Search and Decoding

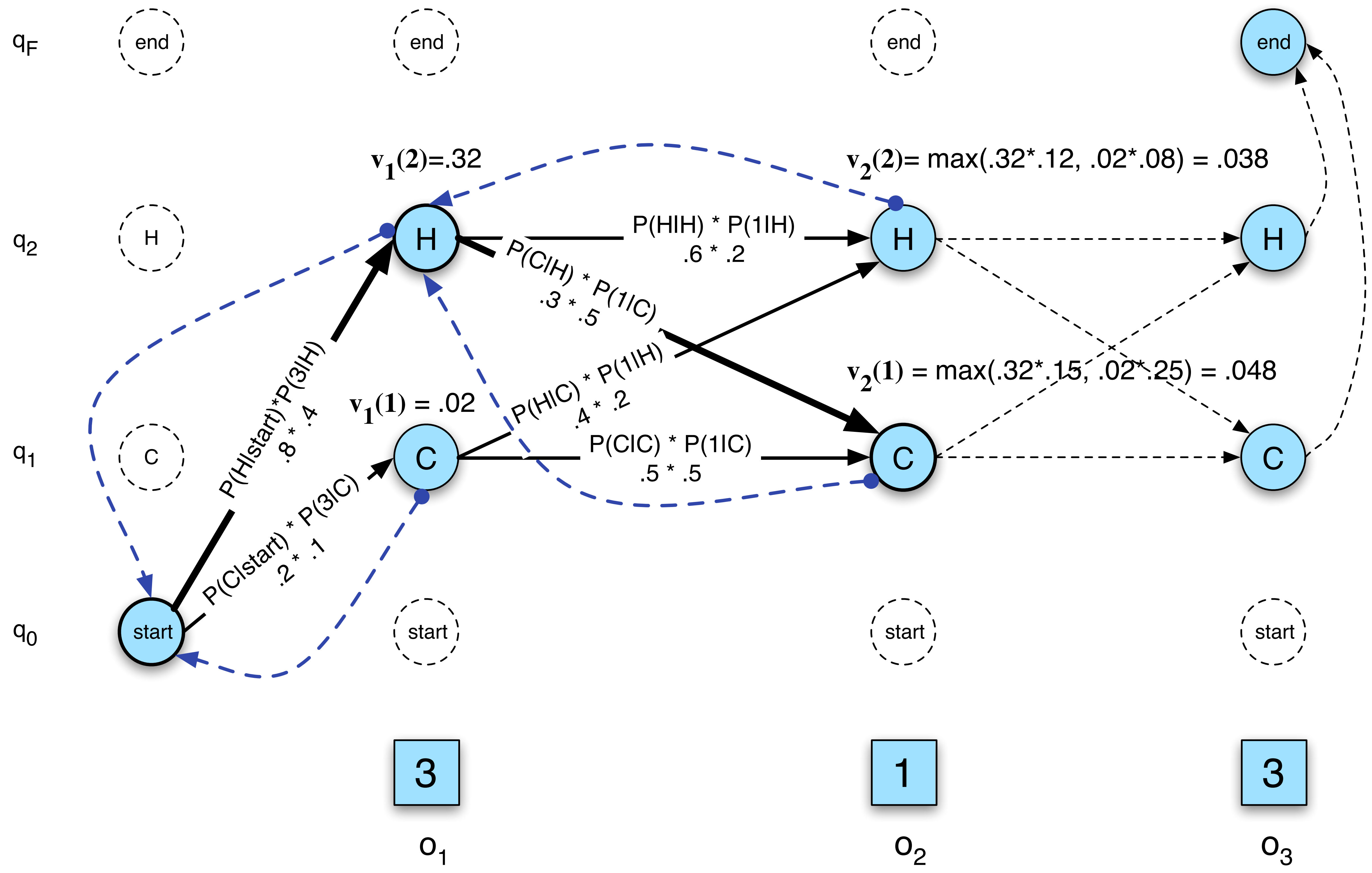
Lecture 16

CS 753

Instructor: Preethi Jyothi

Recall Viterbi search

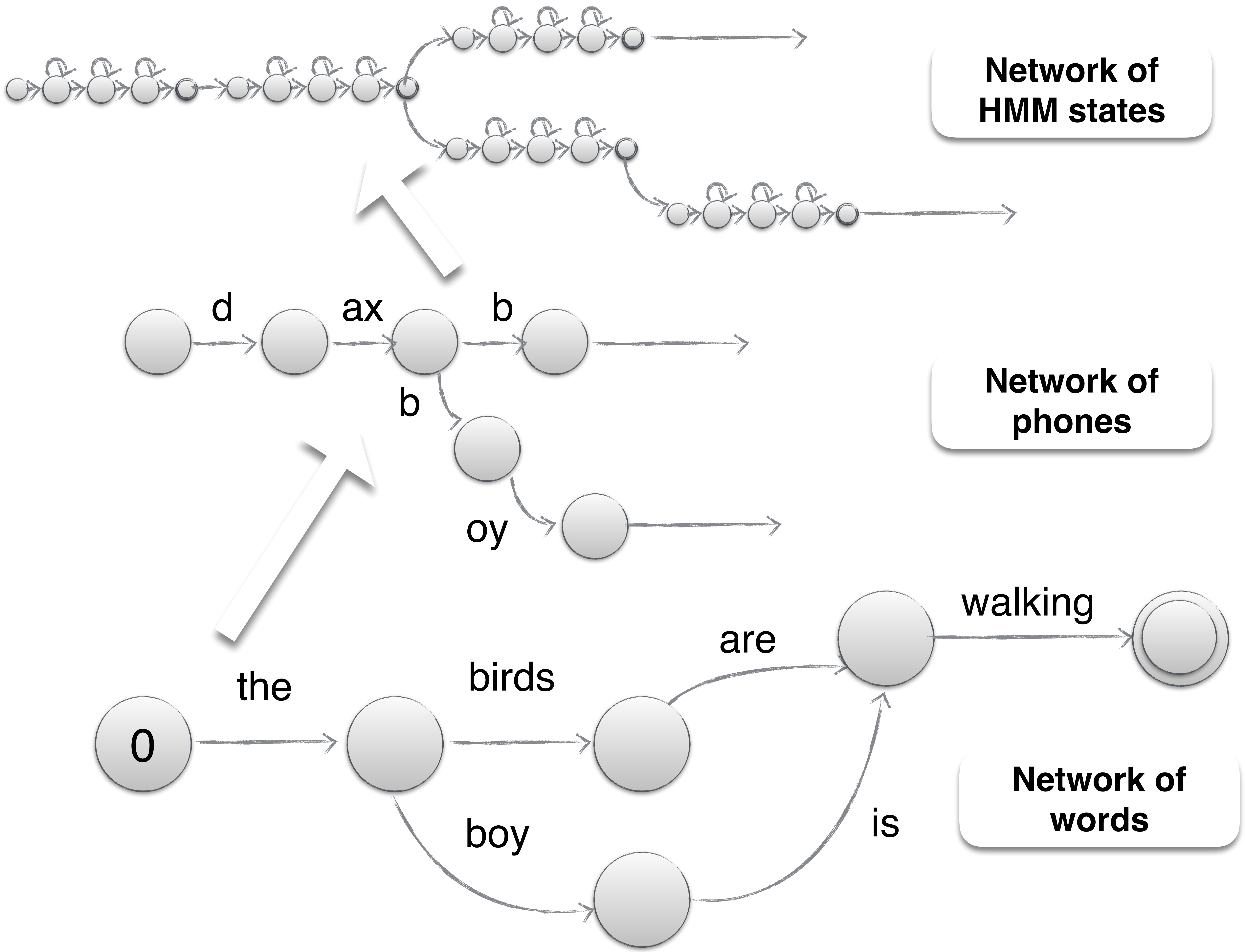
- Viterbi search finds the most probable path through a trellis of time on the X-axis and states on the Y-axis



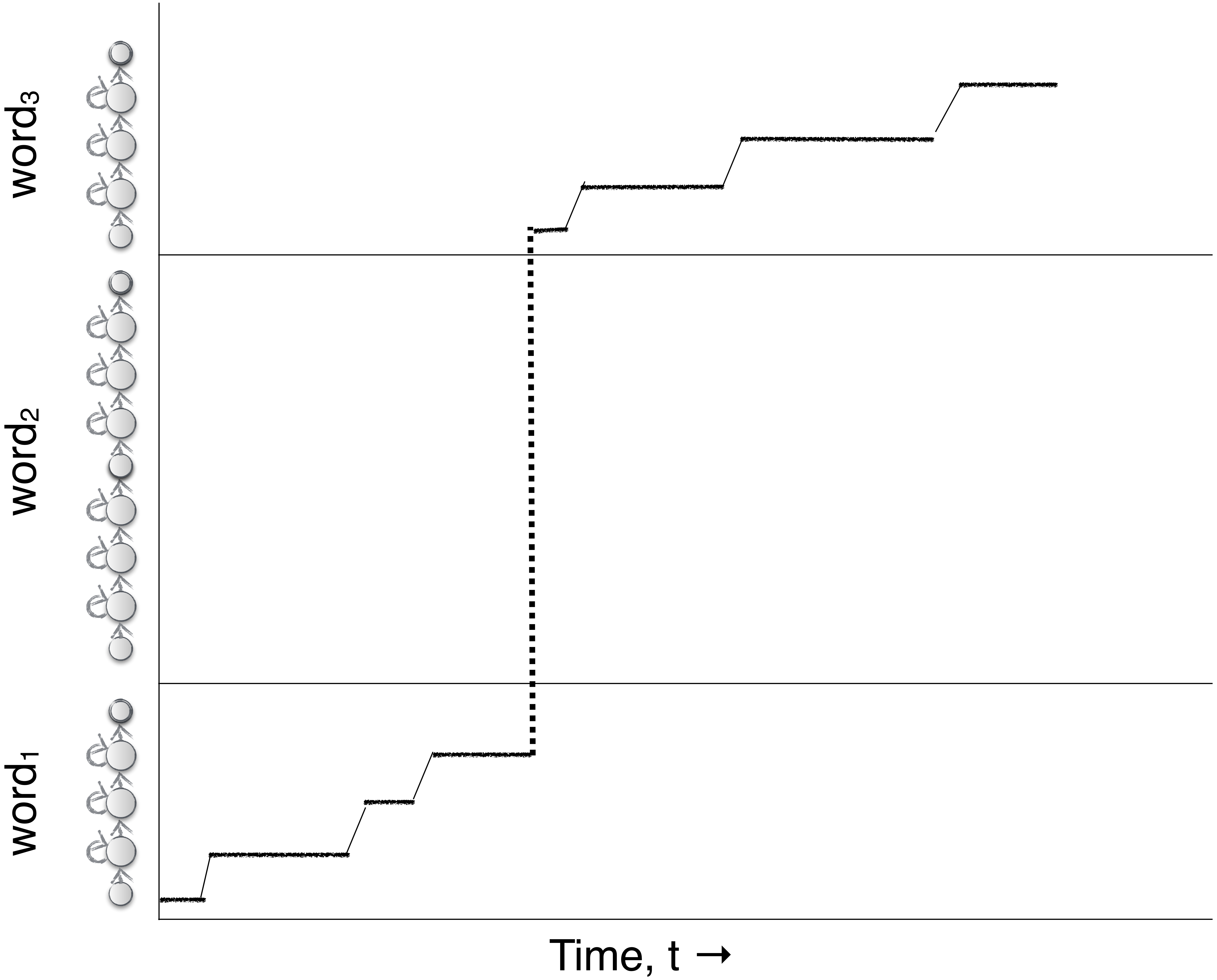
- Viterbi algorithm: Only needs to maintain information about the most probable path at each state

Image from [JM]: Jurafsky & Martin, 3rd edition, Chapter 9

ASR Search Network



Time-state trellis



Viterbi search over the large trellis

- Exact search is infeasible for large vocabulary tasks
  - Unknown word boundaries
  - Ngram language models greatly increase the search space
- Solutions
  - Compactly represent the search space using WFST-based optimisations
  - Beam search: Prune away parts of the search space that aren’t promising

Viterbi search over the large trellis

- Exact search is infeasible for large vocabulary tasks
  - Unknown word boundaries
  - Ngram language models greatly increase the search space
- Solutions
  - Compactly represent the search space using WFST-based optimisations
  - Beam search: Prune away parts of the search space that aren’t promising

Two main WFST Optimizations

- Use determinization to reduce/eliminate redundancy

Recall not all weighted transducers are determinizable

To ensure determinizability of $L \circ G$, introduce disambiguation symbols in L to deal with homophones in the lexicon

read : r eh d #1
red : r eh d #2

Propagate the disambiguation symbols as self-loops back to C and H. Resulting machines are $\tilde{H}$, $\tilde{C}$, $\tilde{L}$

Two main WFST Optimizations

- Use determinization to reduce/eliminate redundancy
- Use minimization to reduce space requirements

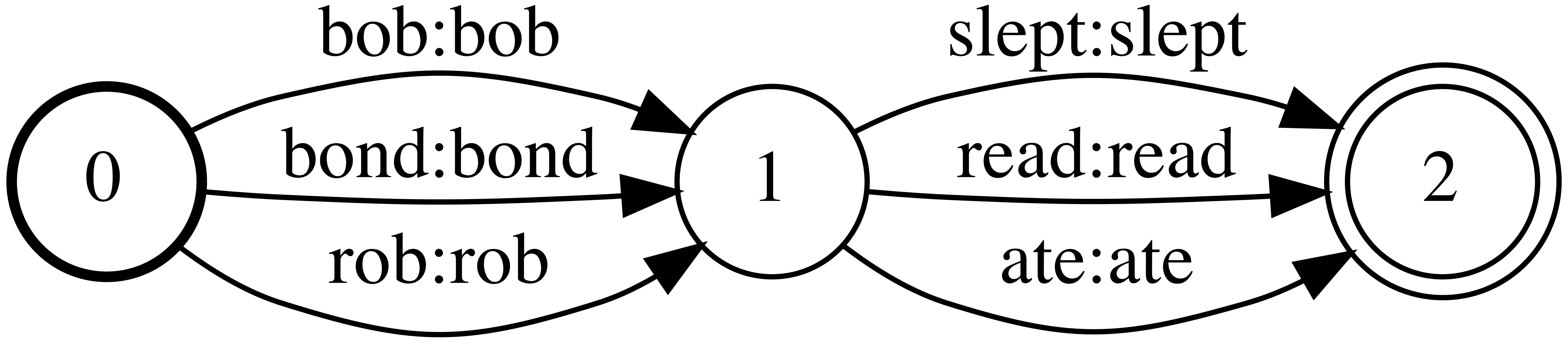
Minimization ensures that the final composed machine has minimum number of states

Final optimization cascade:

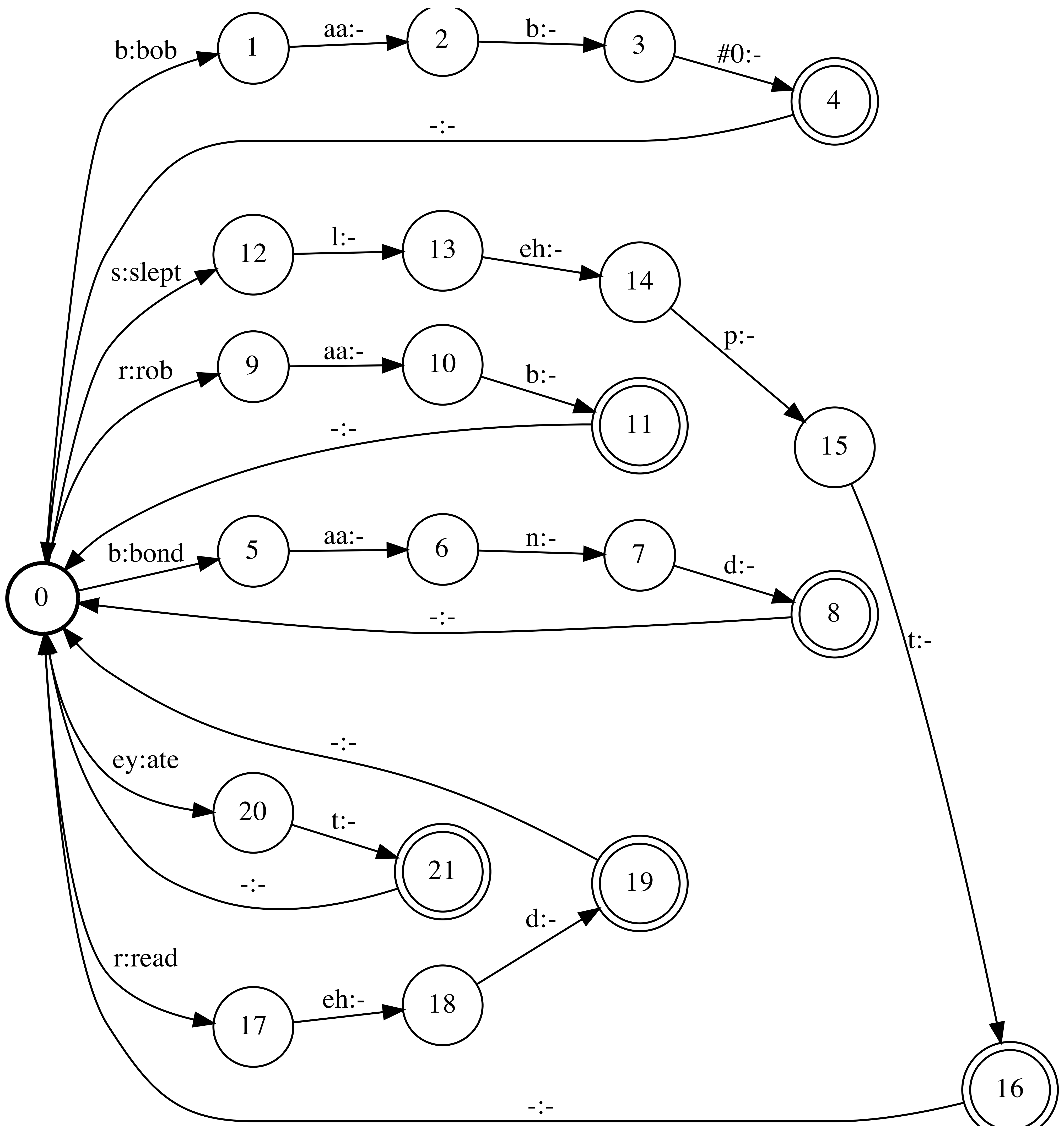
$$N = \pi_\varepsilon(\min(\det(\tilde{H} \circ \det(\tilde{C} \circ \det(\tilde{L} \circ G)))))$$

Replaces disambiguation symbols in input alphabet of $\tilde{H}$ with ε

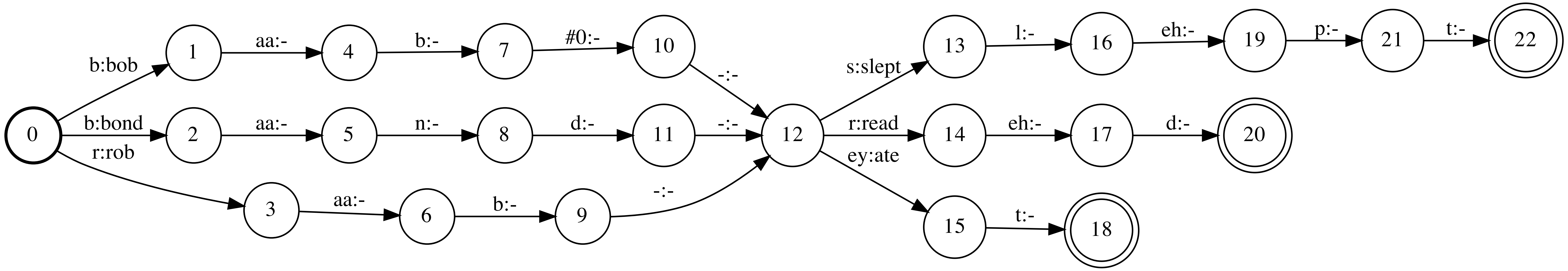
Example G



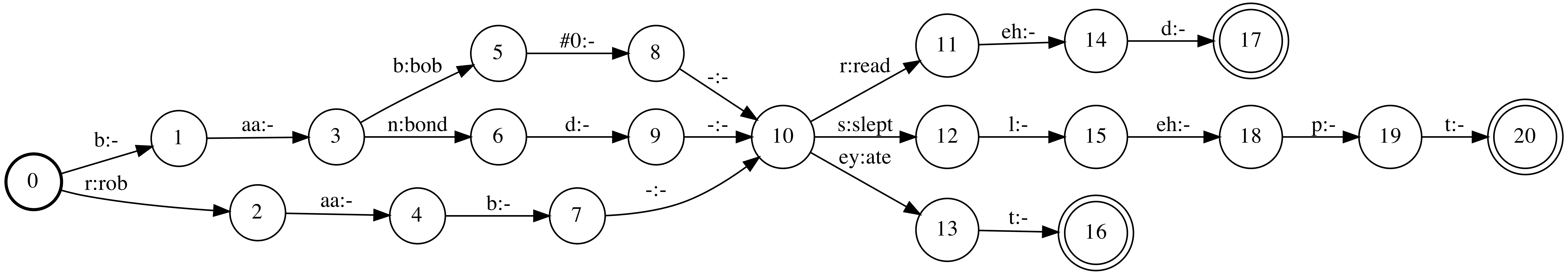
Example L̃ :Lexicon with disambig symbols



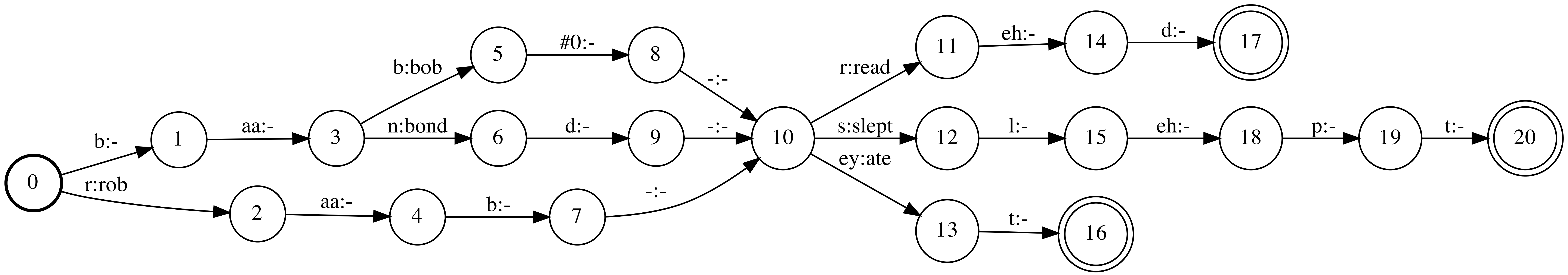
$\tilde{L} \circ G$



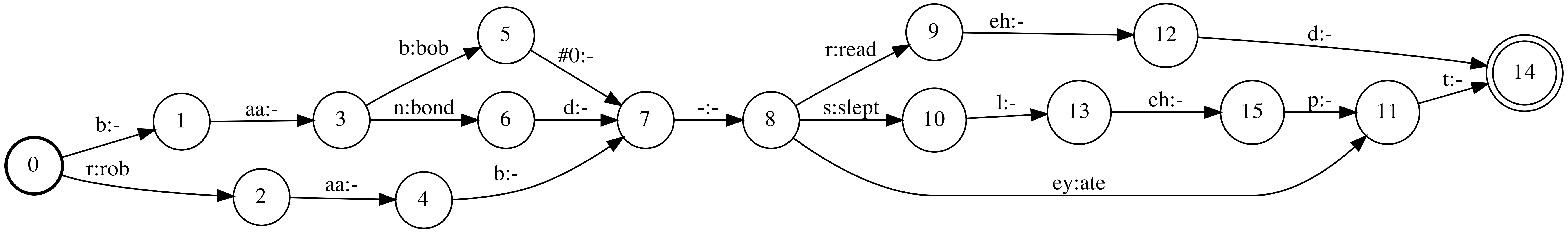
$\det(\tilde{L} \circ G)$



$\det(\tilde{L} \circ G)$



$\min(\det(\tilde{L} \circ G))$



1st pass recognition networks (40K vocab)

transducer	x real-time
$C \circ L \circ G$	12.5
$C \circ det(L \circ G)$	1.2
$det(H \circ C \circ L \circ G)$	1.0

Recognition speeds for systems with an accuracy of 83%

Static and dynamic networks

- What we've seen so far: Static decoding graph
  - $H \circ C \circ L \circ G$
  - Determinize/minimize to make this graph more compact
- Another approach: Dynamic graph expansion
  - Dynamically build the graph with active states on the fly
  - Do on-the-fly composition with the language model G
    - $(H \circ C \circ L) \circ G$

Viterbi search over the large trellis

- Exact search is infeasible for large vocabulary tasks
  - Unknown word boundaries
  - Ngram language models greatly increase the search space
- Solutions
  - Compactly represent the search space using WFST-based optimisations
  - Beam search: Prune away parts of the search space that aren't promising

Beam pruning

- At each time-step t, only retain those nodes in the time-state trellis that are within a fixed threshold δ (beam width) of the best path

- Given active nodes from the last time-step:

  - Examine nodes in the current time-step …

  - … that are reachable from active nodes in the previous time-step

  - Get active nodes for the current time-step by only retaining nodes with hypotheses that score close to the score of the best hypothesis

Viterbi beam search decoder

- Time-synchronous search algorithm:
  - For time t, each state is updated by the best score from all states in time t-1
- Beam search prunes unpromising states at every time step.
- At each time-step t, only retain those nodes in the time-state trellis that are within a fixed threshold δ (beam width) of the score of the best hypothesis.

Beam search algorithm

Initialization: current states := initial state

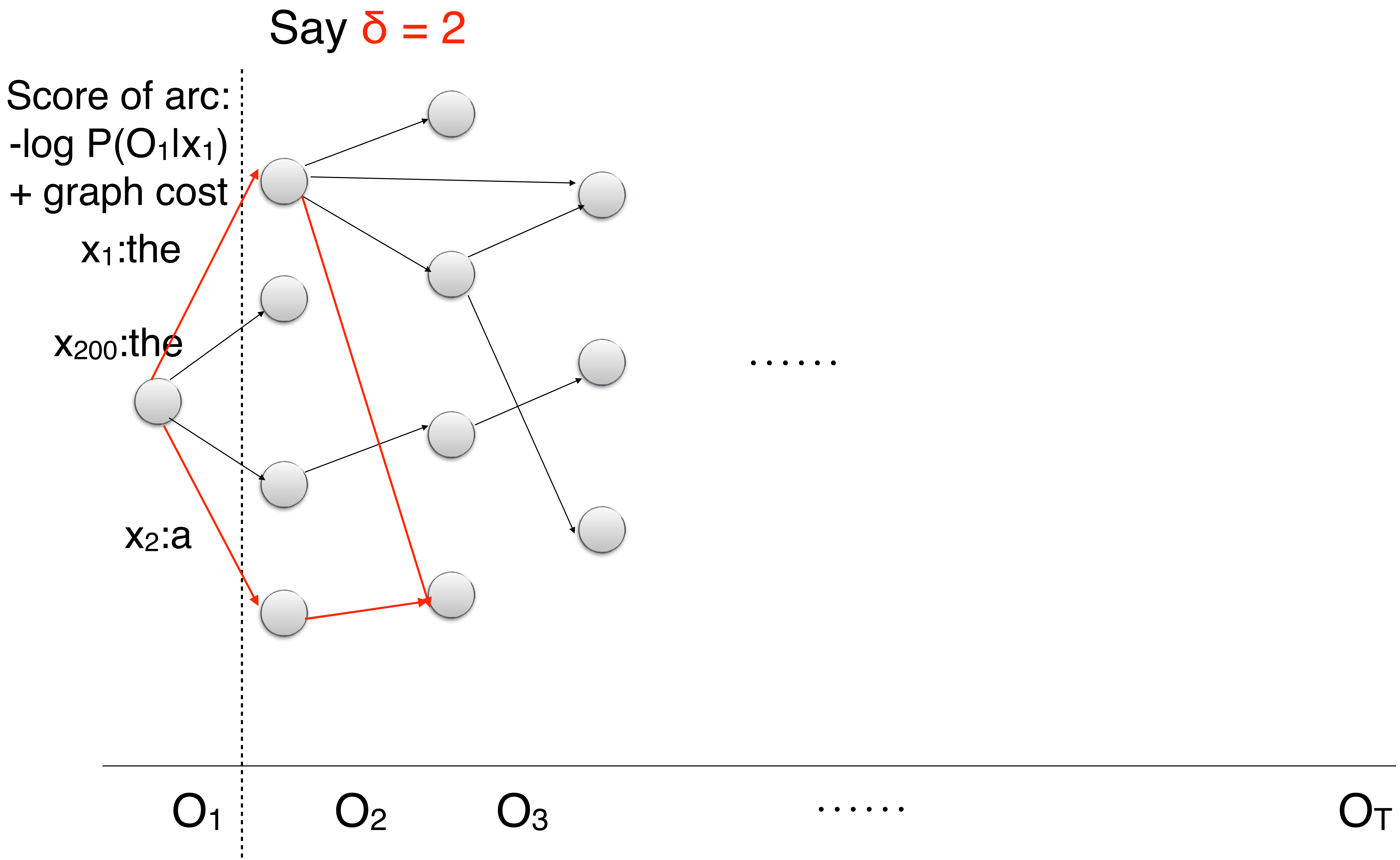
while (current states do not contain the goal state) do:

successor states := NEXT(current states)
    where NEXT is next state function

score the successor states

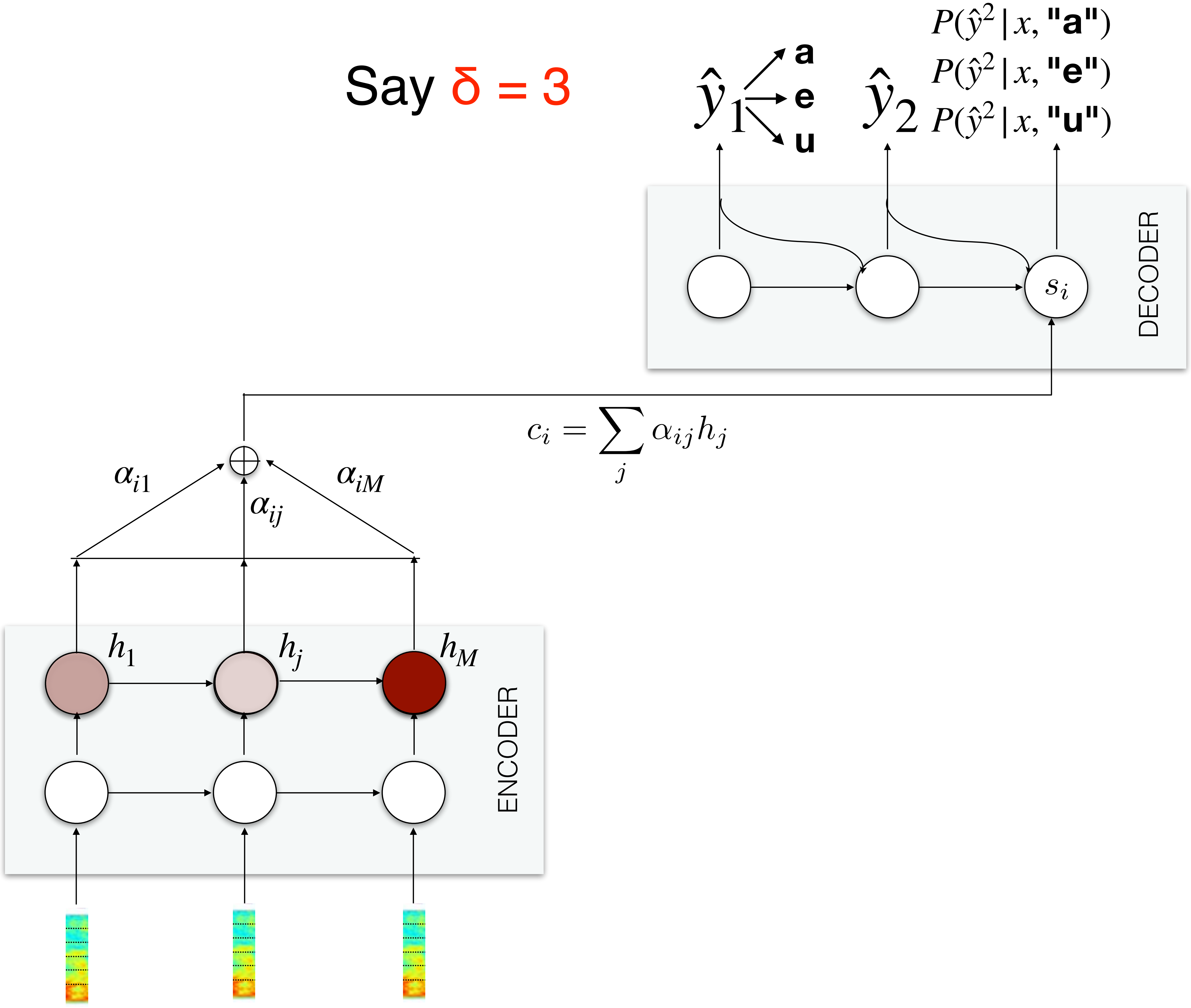
set current states to a pruned set of successor states using beam width δ
    only retain those successor states that are within δ times the best path weight

Beam search over the decoding graph



Beam search in a seq2seq model

Say δ = 3



Lattices

- "**Lattices**" are useful when more than one hypothesis is desired from a recognition pass
- A lattice is a weighted, directed acyclic graph which encodes a large number of ASR hypotheses weighted by acoustic model +language model scores specific to a given utterance

Lattice Generation

- Say we want to decode an utterance, U, of T frames.
- Construct a sausage acceptor for this utterance, X, with T+1 states and arcs for each context-dependent HMM state at each time-step
- Search the following composed machine for the best word sequence corresponding to U:

$$D = X \circ HCLG$$

Lattice Generation

- For all practical applications, we have to use beam pruning over D such that only a subset of states/arcs in D are visited. Call this resulting pruned machine, B.

- Word lattice, say L, is a further pruned version of B defined by a lattice beam, β. L satisfies the following requirements:

  - L should have a path for every word sequence within β of the best-scoring path in B

  - All scores and alignments in L correspond to actual paths through B

  - L does not contain duplicate paths with the same word sequence