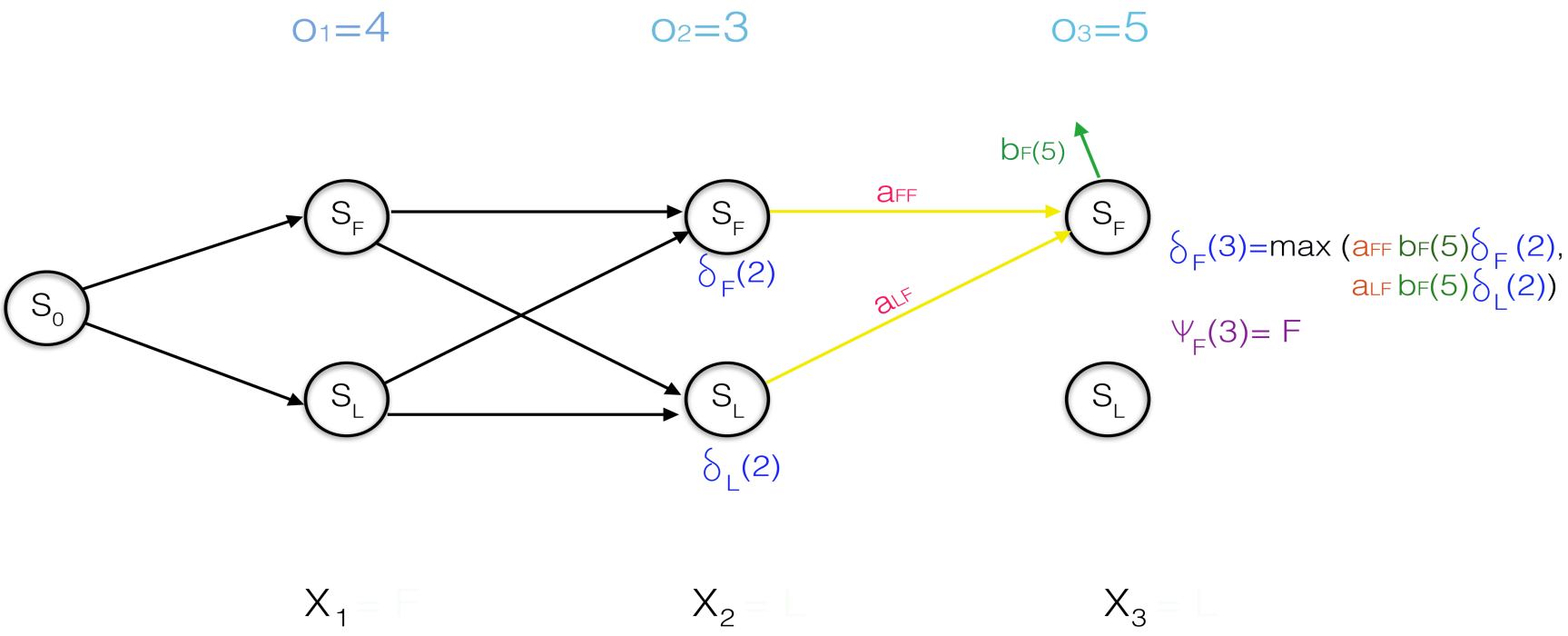
Viterbi algorithm, main step: observation is 3



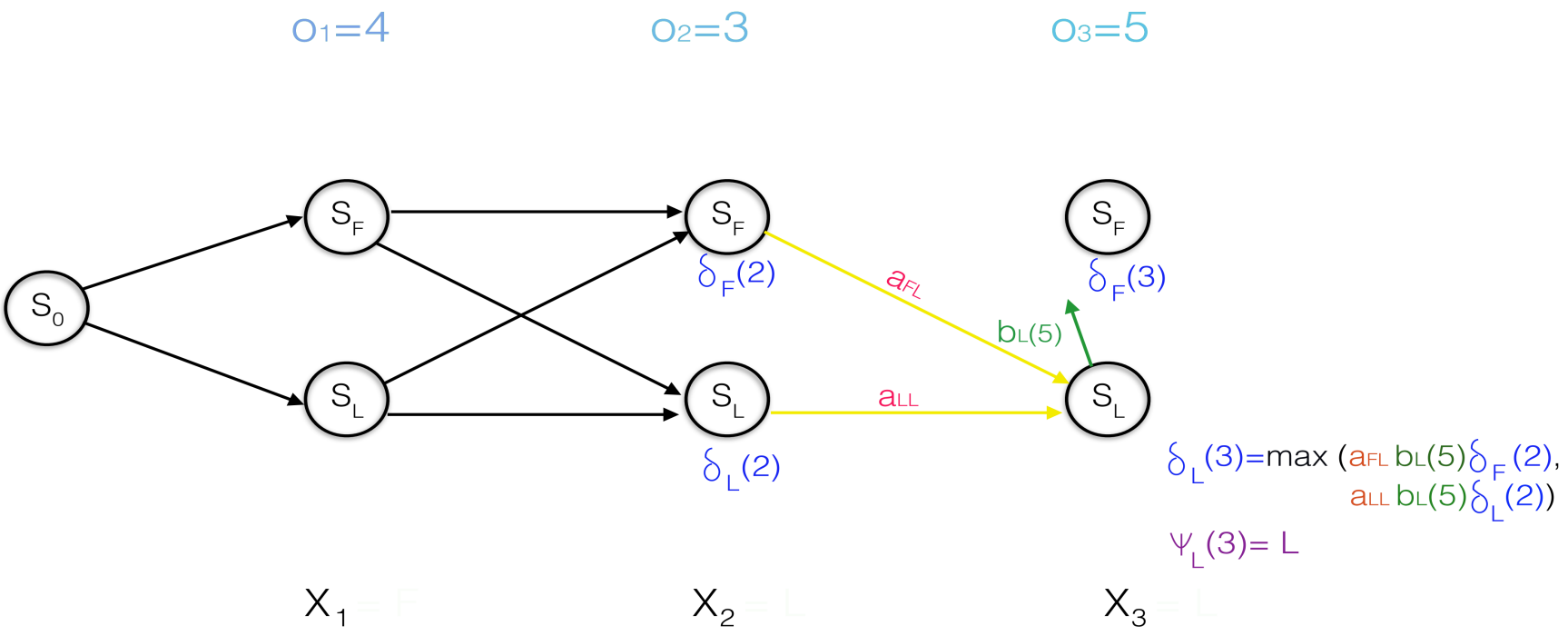
Viterbi algorithm, main step: observation is 3



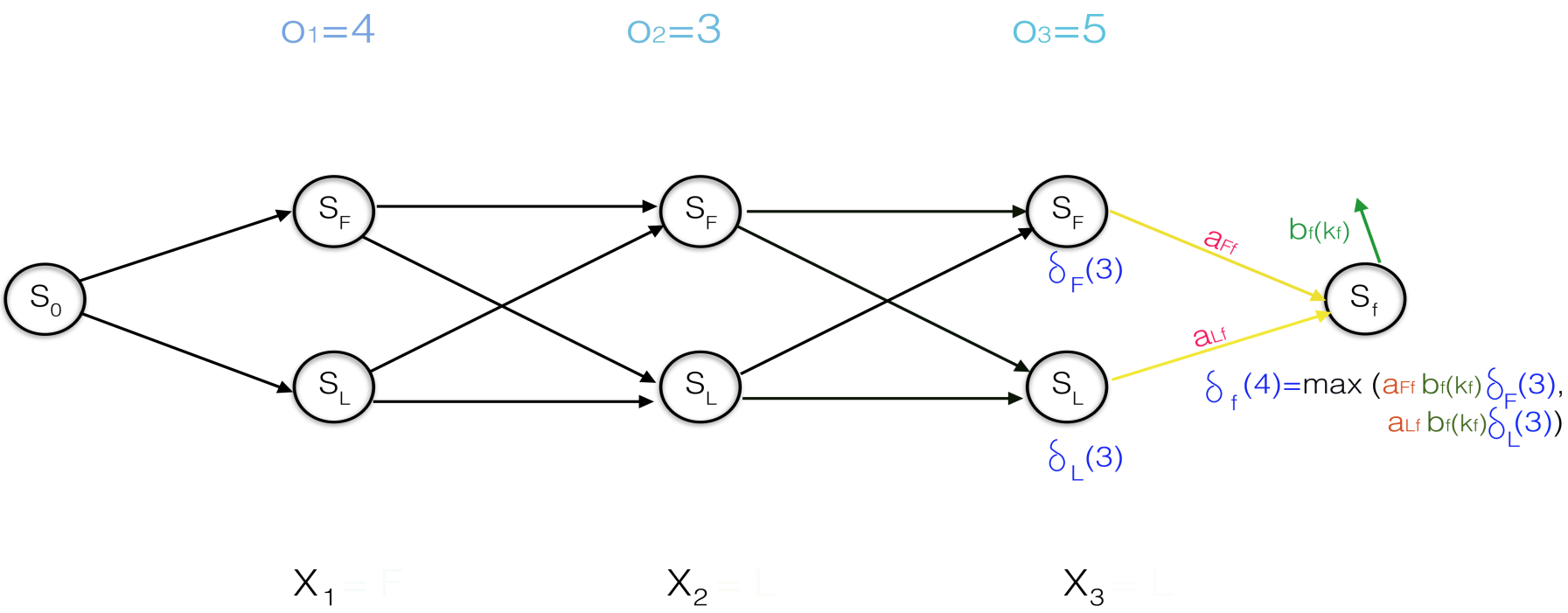
Viterbi algorithm, main step: observation is 5



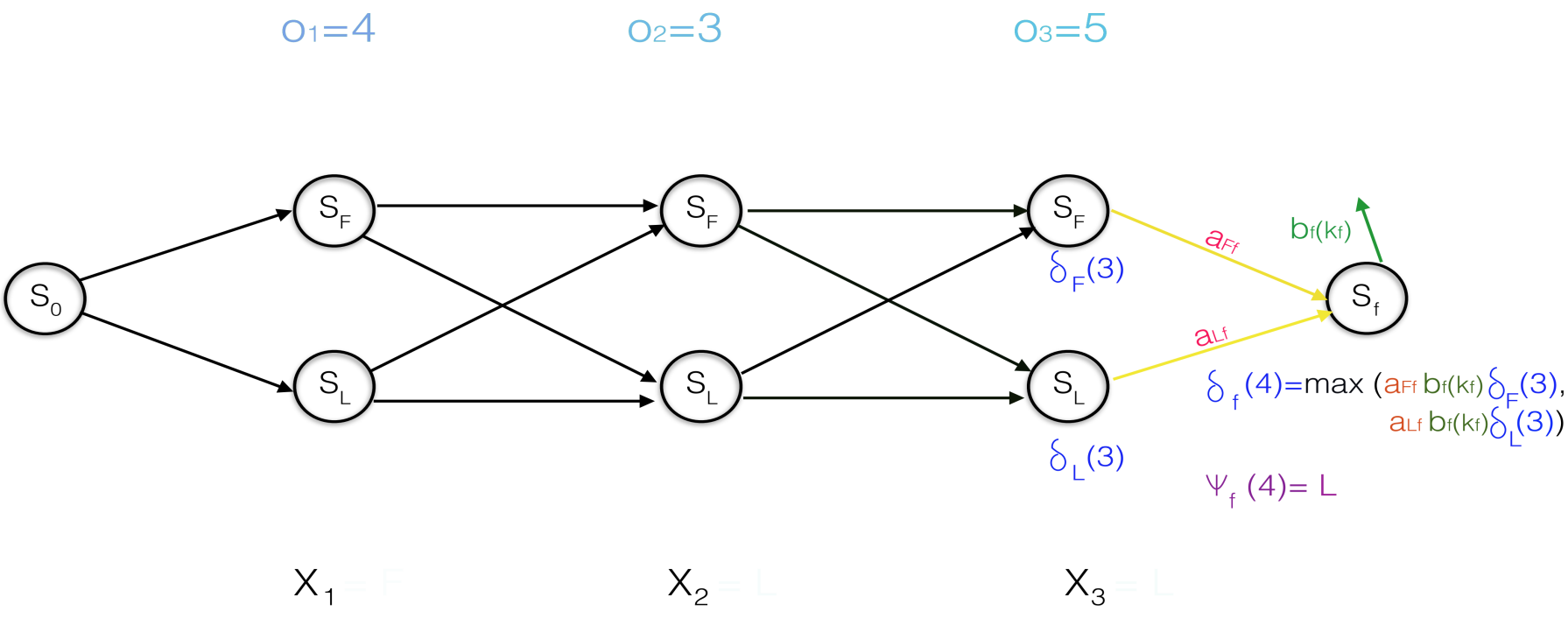
Viterbi algorithm, main step: observation is 5



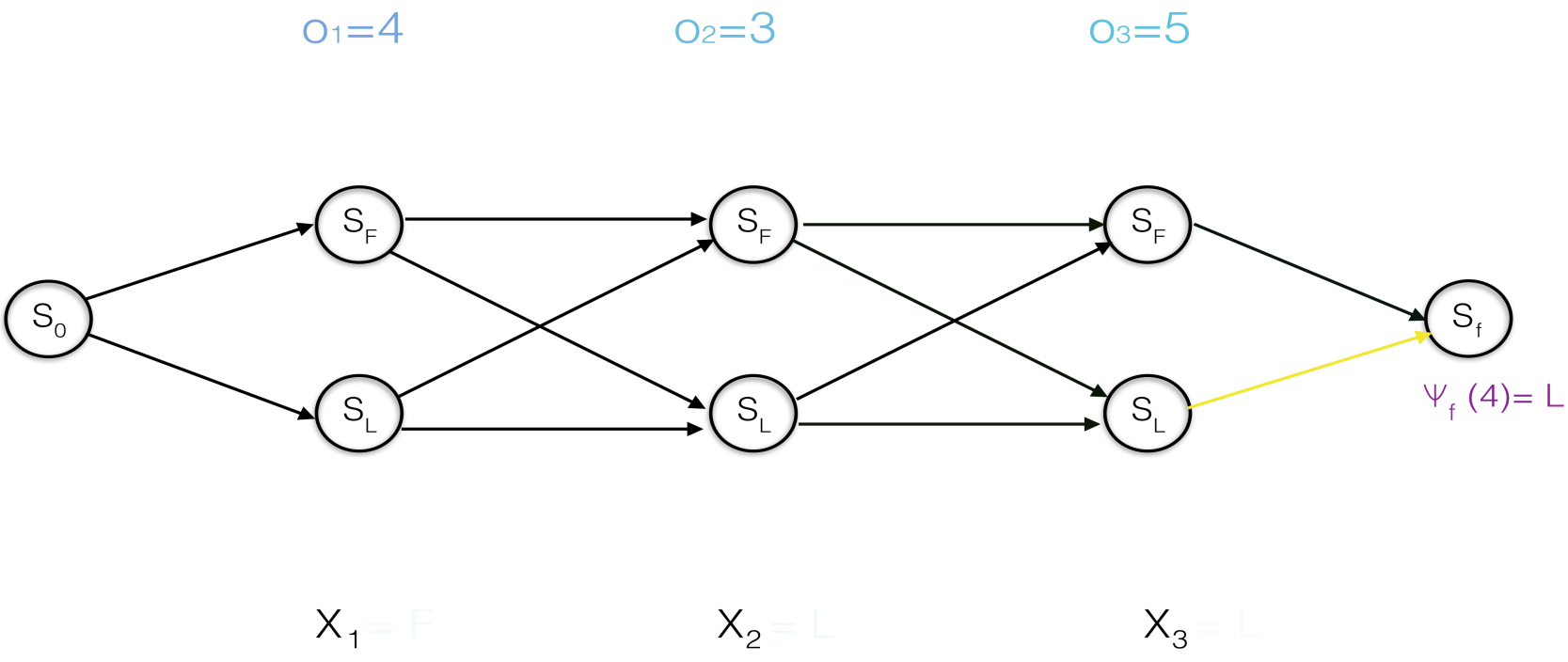
Viterbi algorithm, termination



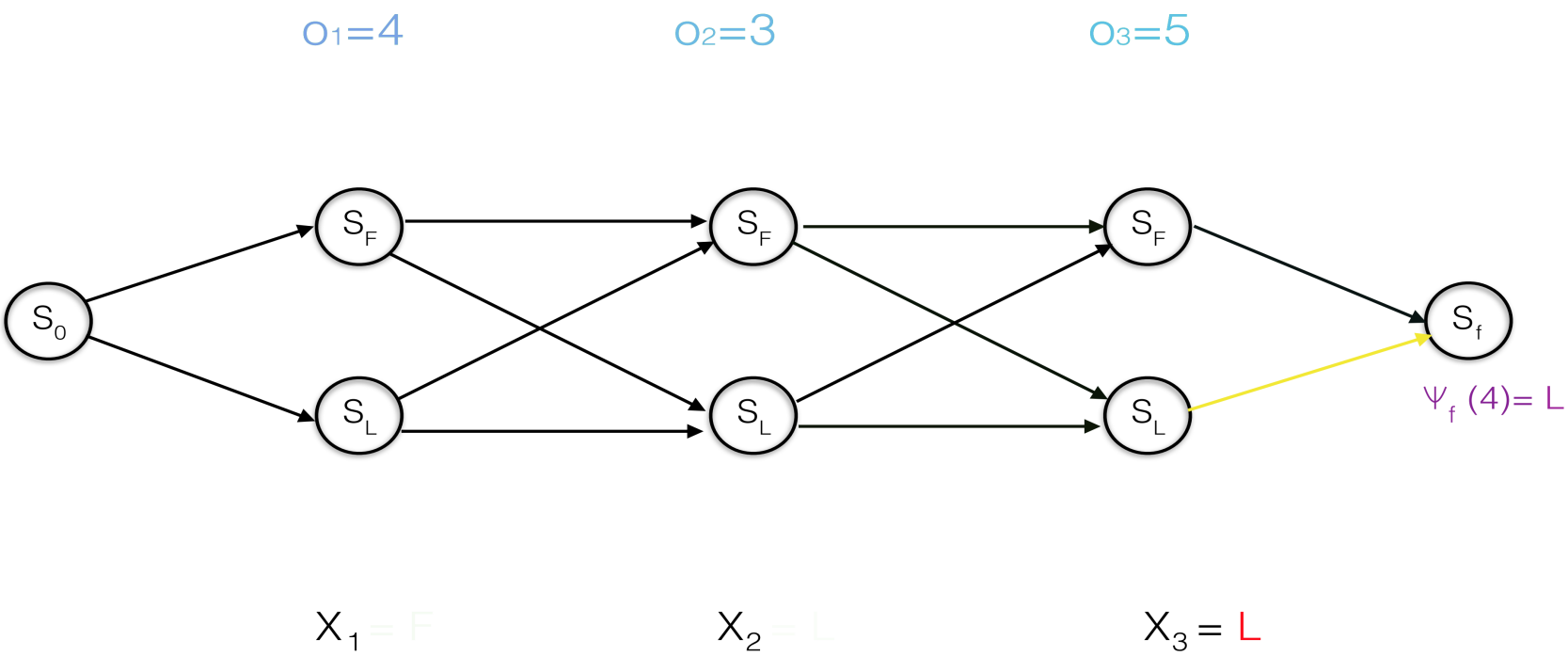
Viterbi algorithm, termination



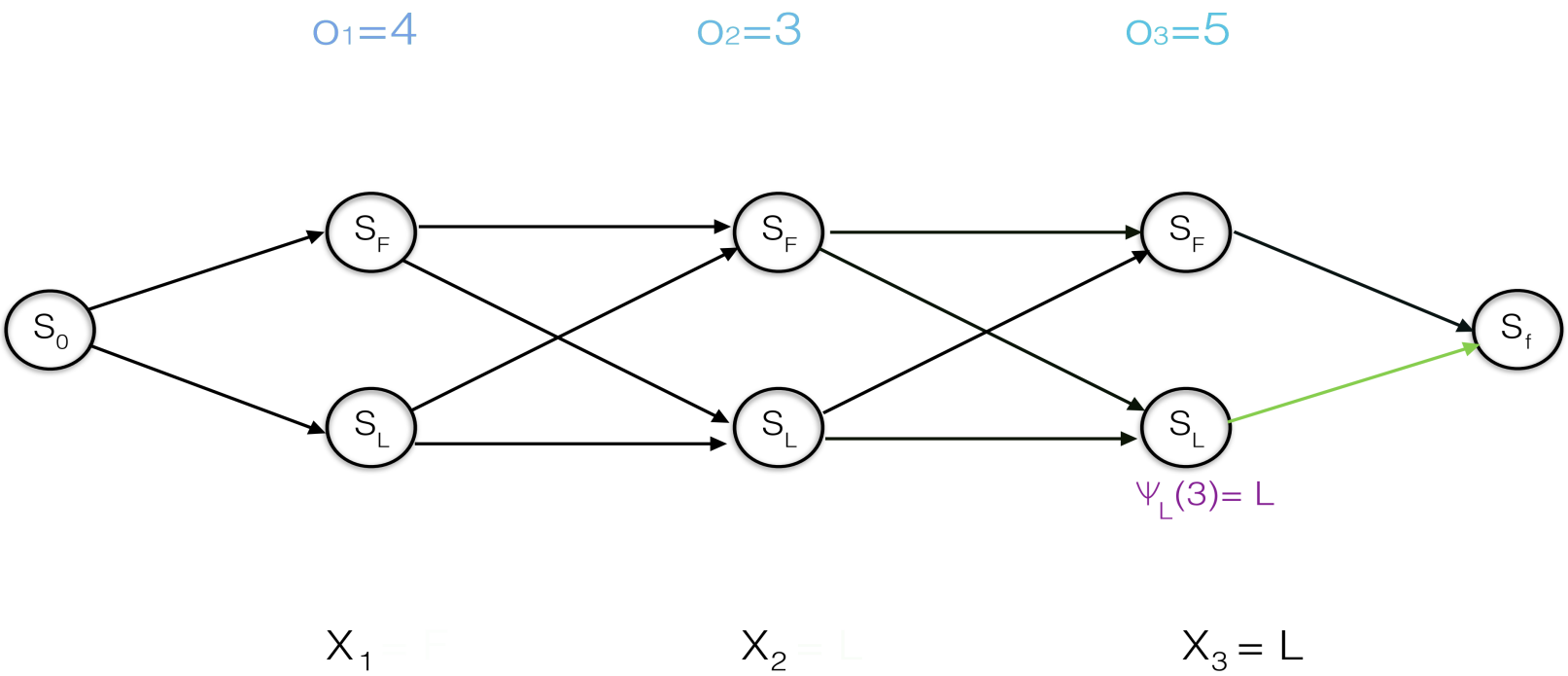
Viterbi algorithm, backtracing



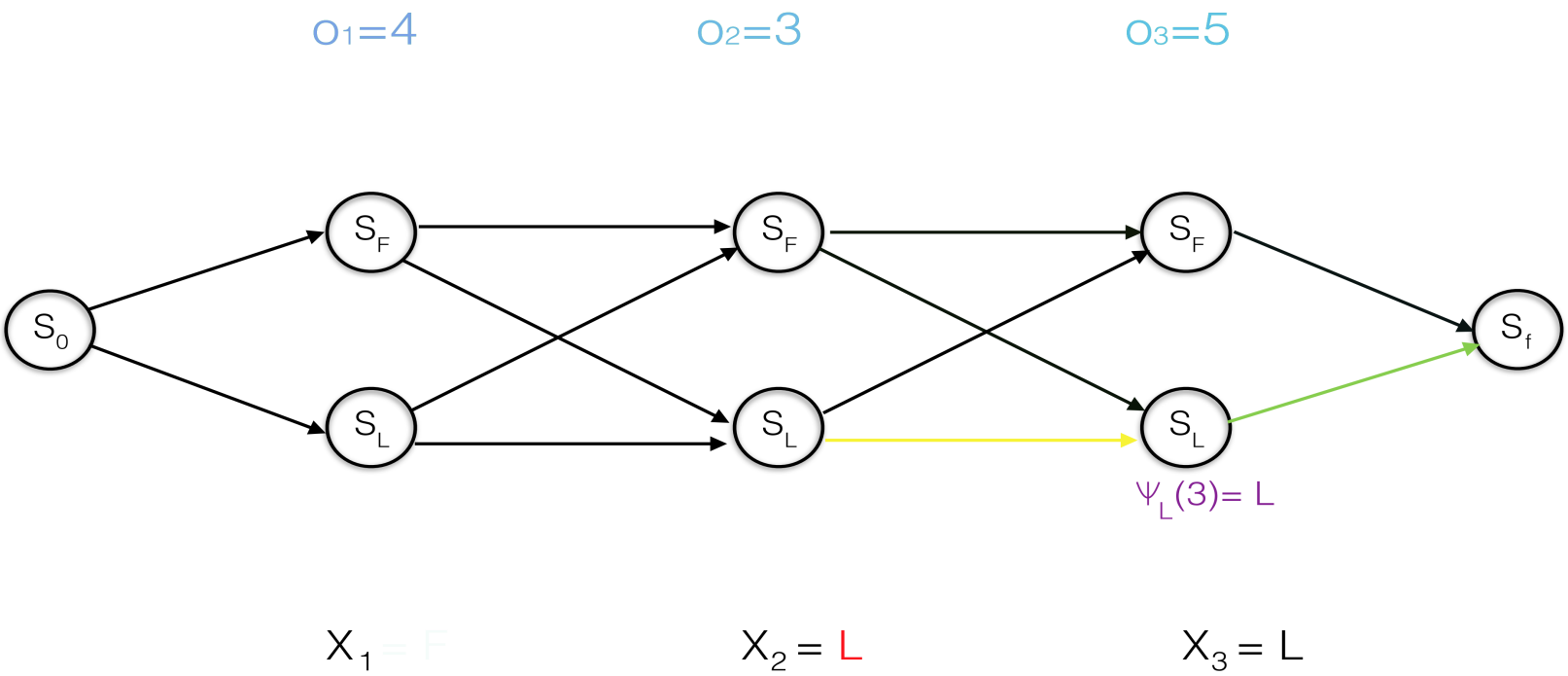
Viterbi algorithm, backtracing



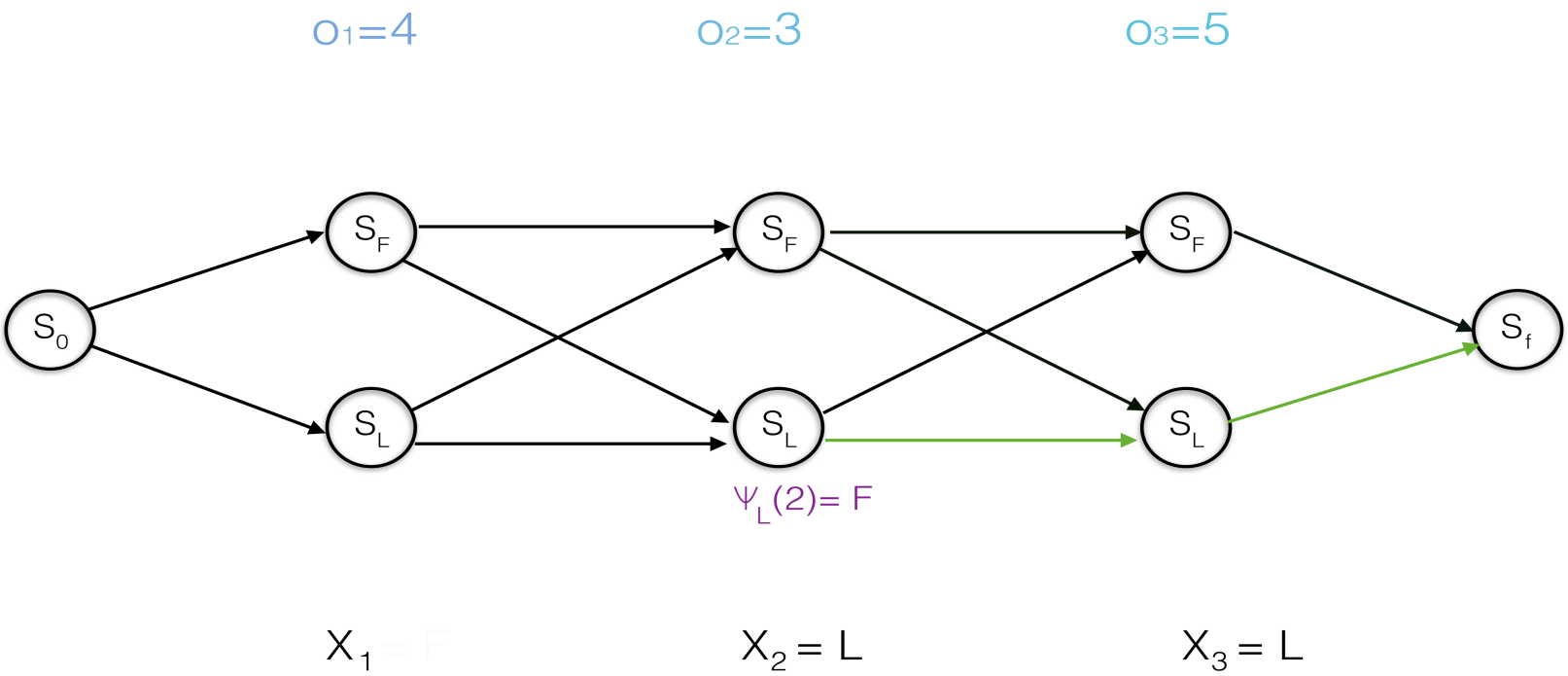
Viterbi algorithm, backtracing



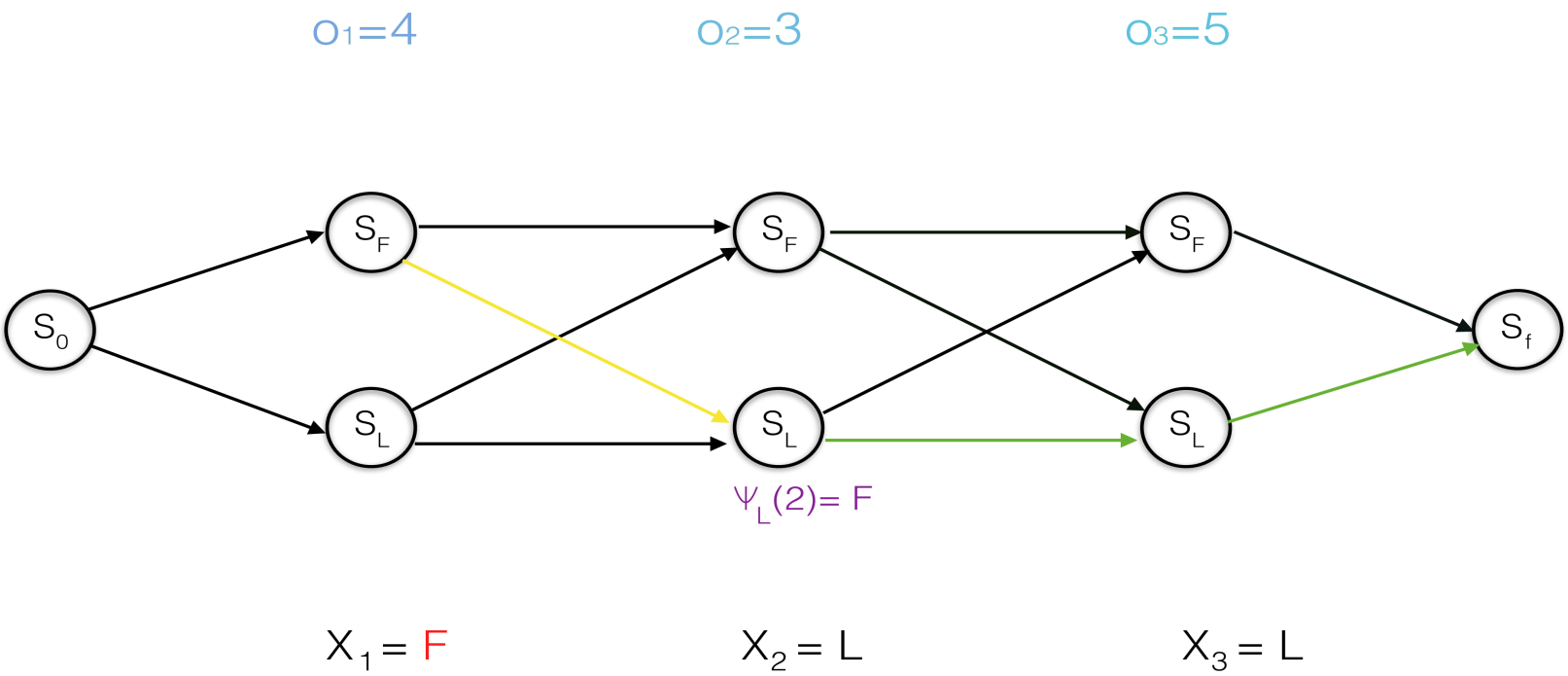
Viterbi algorithm, backtracing



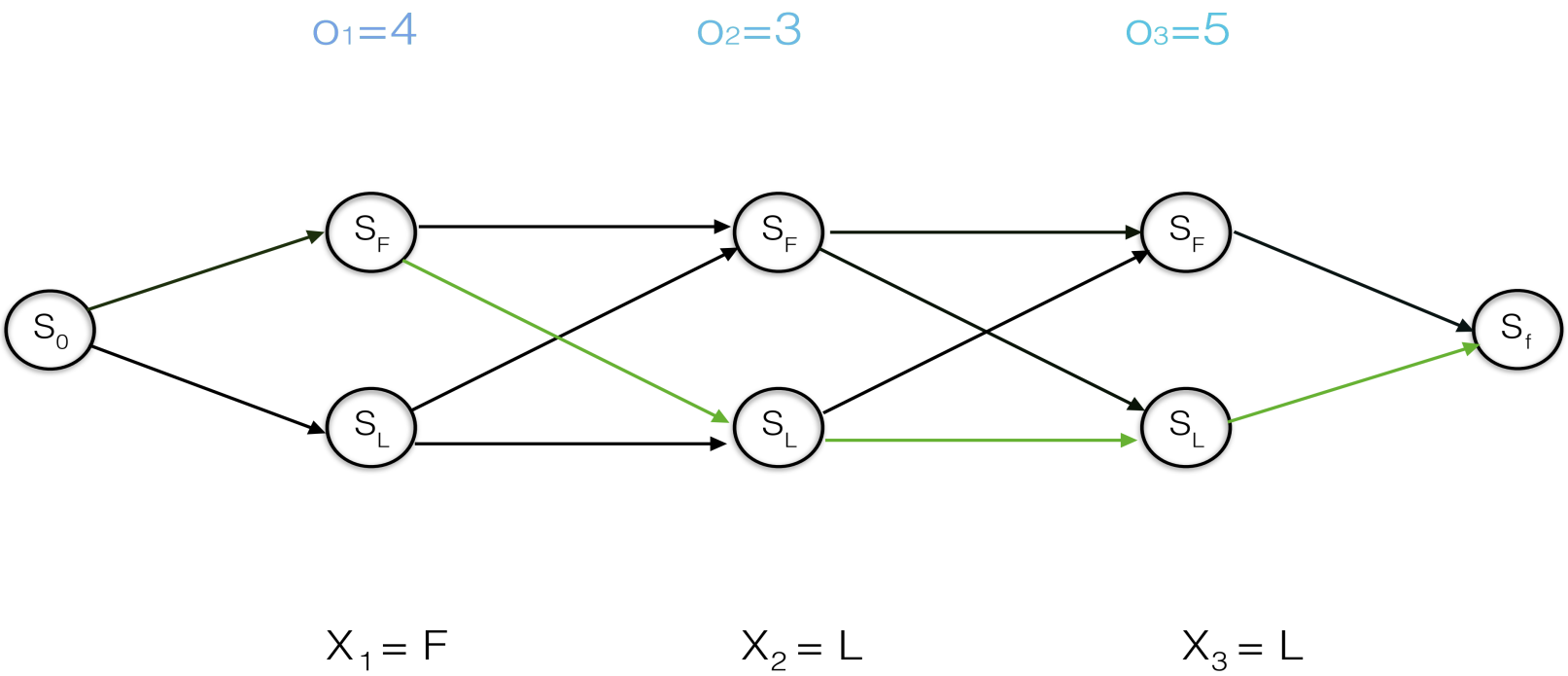
Viterbi algorithm, backtracing



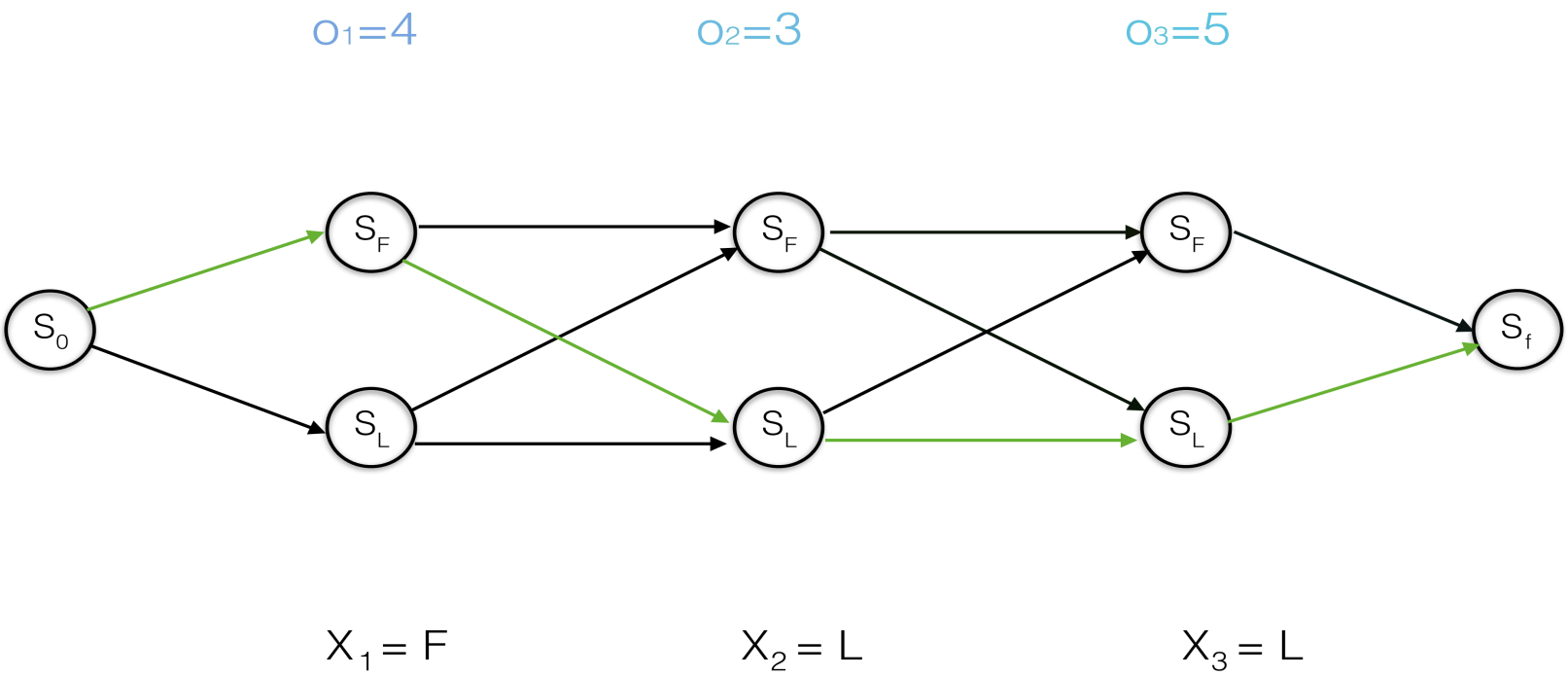
Viterbi algorithm, backtracing



Viterbi algorithm, backtracing



Viterbi algorithm, backtracing



Why is it necessary to keep N states at each time step?

- We have convinced ourselves that it's not necessary to keep more than N ("real") states per time step.
- But could we cut down the table to just a one-dimensional table of T time slots by choosing the probability of the best path overall ending in that time slot, in any of the states?
  - This would be the greedy choice
  - But think about what could happen in a later time slot.
  - You could encounter a zero or very low probability concerning all paths going through your chosen state s_j at time t.
  - Now a state s_k that looked suboptimal in comparison to s_j at time t becomes the best candidate.
  - As we don't know the future, this could happen to any state, so we need to keep the probabilities for each state at each time slot.
- But thankfully, no more.

Precision and Recall

- So far, we have measured system success in accuracy or agreement in Kappa.
- But sometimes it's only one type of instances that we find interesting.
- We don't want a summary measure that averages over interesting and non-interesting instances, as accuracy does.
- In those cases, we use precision, recall and F-measure.
- These metrics are imported from the field of information retrieval, where the difference (in numbers) beween interesting and non-interesting examples is particularly high.
- Accuracy doesn't work well when the types of instances are unbalanced.

True positives, false negatives. . .

		System says: L	System says: F	System says: Total
Truth is:	L	TP	FN	TP+FN
	F	FP	TN	FP+TN
	Total	TP+FP	FN+FP	TP+FP+FN+FP

L is the category we are interested in.

- TP are the true positives.
  - The system correctly declared them as positive.
- FN are the false negatives.
  - The system didn't declare them as as a positive, but should have.
- TN are the true negatives.
  - The system didn't declare them as a positive, and was right.
- FP are the false positives.
  - The system declared them as a positive, but shouldn't have.

Precision and Recall

		System says: L	System says: F	System says: Total
Truth is:	L	TP	FN	TP+FN
	F	FP	TN	FP+TN
	Total	TP+FP	FN+TN	TP+FP+FN+TN

- Precision of L: $P_L = \frac{TP}{TP+FP}$
- Recall of L: $R_L = \frac{TP}{TP+FN}$
- F-measure of L: $F_L = \frac{2P_L R_L}{P_L+R_L}$
- Accuracy: $A = \frac{TP+TN}{TP+FP+FN+TN}$

Your task today

Task 8:

- Implement the Viterbi algorithm.
- Run it on the dice dataset and measure precision of L (P_L), recall of L (R_L) and F-measure of L (F_L).

Literature

- Collin's notes: `http://www.cs.columbia.edu/~mcollins/hmms-spring2013.pdf`
- Jurafsky and Martin, 3rd Edition, `https://web.stanford.edu/~jurafsky/slp3/8.pdf`, Chapter 8.4 (but careful, notation!)
- Smith, Noah A. (2004). Hidden Markov Models: All the Glorious Gory Details: `https://www.cs.cmu.edu/~nasmith/papers/smith.tut04a.pdf`