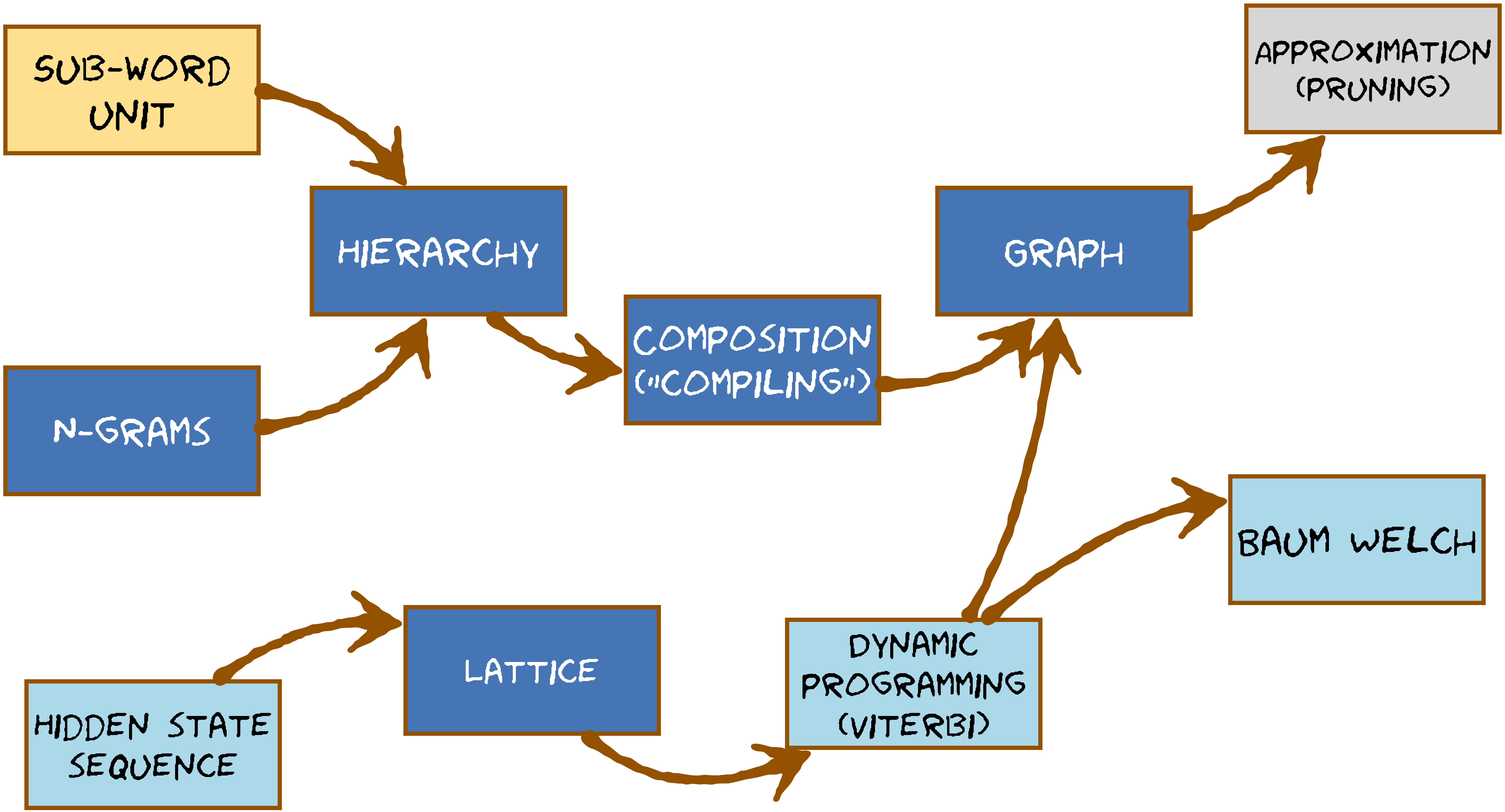
Speech Processing

Simon King
University of Edinburgh

additional class slides for 2020-21

Module 9

Connected speech & HMM training



SUB-WORD UNIT
HIERARCHY
N-GRAMS
COMPOSITION ("COMPILING")
GRAPH
APPROXIMATION (PRUNING)
HIDDEN STATE SEQUENCE
LATTICE
DYNAMIC PROGRAMMING (VITERBI)
BAUM WELCH

Orientation

- Hidden Markov Models (HMMs)
  - Motivated from pattern **matching**
  - Finite state network
  - Gaussian probability density functions
  - A probabilistic **generative** model

What you should already know

- Account for variability using probability density functions (pdfs)
- Use a finite state machine to specify the sequence in which to use the pdfs
- The model is a **generative** model
- The state sequence is unknown to us

  = "**hidden**" from us

so now we call the acoustic features "**observations**"

- state sequence = alignment between the model's states and the observations
- many possible state sequences can generate the same observation sequence
- Viterbi algorithm only finds the **single most likely** state sequence

Problem 1: word sequences

- So far, we have described only isolated word models
- Each model emits an observation sequence

- We assumed that we need to compute separately for each model:
  - the probability that this model emitted the given observation sequence
- and then we would compare those across all our models, choosing the most probable

- <u>This fails to</u>
  - account for prior probability of each word
  - work for word sequences

N-GRAMS

Solution for word sequences: utterance model (language model)

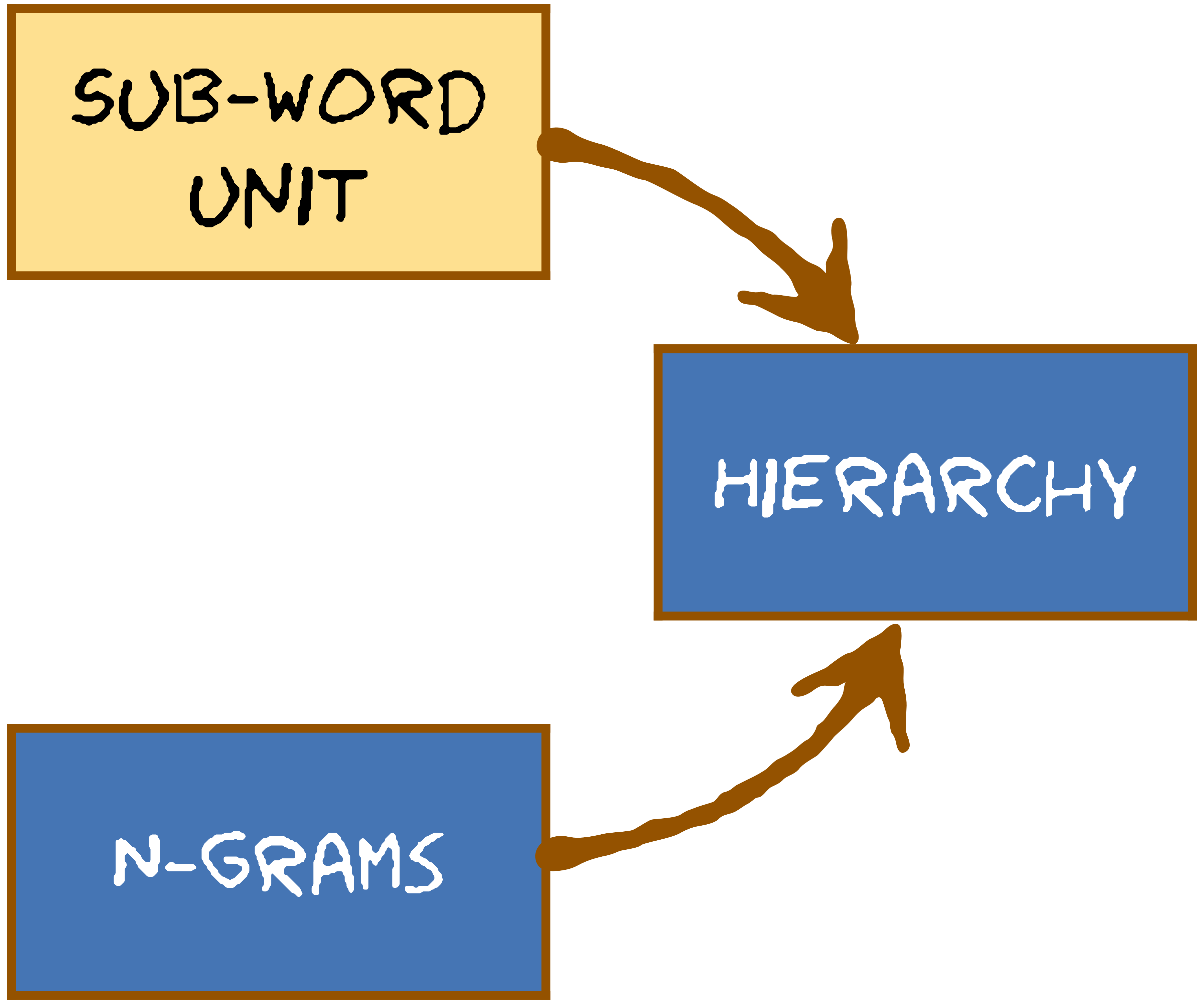
Problem 2: large vocabulary

- Need to handle an arbitrary vocabulary, defined in advance
- Training data may not contain examples of all words
  - so, cannot use whole word models
  - must **create word models** from models of smaller units: phonemes
    - essentially the same solution we used for concatenative speech synthesis

SUB-WORD UNIT

N-GRAMS

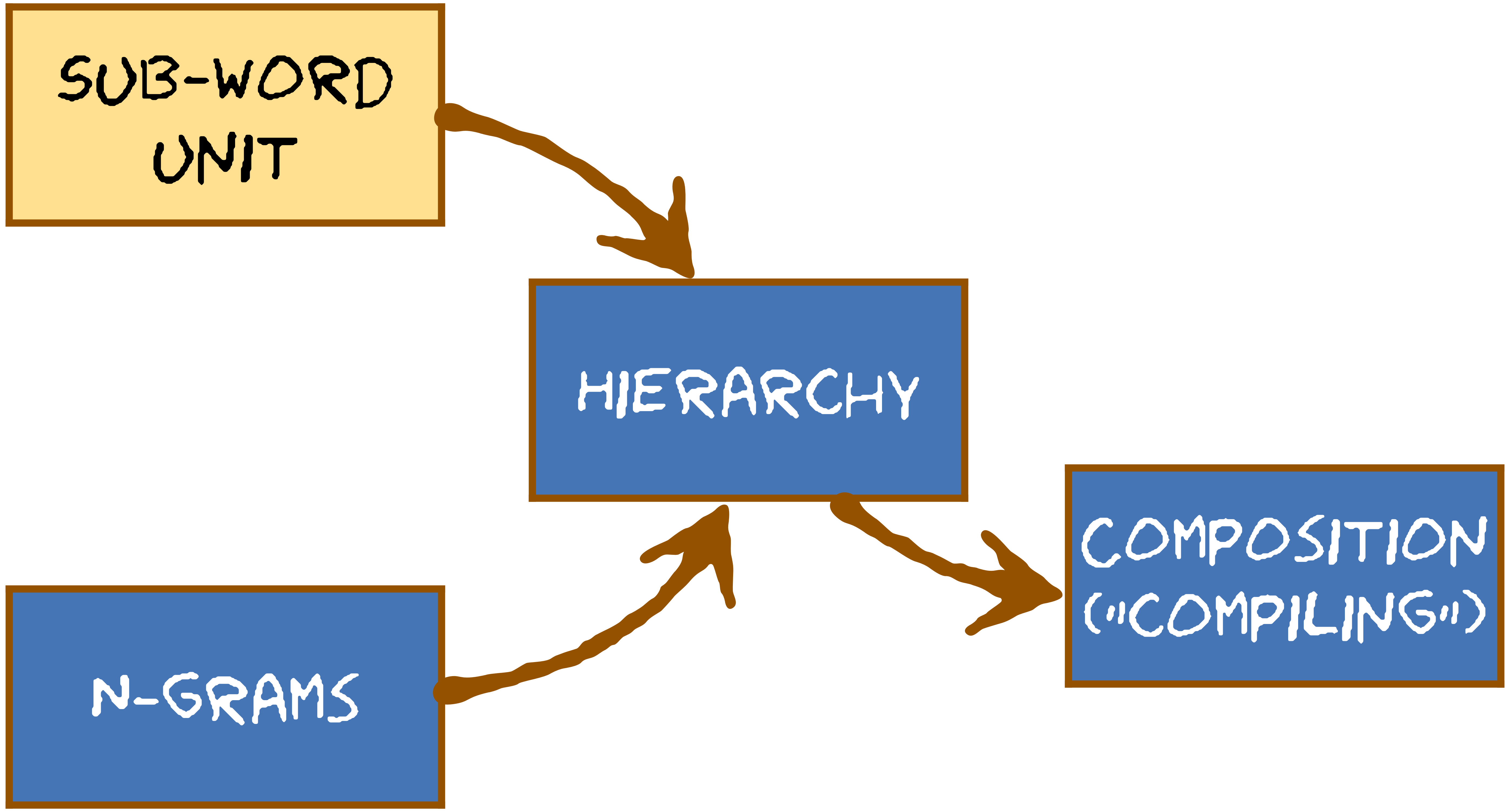
Solution for large vocabulary: word model (dictionary)


SUB-WORD UNIT
HIERARCHY
N-GRAMS

Large-vocabulary connected speech recognition (LVCSR)

- **We simply create a generative model of a complete utterance**
- There is a hierarchy of models
  1. a generative model of an utterance that <u>emits a word sequence</u>
  2. a generative model of each word that <u>emits a phoneme sequence</u>
  3. generative model of each phoneme <u>emits an observation sequence</u>

(In the assignment, we are using whole word models, which directly emit observations. There are no sub-word models.)

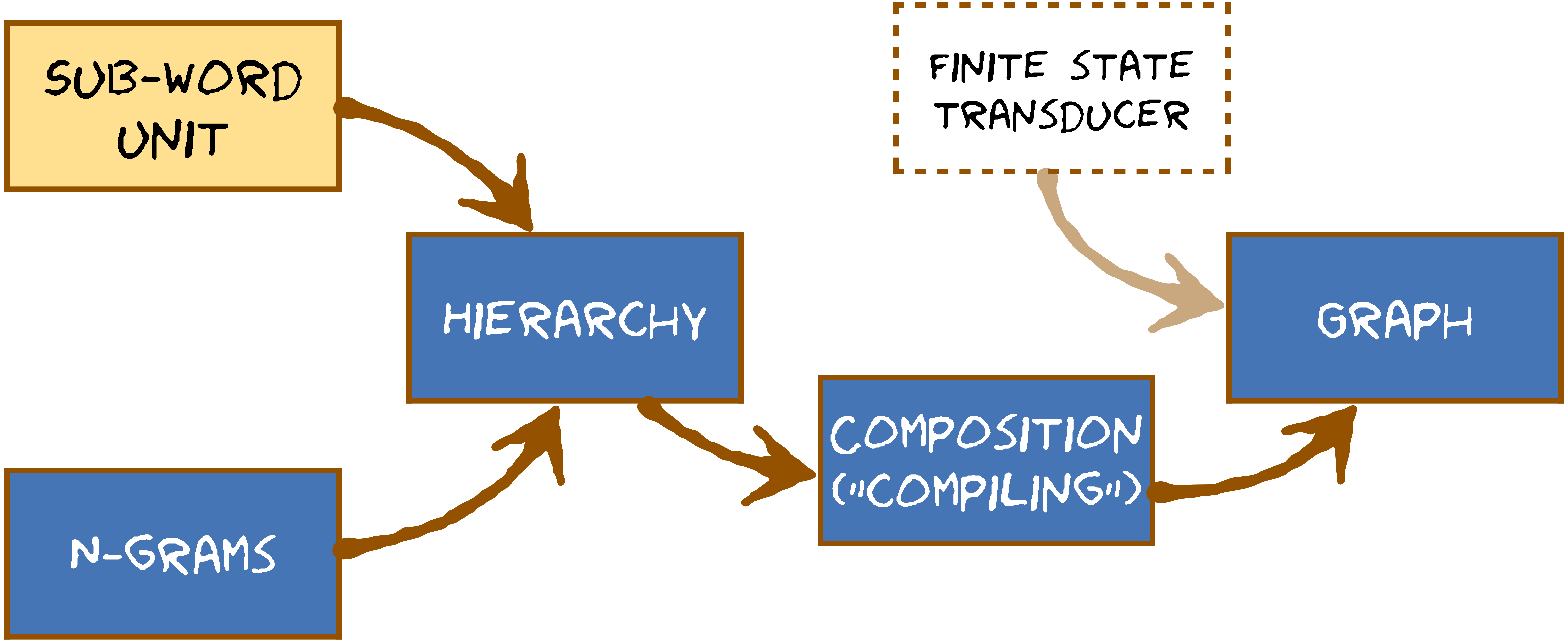

SUB-WORD UNIT
HIERARCHY
N-GRAMS
COMPOSITION ("COMPILING")

Combining models of different linguistic units

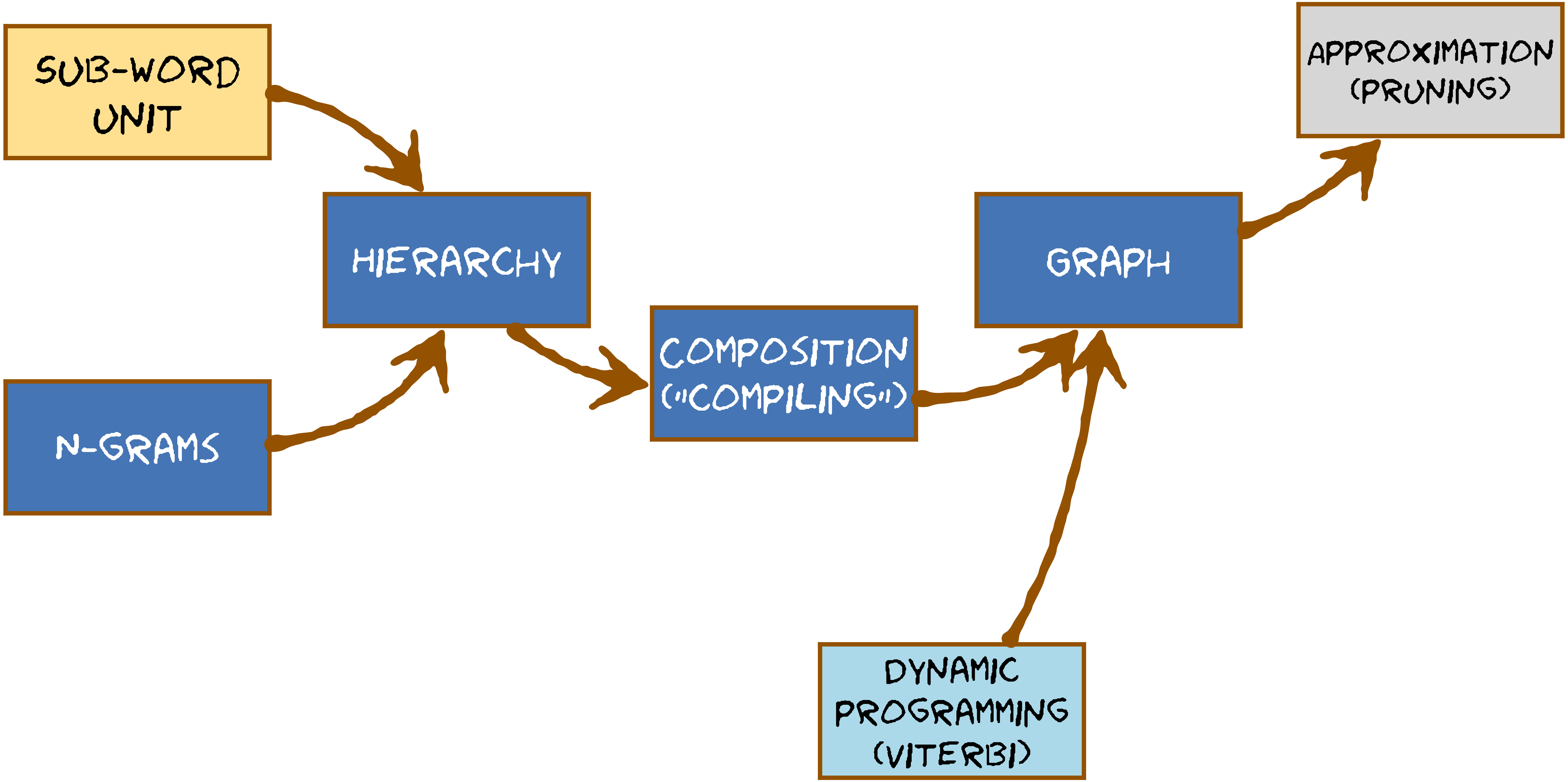
- *If* all of the generative models are finite state
- *then* it is trivial to combine them into a **single finite state model**
  - called "compiling" in HTK, or "composition" in finite-state model terminology

- <u>Terminology:</u>
  - Utterance model = **language model**
  - Word model = **pronunciation dictionary**
  - Sub-word model = **acoustic model**

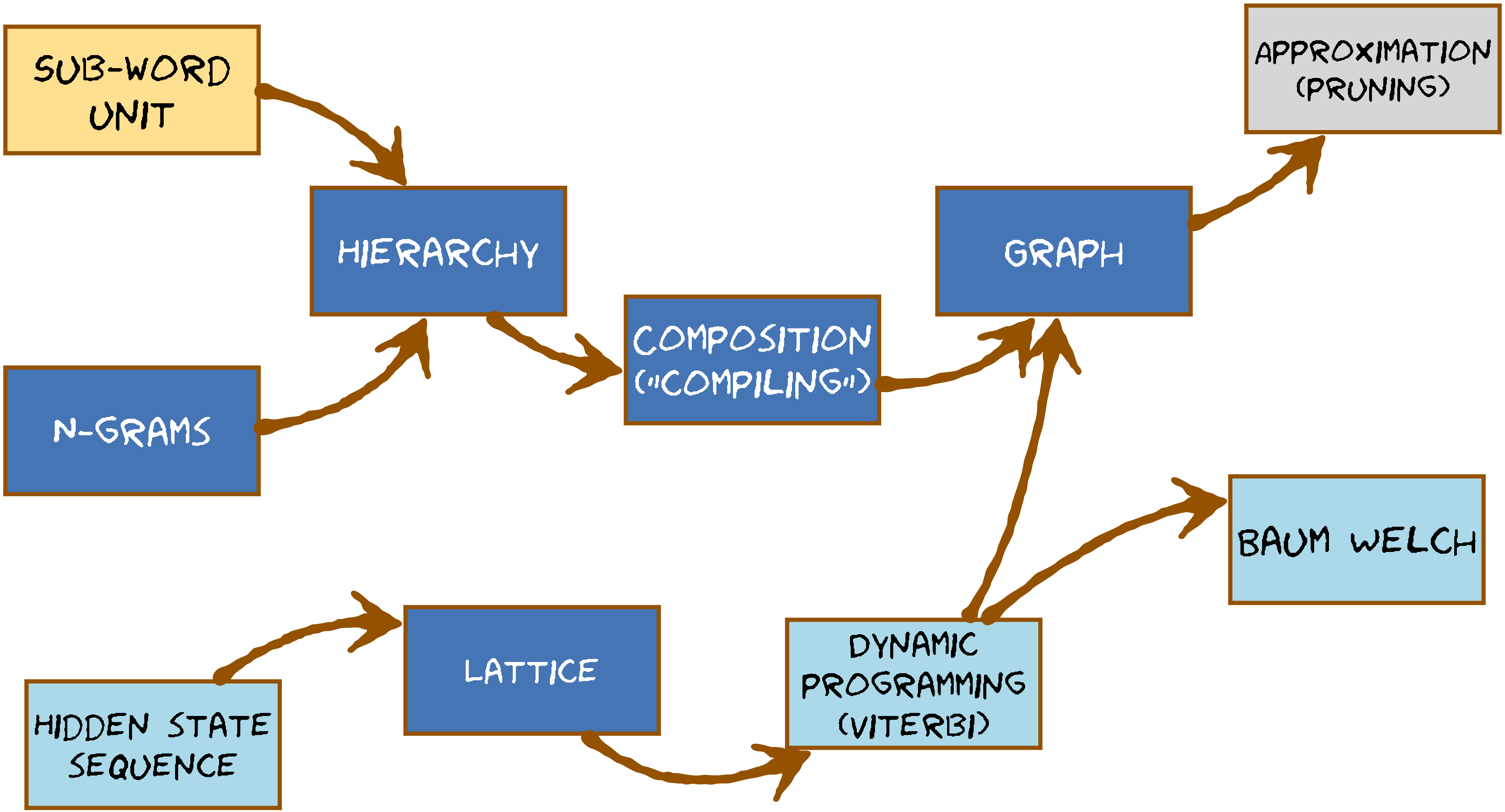
Composing finite state models


SUB-WORD UNIT
FINITE STATE TRANSDUCER
HIERARCHY
GRAPH
COMPOSITION ("COMPILING")
N-GRAMS

Compiling the recognition graph (it's just one big HMM)


SUB-WORD UNIT
HIERARCHY
N-GRAMS
COMPOSITION ("COMPILING")
GRAPH
APPROXIMATION (PRUNING)
DYNAMIC PROGRAMMING (VITERBI)

Pruning



SUB-WORD UNIT
HIERARCHY
N-GRAMS
COMPOSITION ("COMPILING")
GRAPH
APPROXIMATION (PRUNING)
BAUM WELCH
LATTICE
DYNAMIC PROGRAMMING (VITERBI)
HIDDEN STATE SEQUENCE

Prior, likelihood, posterior

$$P(W|O) = \frac{P(O|W)\ P(W)}{P(O)}$$

$P(W)$ **Prior** probability of word sequence W

$P(O)$ **Prior** probability of observation sequence O

$P(W|O)$ **Posterior** probability of word sequence W, after we have observed O

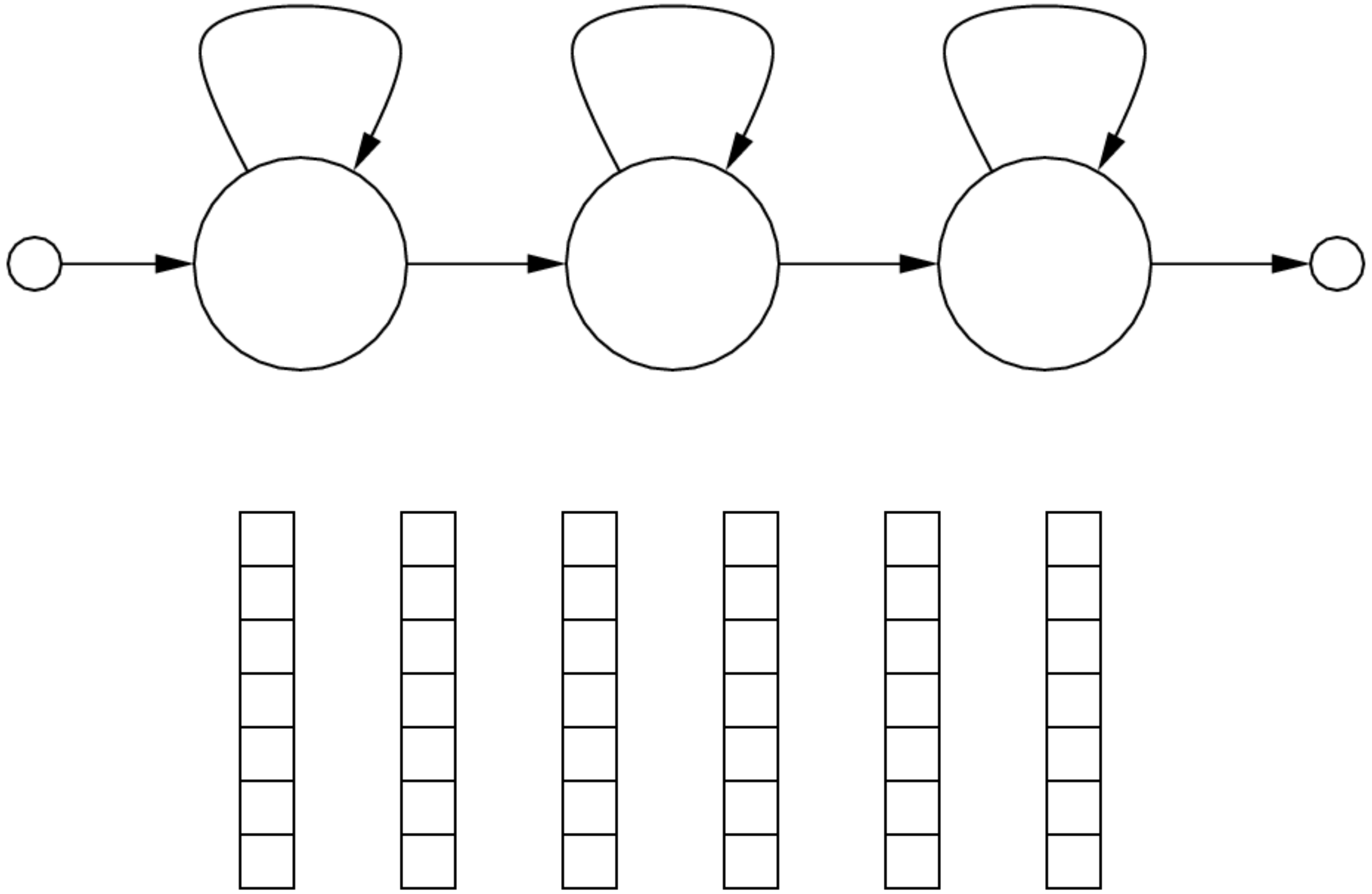
$P(O|W)$ **Likelihood** of observation sequence O being generated by model of word sequence W

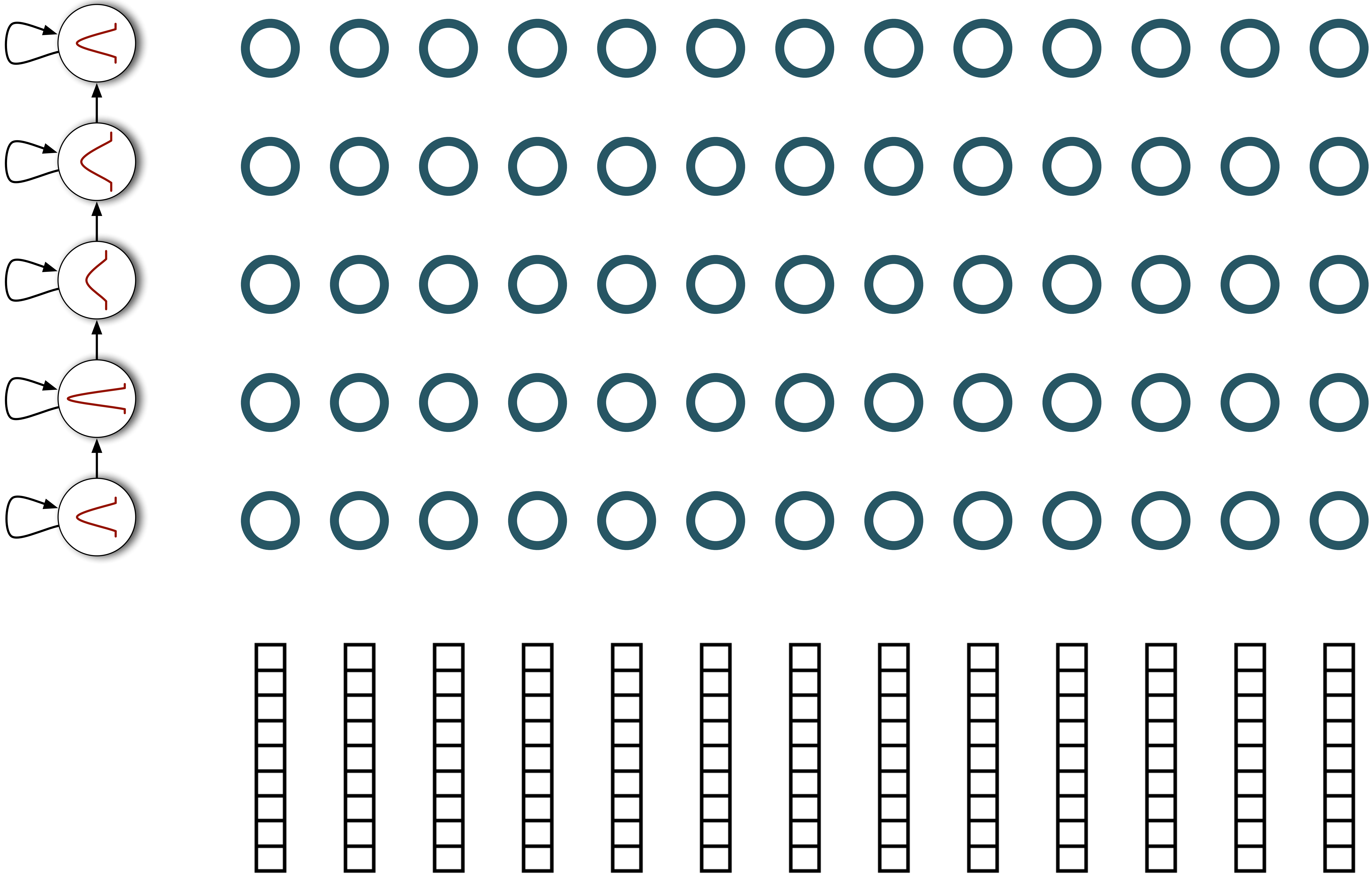
The hidden state sequence means we have several non-trivial problems

- Computing probability of a sequence of observations, given a model
  - Forward algorithm - gives total probability (a sum)
  - Viterbi algorithm - approximates that sum with a max
- Estimating the parameters of the model, given an observation sequence
  - Forward-Backward algorithm - effectively “aligns” observations with states

Key properties of the HMM

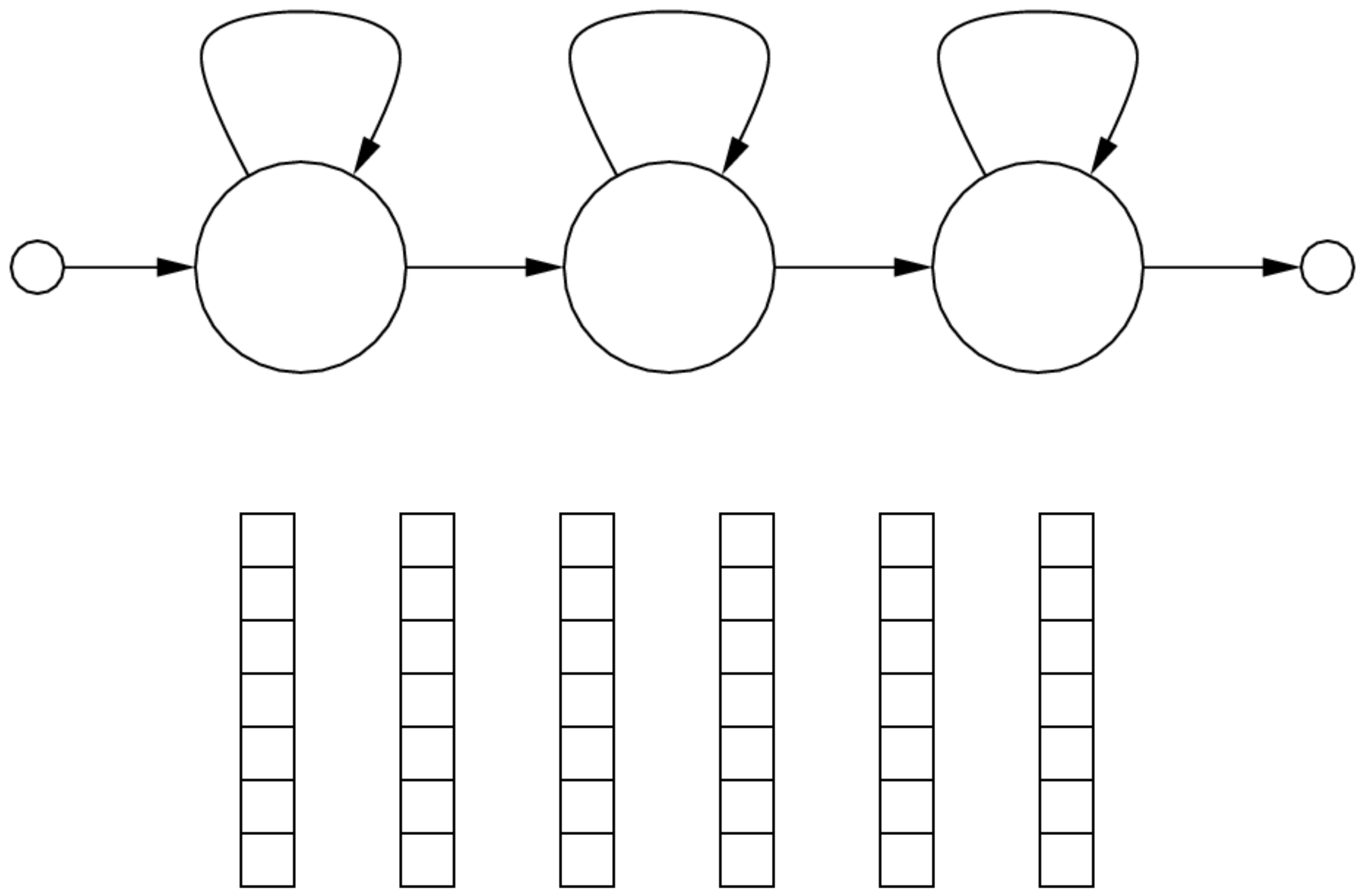
- State sequence is hidden

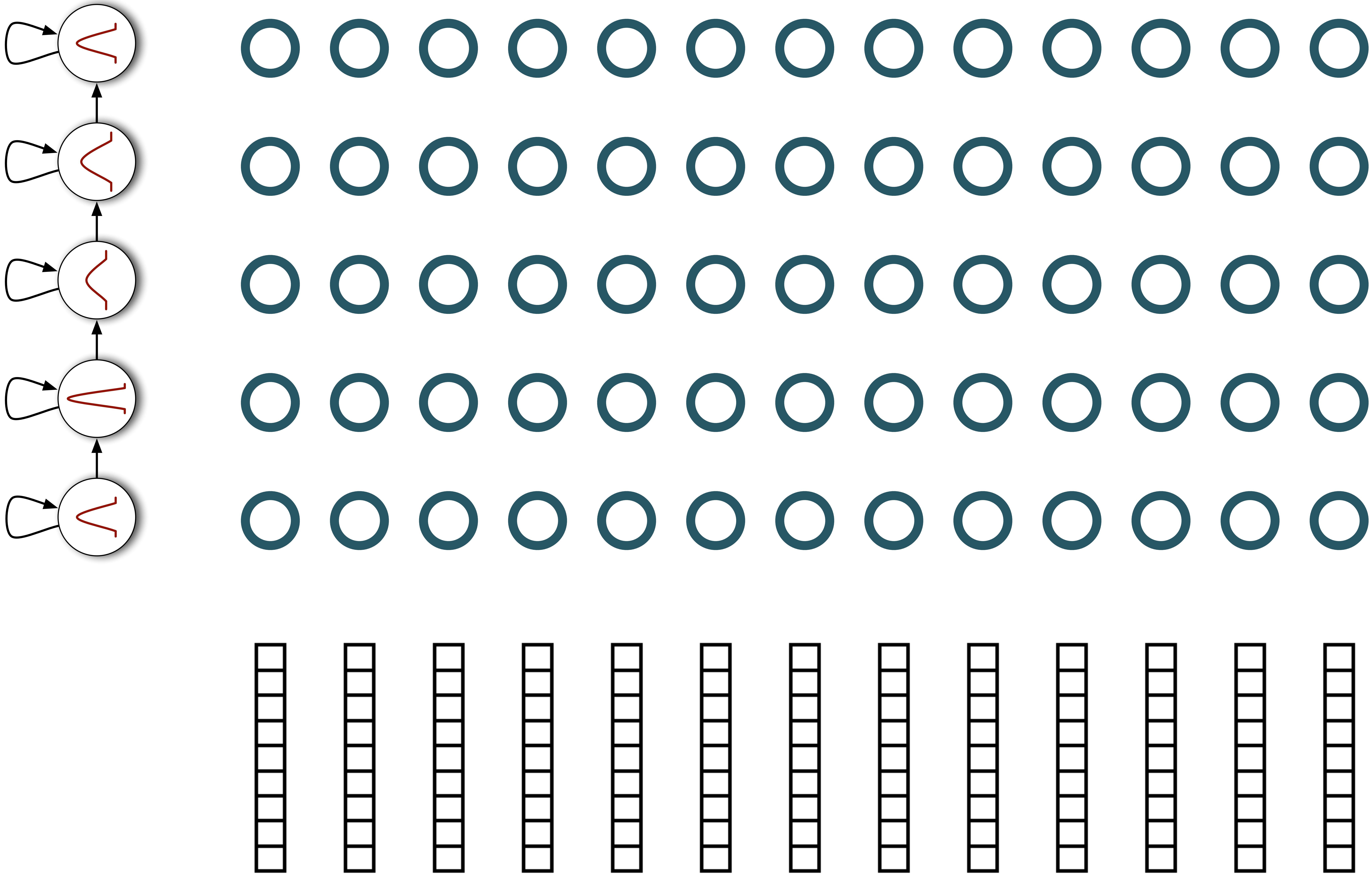




Key properties of the HMM

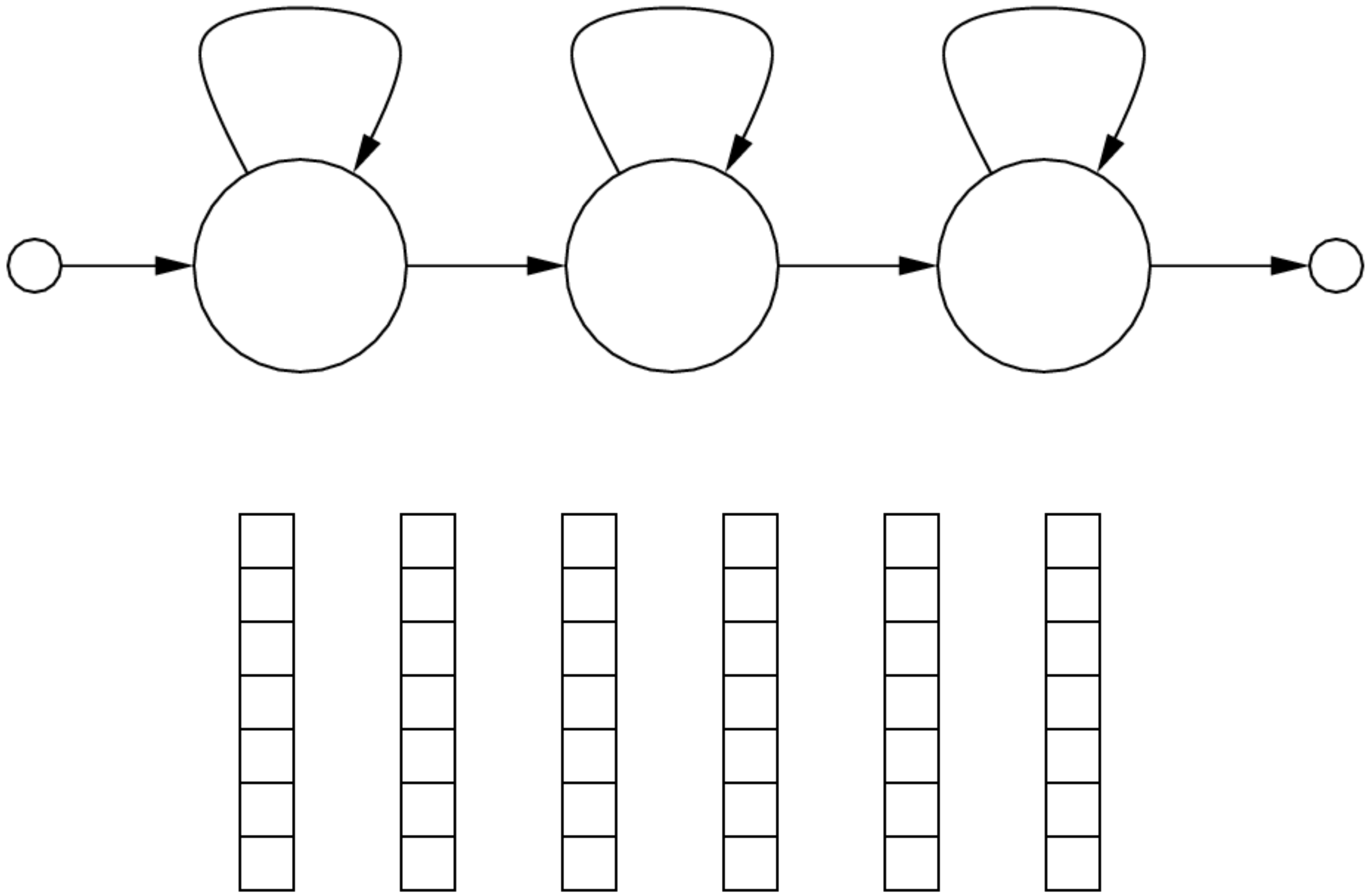
- Markov = memoryless
- = The future is independent of the past, given the present (“the present” = the state)

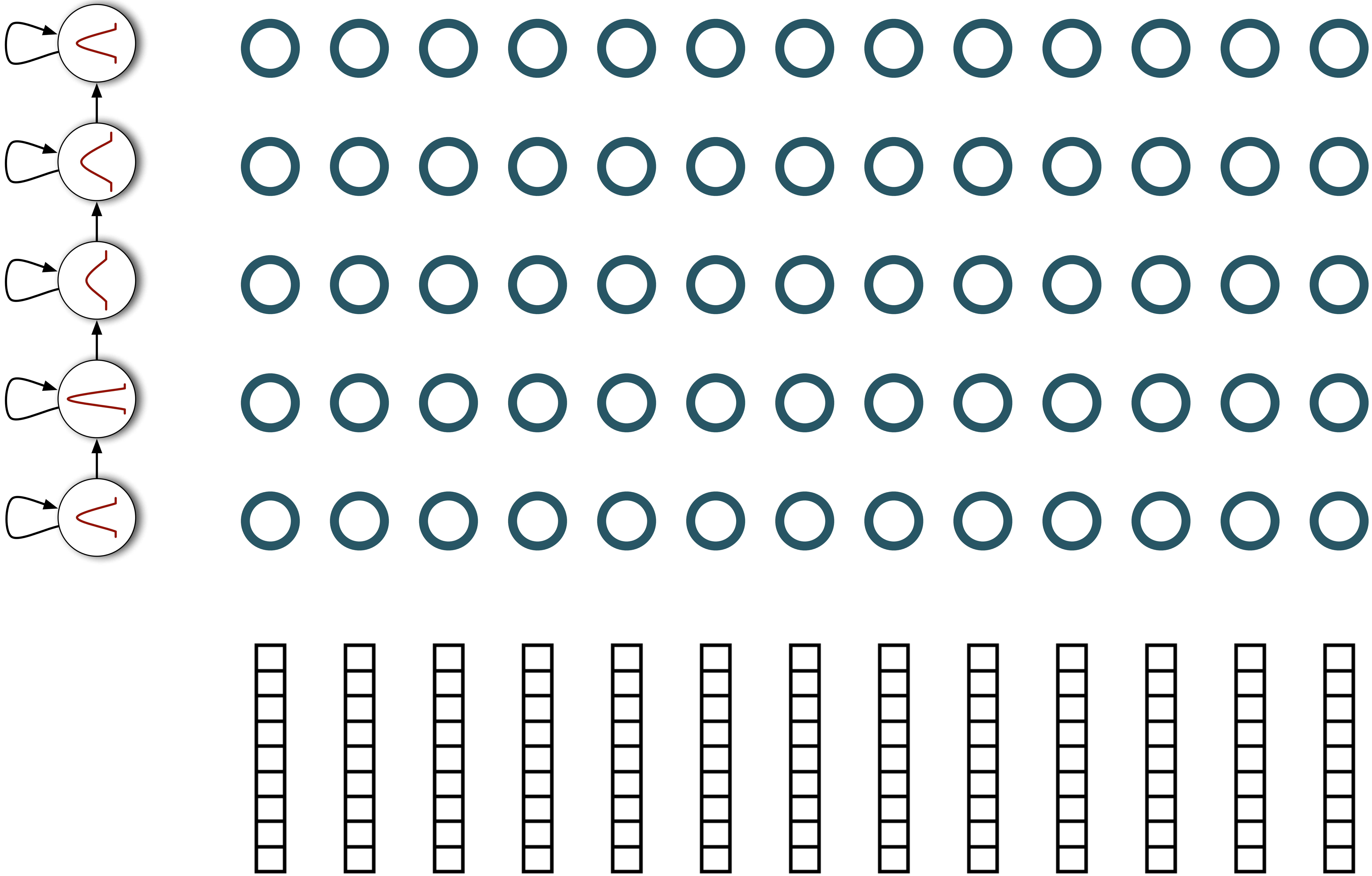




Key properties of the HMM

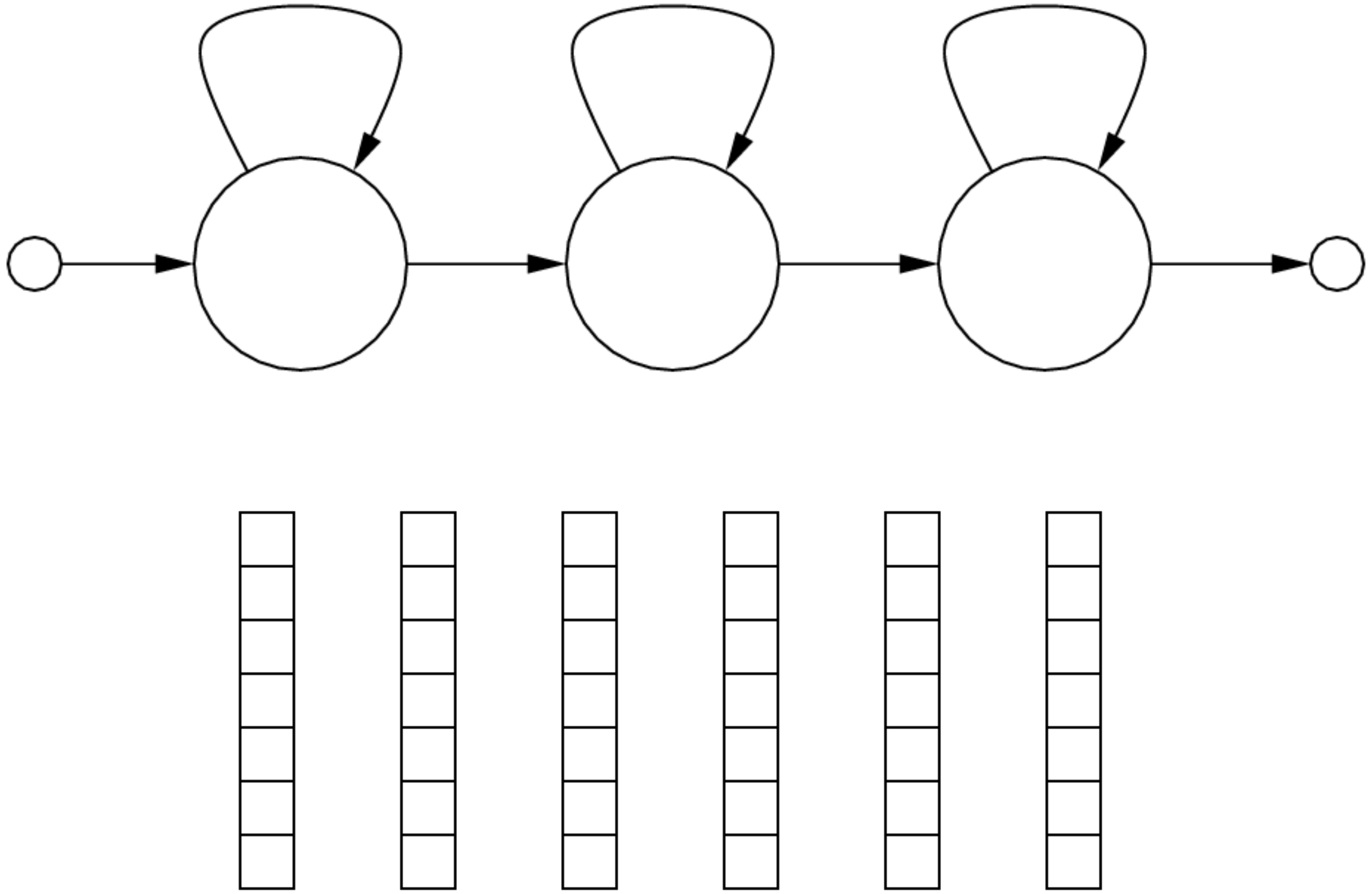
- Observations are conditionally independent, given the state
- = Probability of each observation depends only on current state

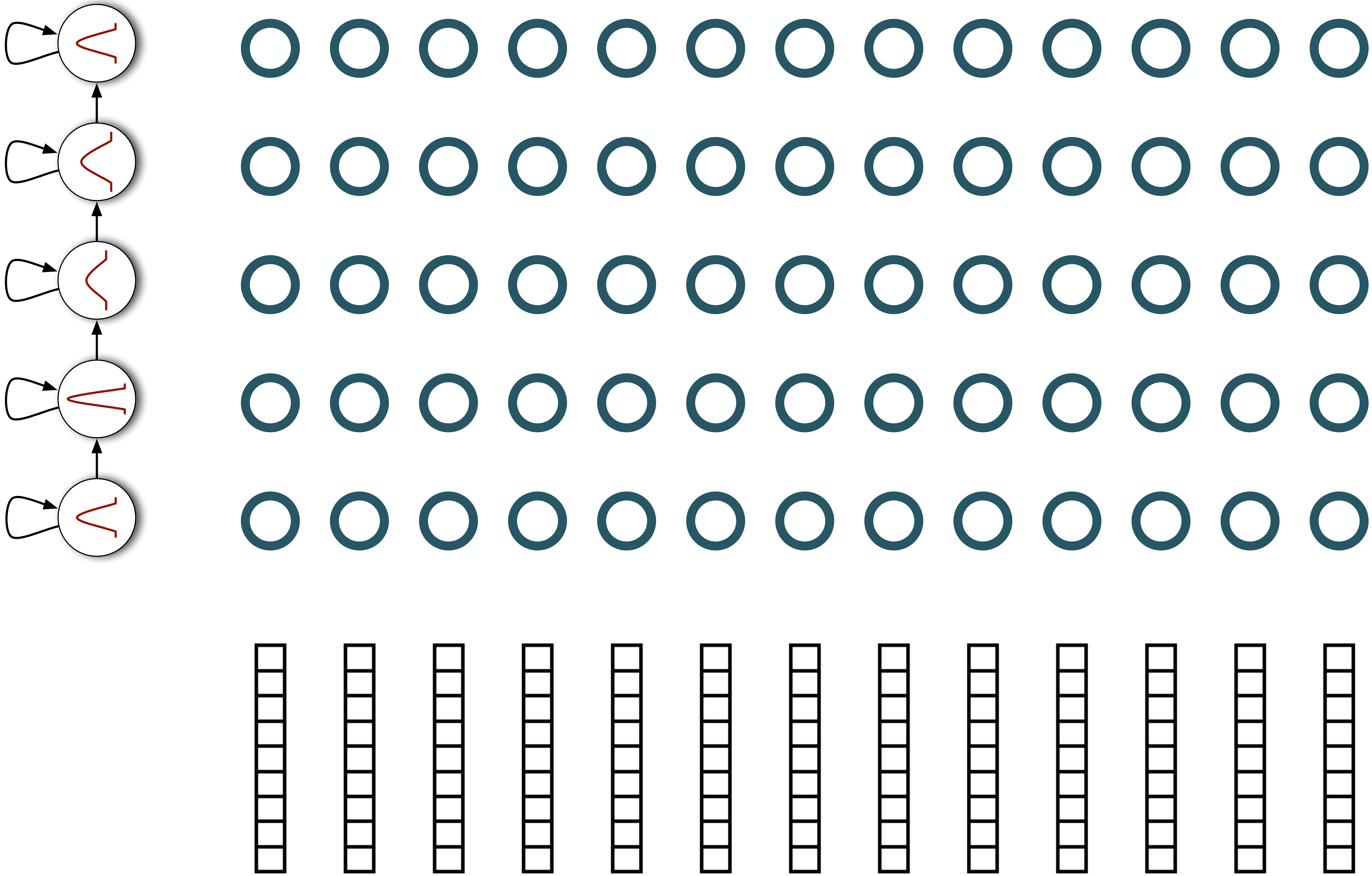




Computations needed for HMMs

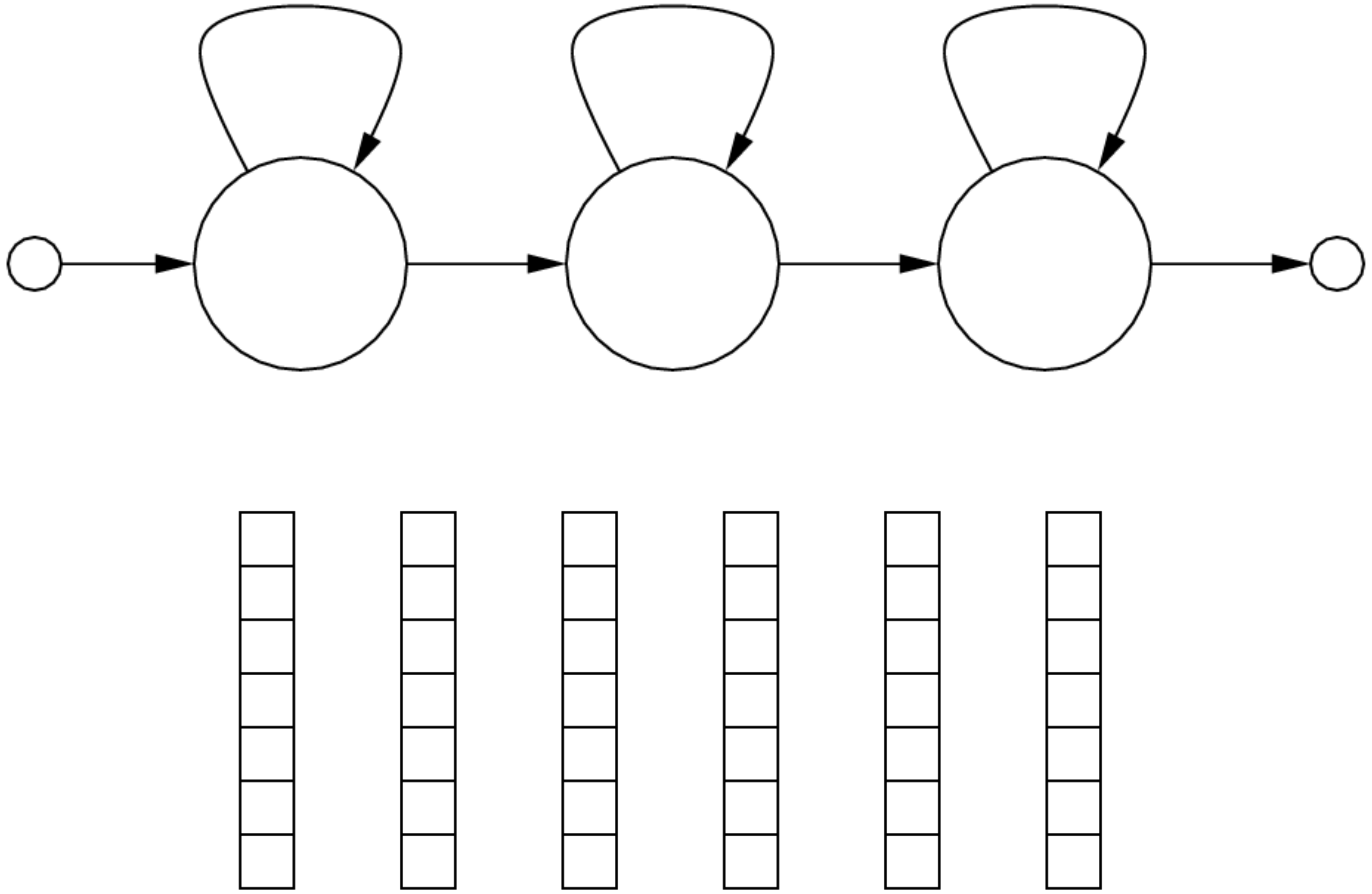
- Approximately computing the probability of a sequence of observations
  - by assuming the model used the single most likely state sequence
- Finding that sequence **exactly** and efficiently
- **Viterbi** algorithm

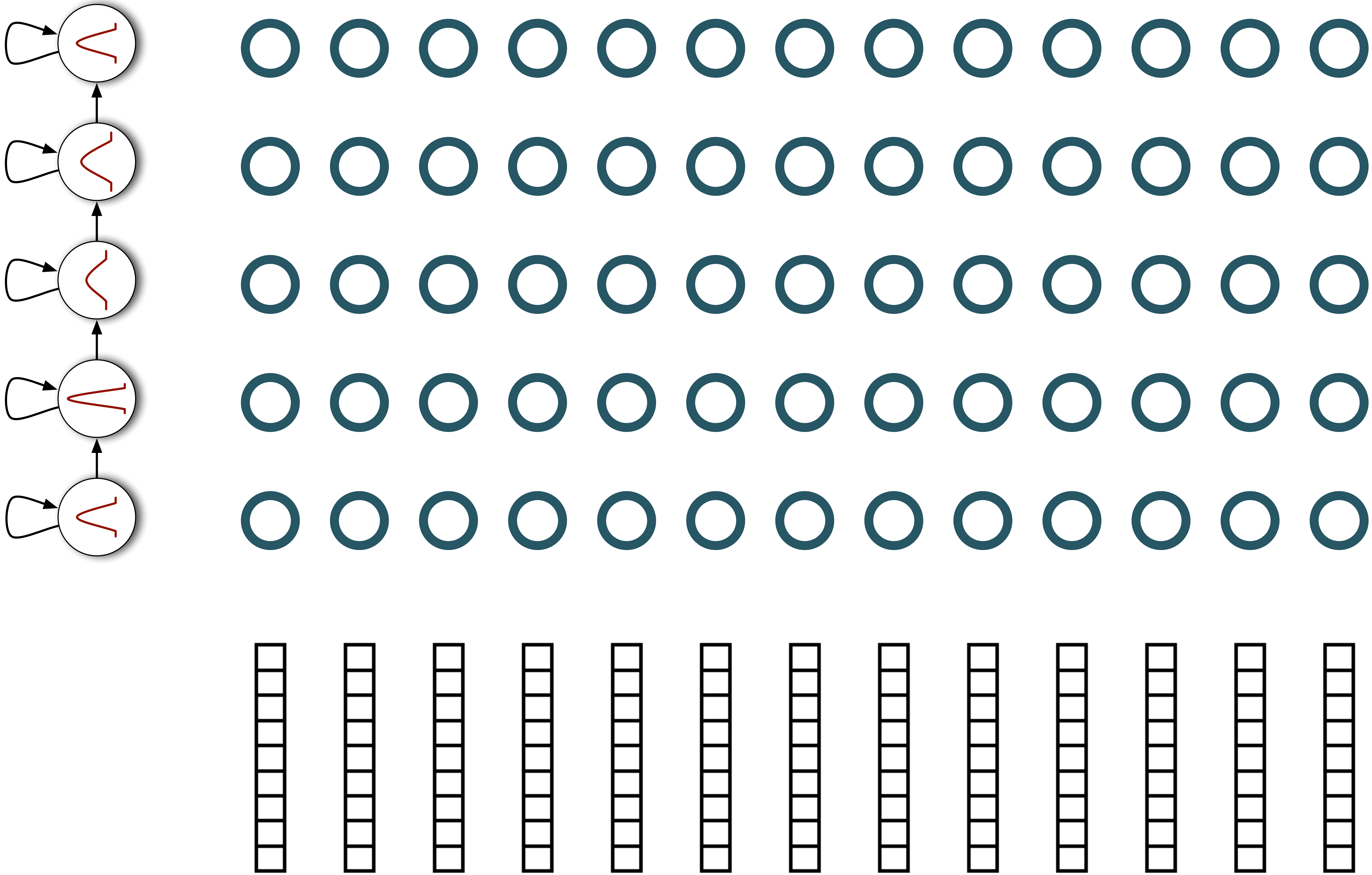




Computations needed for HMMs

- Exactly computing the probability of a sequence of observations
  - by considering all possible state sequences
- **Forward** algorithm

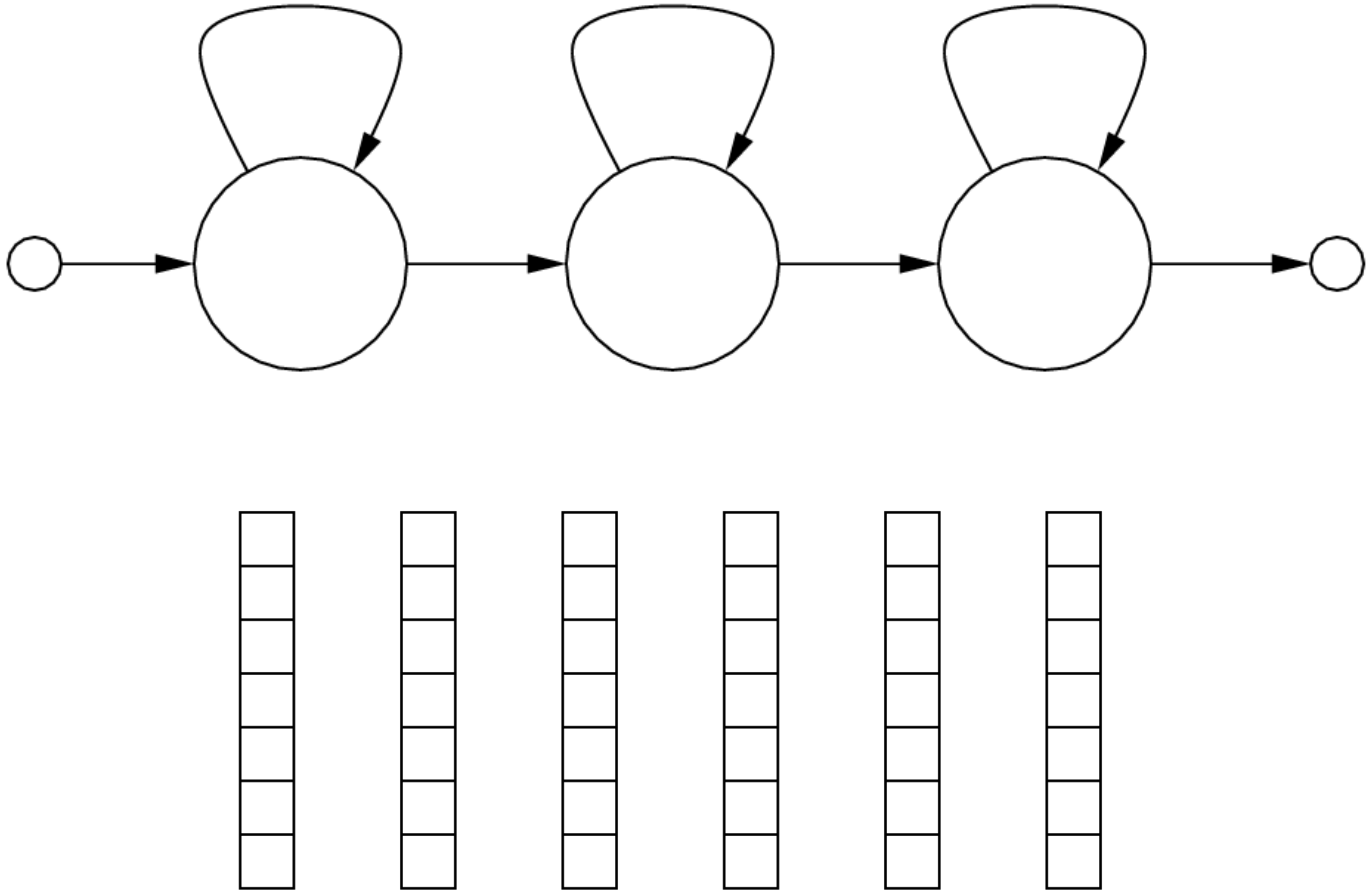


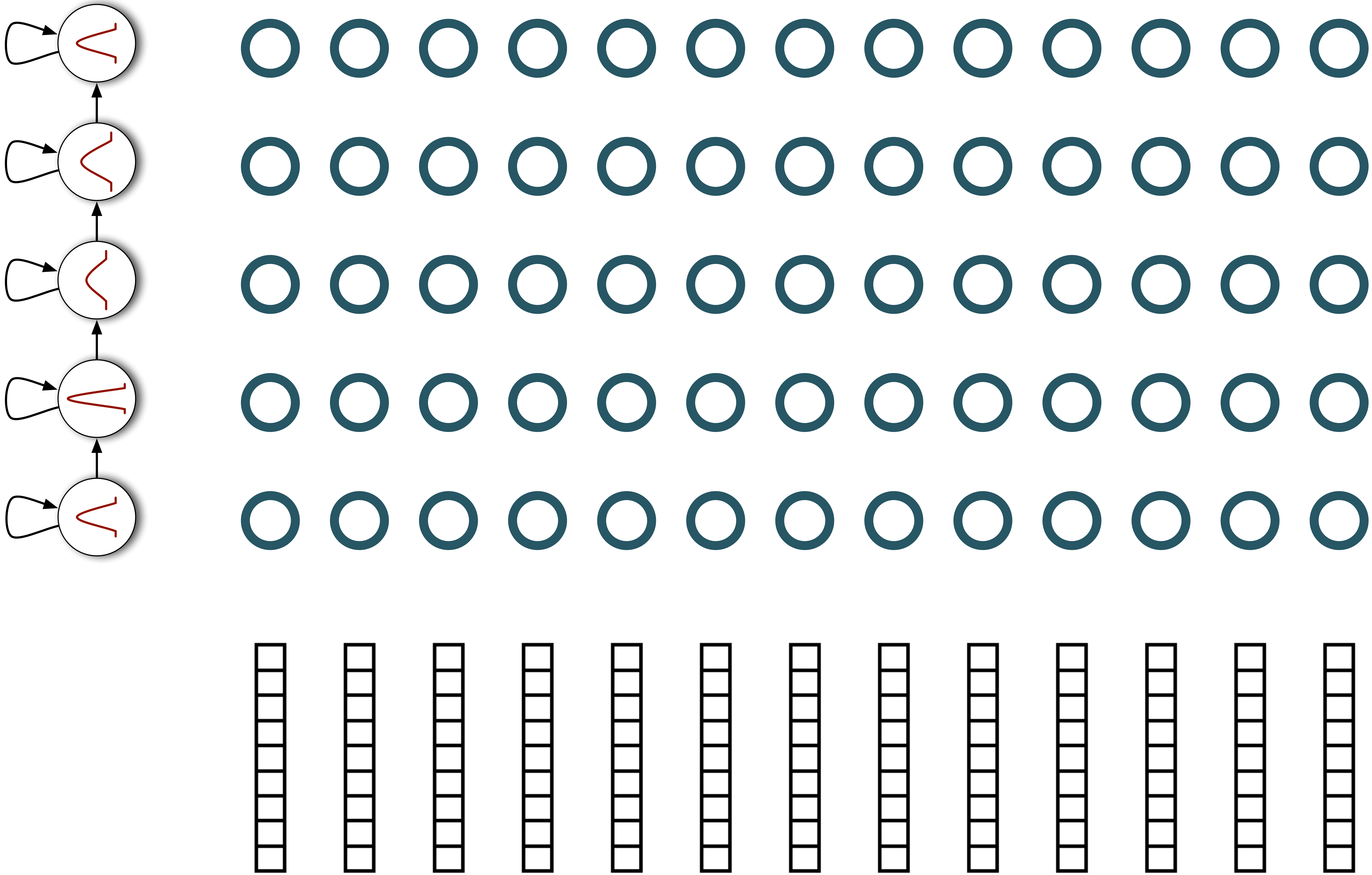


Computations needed for HMMs

- Find the probability that a particular state emitted a given observation

  = the probability of being in that state at the given time = state occupancy probability
- Forward-Backward algorithm = **Baum-Welch** algorithm



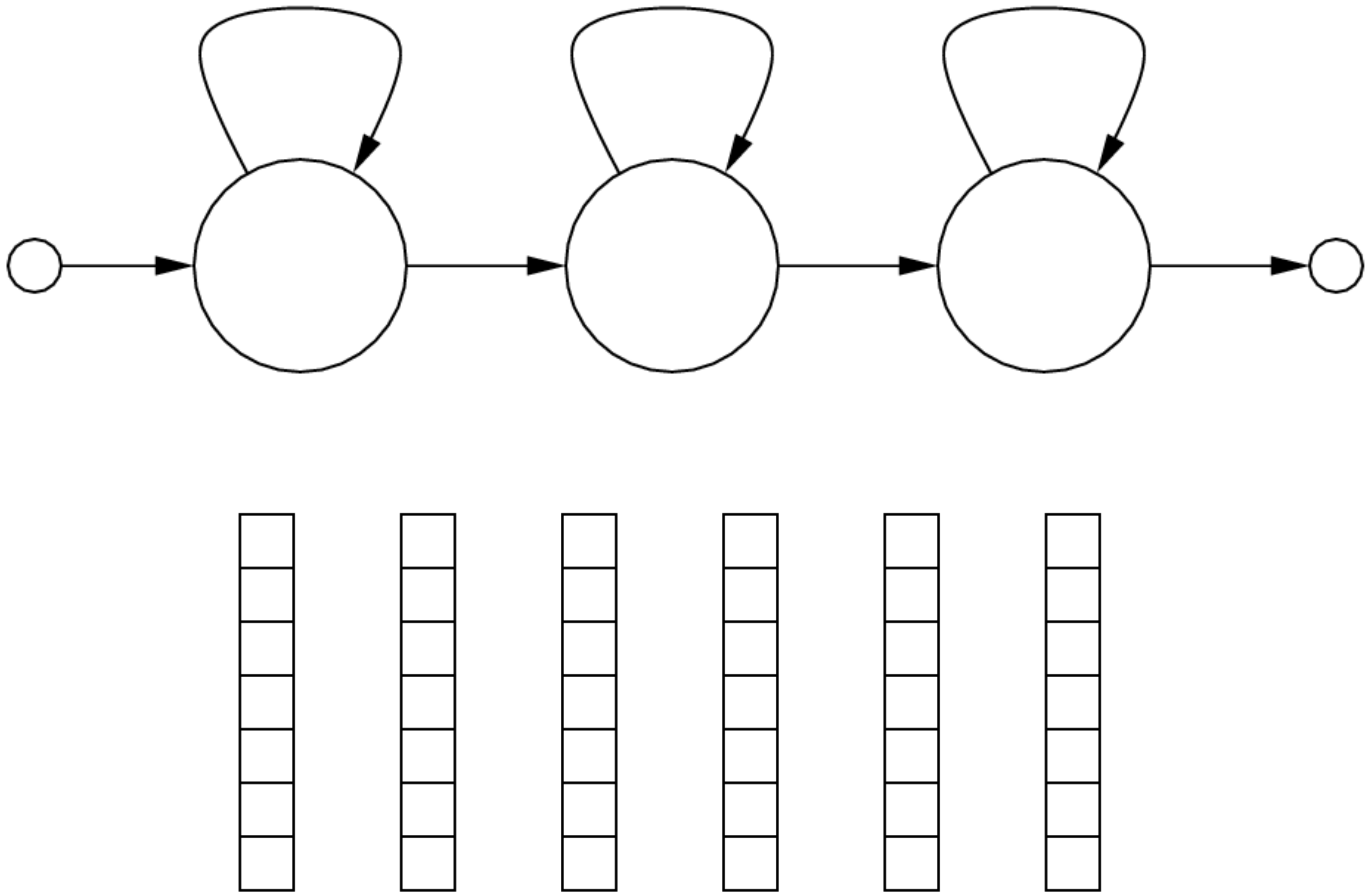


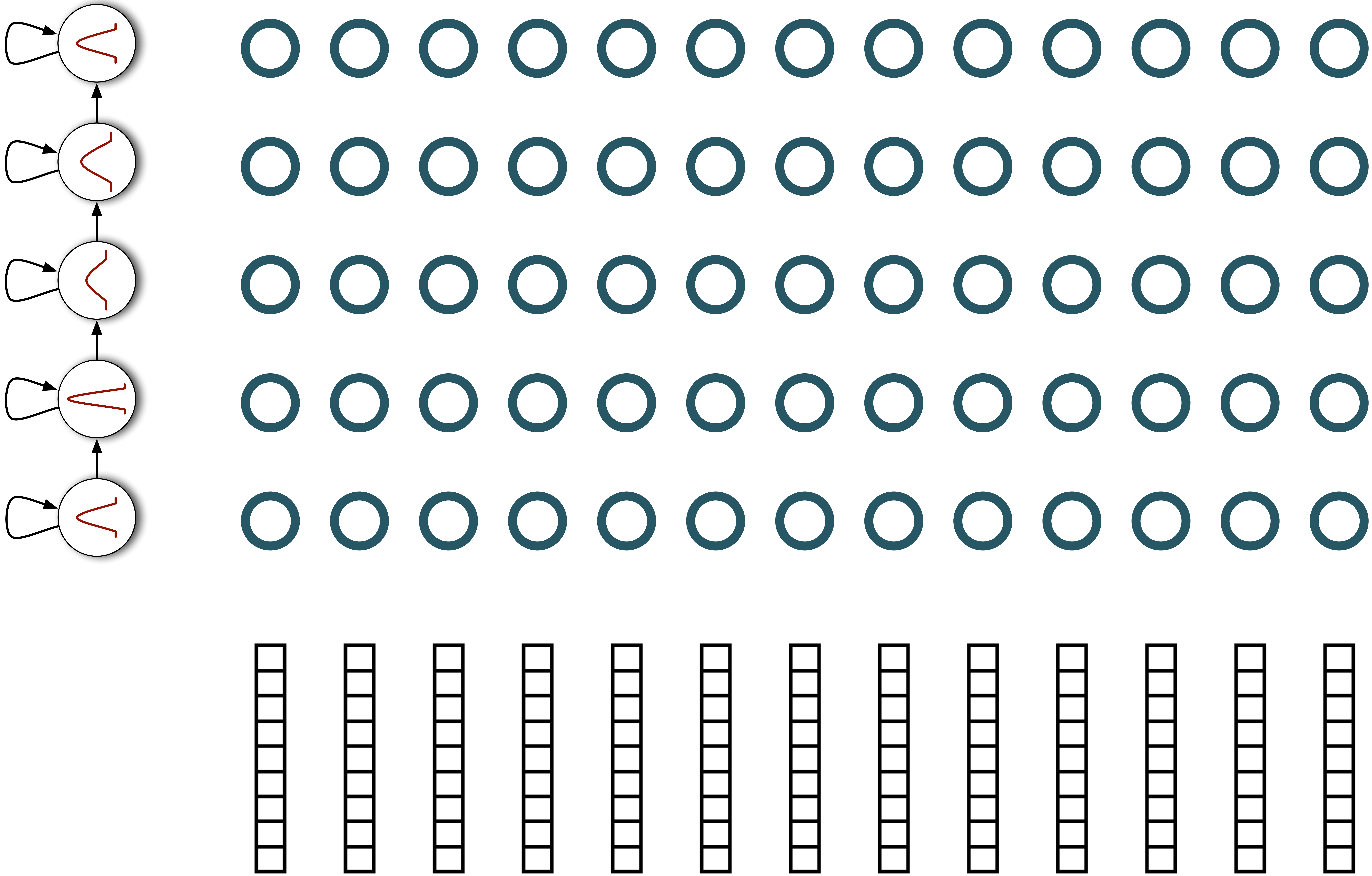
The Baum-Welch algorithm

- We're going to develop it starting from simpler training methods
  - uniform segmentation
  - Viterbi training
- These simpler methods make a 'hard' alignment between observations and states
  - they only consider one possible state sequence - an approximation

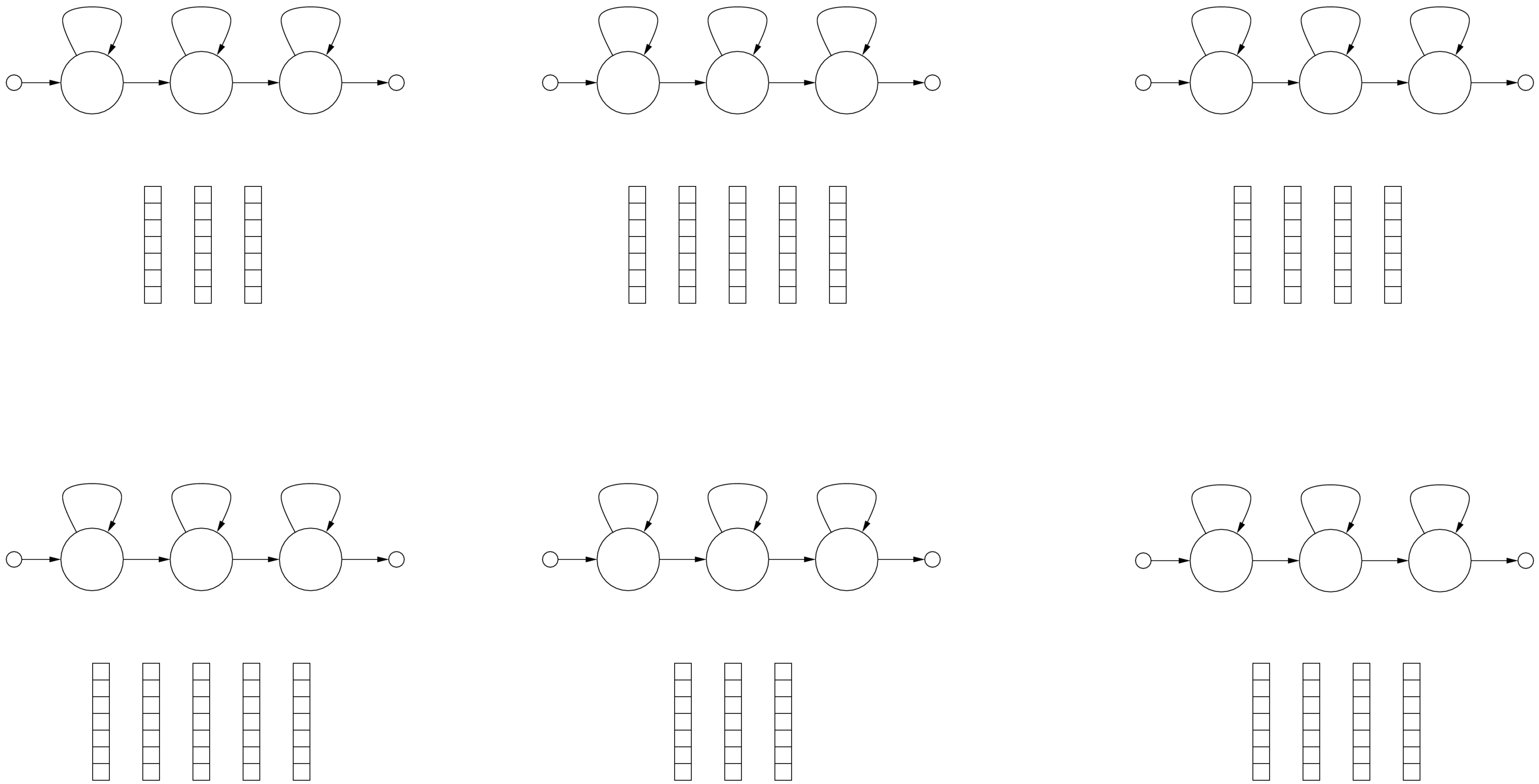
- We could describe Baum-Welch as using a 'soft' or probabilistic alignment
  - it considers all possible state sequences - the correct thing to do

Before Baum-Welch: two simpler training methods



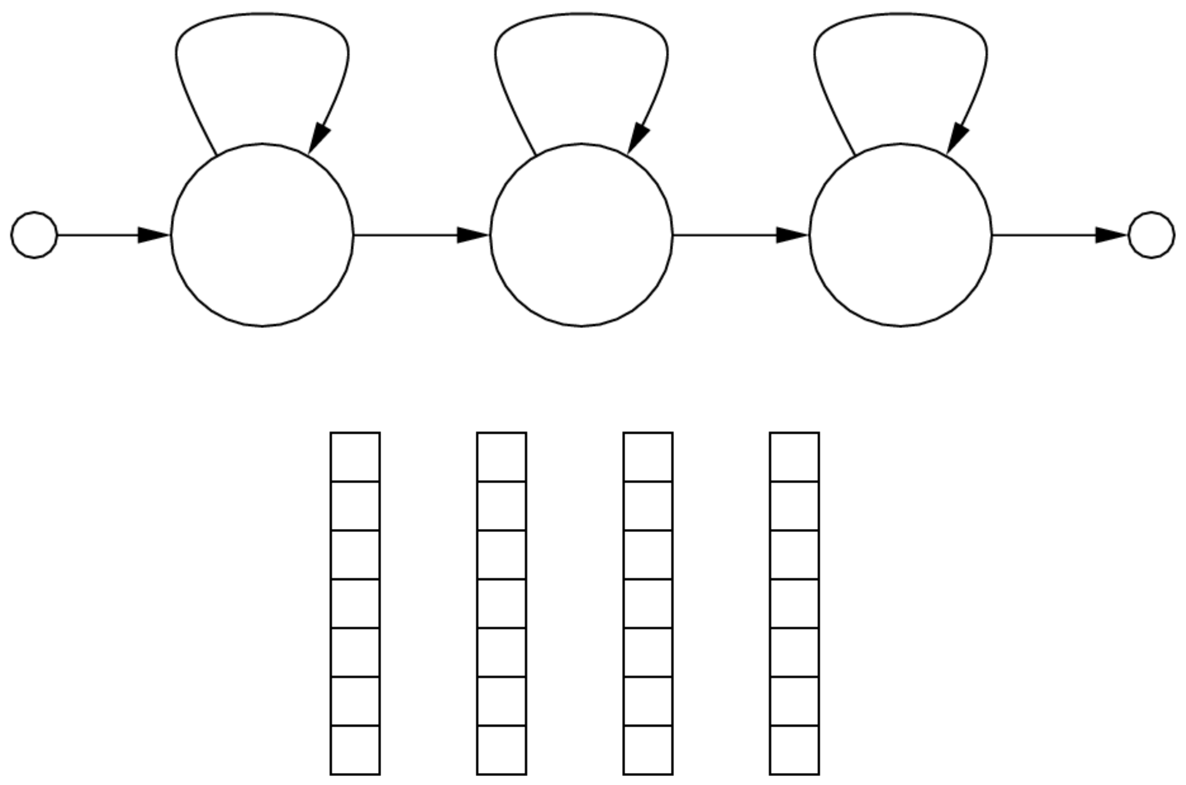


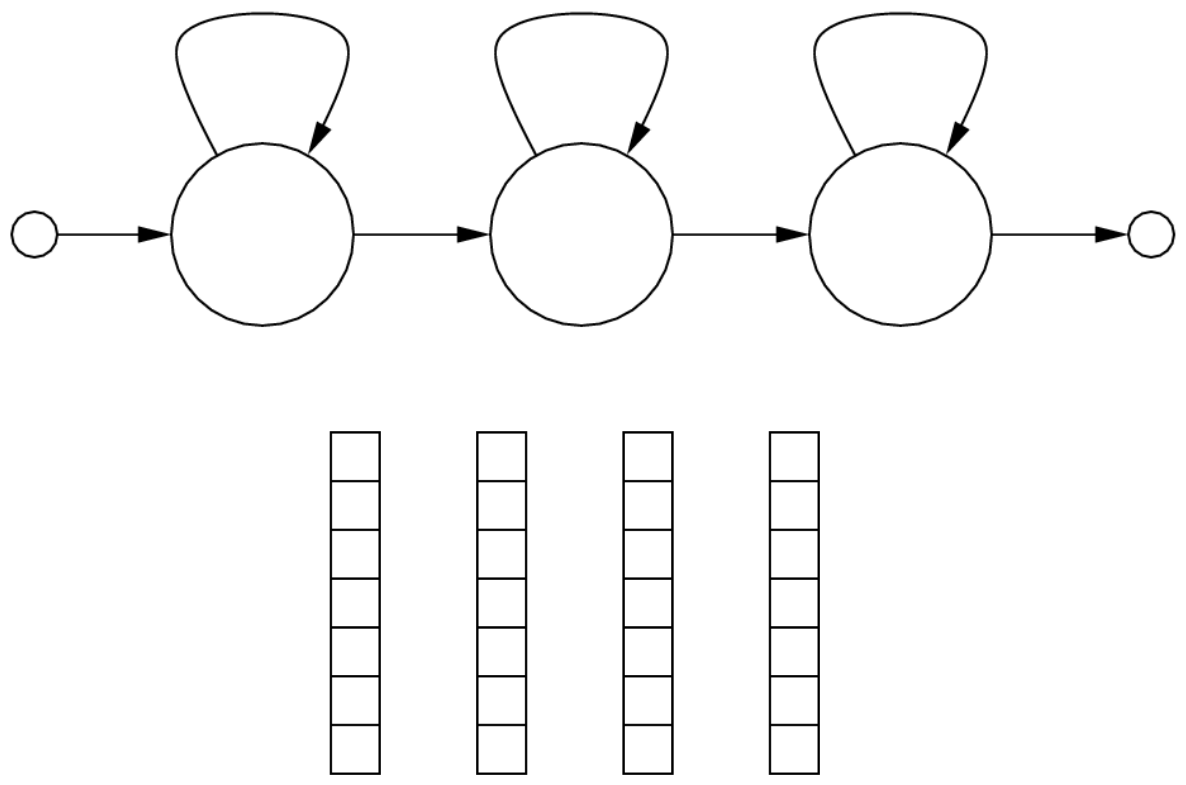
Using multiple observation sequences (i.e., multiple training examples)

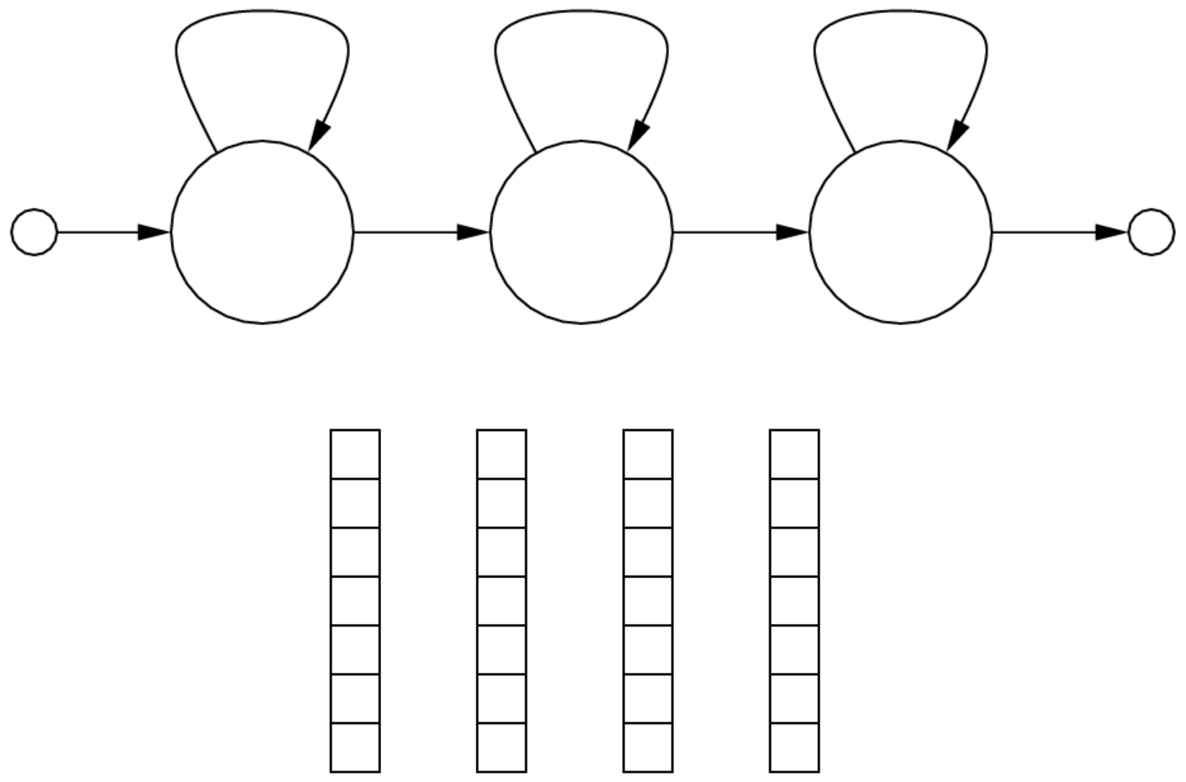


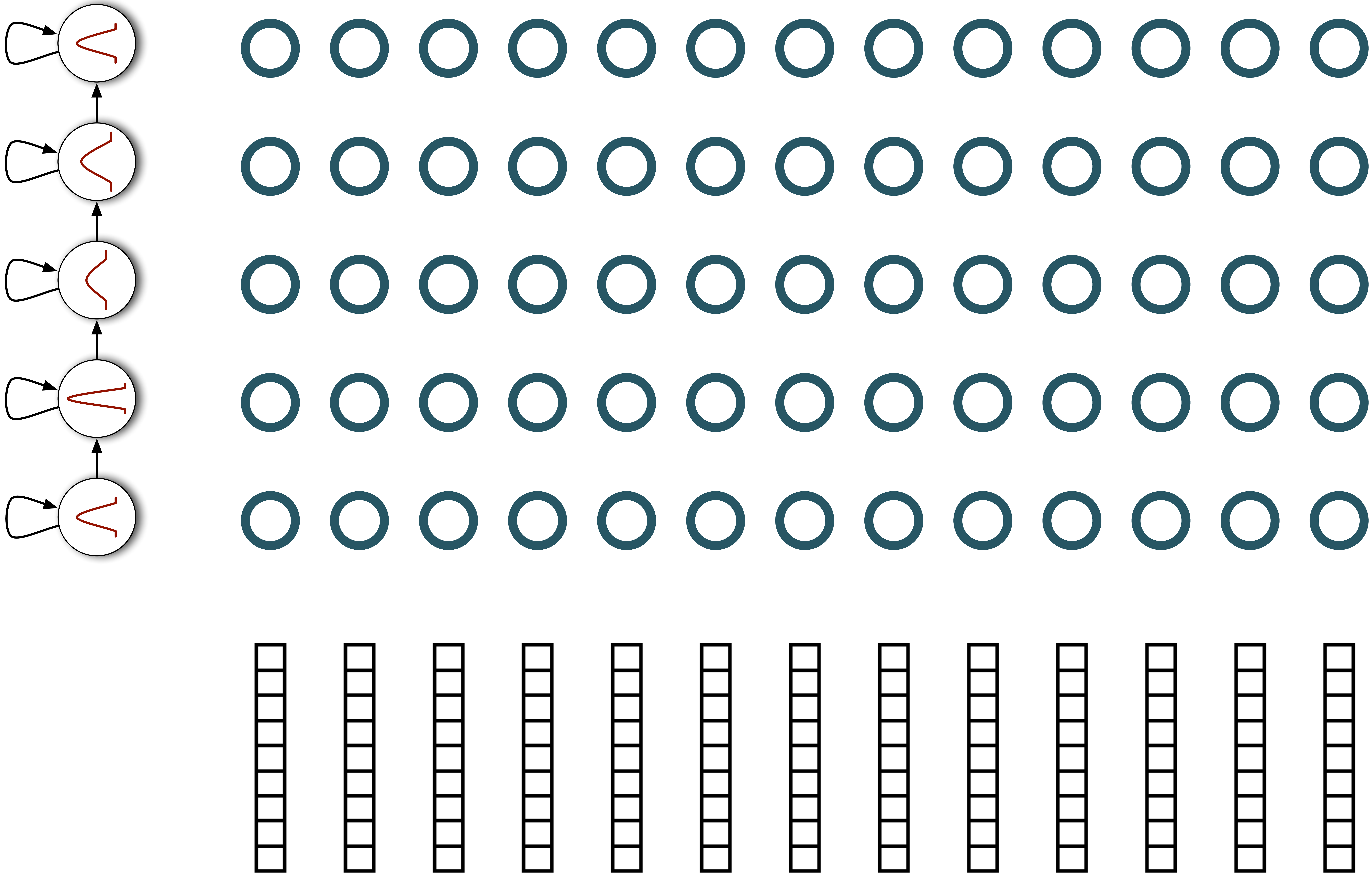
From one alignment to all alignments

(alignment *is the same thing as* state sequence)

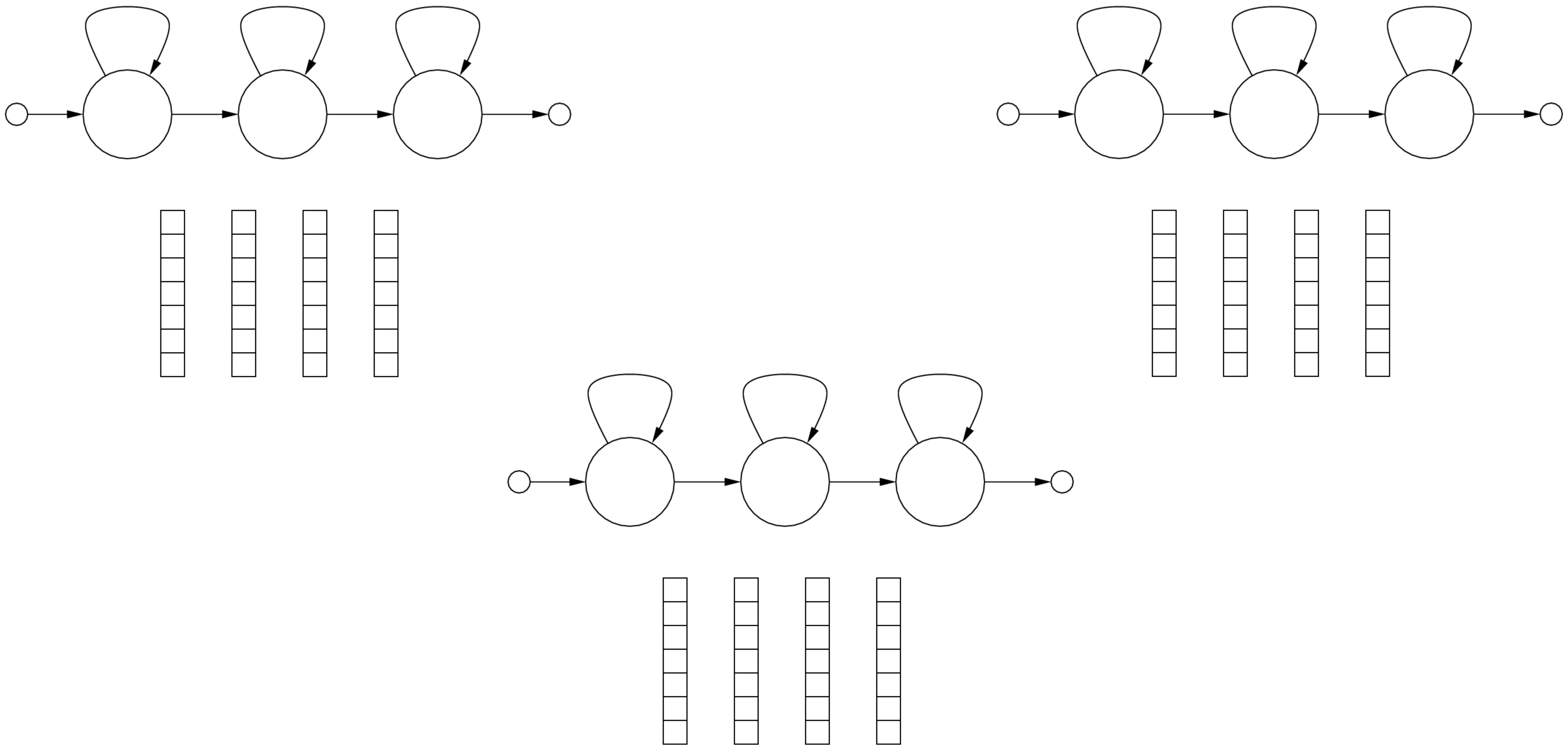








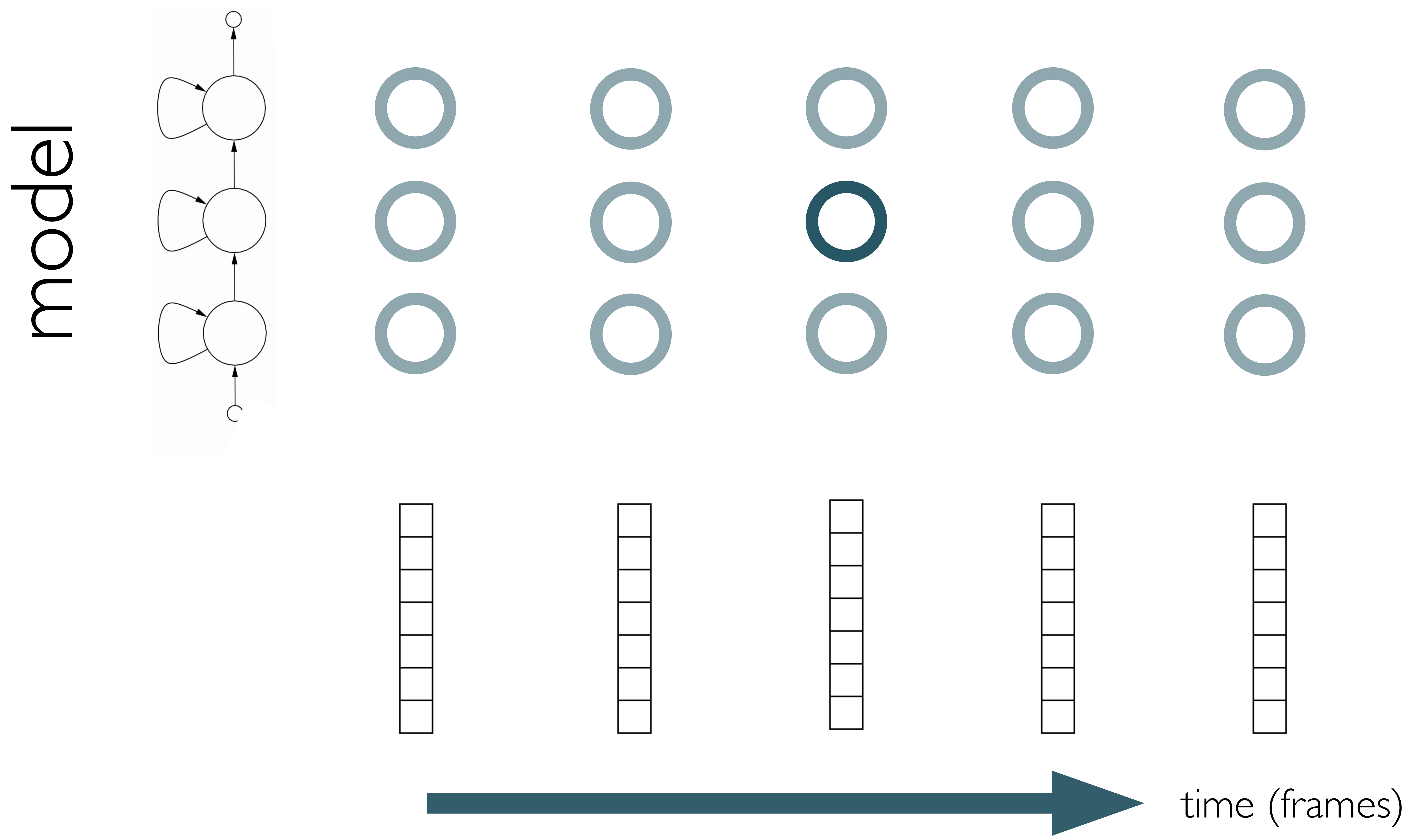
State occupancy probability



How much should each observation contribute to estimating each state's Gaussian pdf parameters (mean & variance)?

State occupancy probability

= probability of being in a particular state at a particular time



What next?

- Speech Synthesis

- Automatic Speech Recognition