Decoding

Bhiksha Raj and Rita Singh

Recap and Lookahead

- Covered so far:
  - String Matching based Recognition
  - Introduction to HMMs
  - Recognizing Isolated Words
  - Learning word models from continuous recordings
  - Building word models from phoneme models
  - Context-independent and context-dependent models
  - Building decision trees
  - Tied-state models
  - Language Modelling

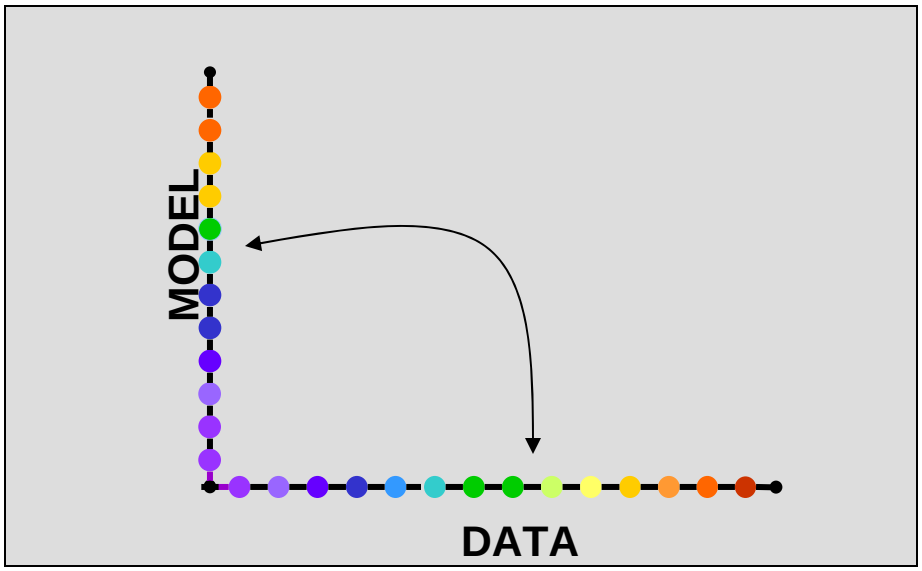
  - Exercise: Training phoneme models
  - Exercise: Training context-dependent models
  - Exercise: Building decision trees
  - Exercise: Training tied-state models

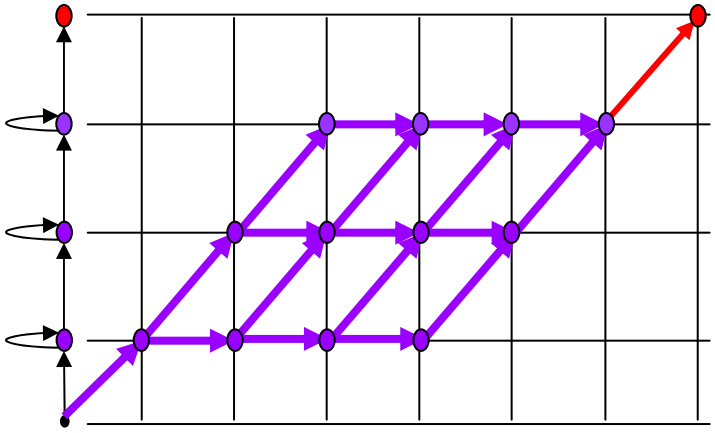
- Recognition

Analogy between DTW vs. HMM

- DTW: Edge Score; HMM: Transition probability
  - The edge score of the DTW template is analogous to the *log* of the transition probability for the HMM
- DTW: Symbol matching cost; HMM: State probability
  - The matching cost of DTW is analogous to the *log* of the probability of the observation computed from the probability distribution associated with the state
- The string matching algorithm for DTW actually finds the sequence of states in the HMM that matches the observation

Speech Recognition as String Matching





- We find the distance of the data from the "model" using the Trellis for the word
- Pick the word for which this distance is lowest
- Word = argmin $_{word}$ distance(data, model(word))
- Using the DTW / HMM analogy
  - Word = argmax $_{word}$ probability(data | model(word))

Speech Recognition as Bayesian Classification

- Different words may occur with different frequency
  - E.g. a person may say “SEE” much more frequently than “ZEE”
- This must be factored in
  - If we are not very sure they said “SEE” or “ZEE”, choose “SEE”
    - We are more likely to be right than if we chose ZEE
- The basic DTW equation does not factor this in
  - Word = argmax $_{word}$ probability(data | model(word)) does not account for prior bias
- Cast the problem instead as a Bayesian classification problem
  - Word = argmax $_{word}$ p(word) probability(data | model(word))
  - “p(word)” is the *a priori* probability of the word
  - Naturally accounts for prior bias

Statistical pattern classification

- Given data X, find which of a number of classes C_1, C_2,…C_N it belongs to, based on known distributions of data from C_1, C_2, etc.
  - Bayesian Classification:

$$\text{Class} = C_i : i = \text{argmax}_j \log(P(C_j)) + \log(P(X|C_j))$$

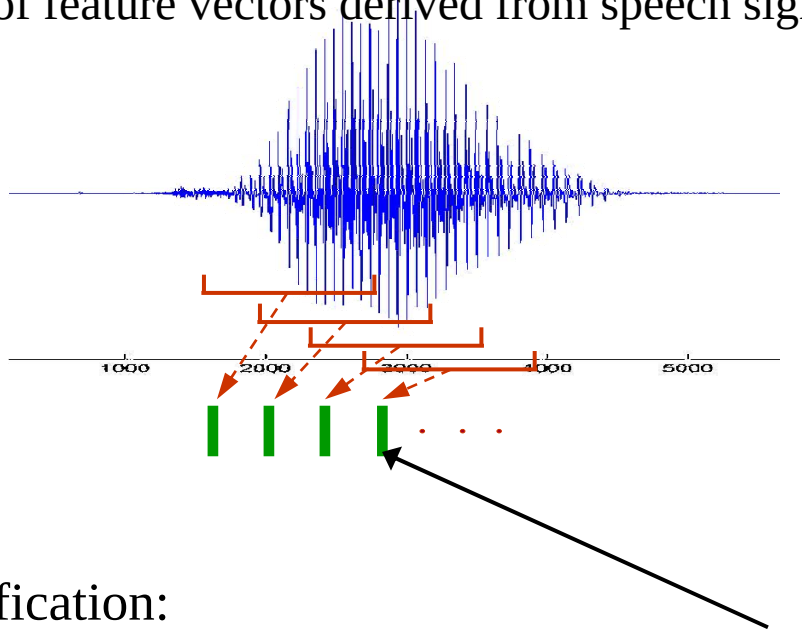
a priori probability of C_j

Probability of X as given by the probability distribution of C_j

- The *a priori* probability accounts for the relative proportions of the classes
  - If you never saw any data, you would guess the class based on these probabilities alone
- $P(X|C_j)$ accounts for evidence obtained from observed data X

Statistical classification of isolated words

- Classes are words
- Data are instances of isolated spoken words
  - Sequence of feature vectors derived from speech signal,



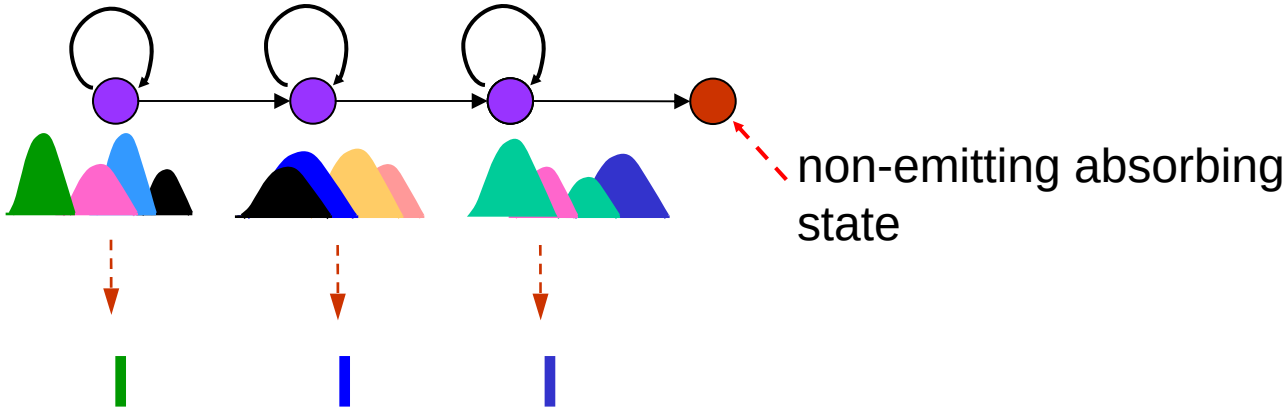
- Bayesian classification:

Recognized_Word = argmax_{word} log(P(*word*)) + log(P(*X*| *word*))

- P(*word*) is *a priori* probability of *word*
  - Obtained from our expectation of the relative frequency of occurrence of the word
- P(*X*|*word*) is the probability of *X* computed on the probability distribution function of *word*

Computing P(X|*word*)

- To compute P(*X*|*word*), there must be a statistical distribution for *X* corresponding to *word*
  - Each word must be represented by some statistical model.
- We represent each word by an HMM
- P(X|word) is (approximated by) the best path score for the the HMM

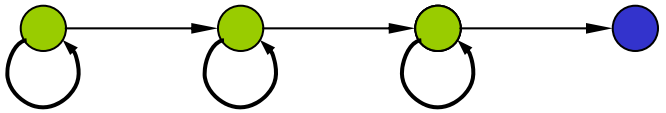


- Useful to model the termination of the word using a non-emitting state
  - Simplifies many things

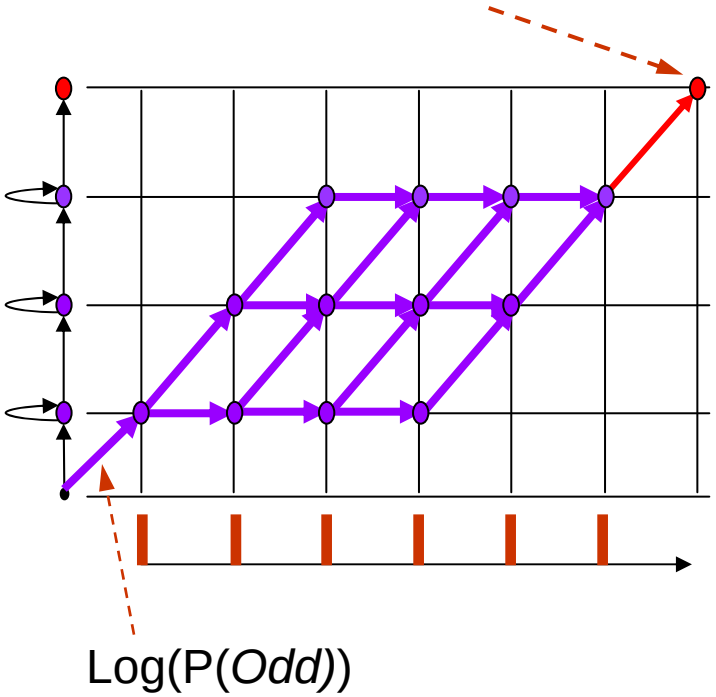
Classifying between two words: *Odd* and *Even*

HMM for *Odd*

HMM for *Even*

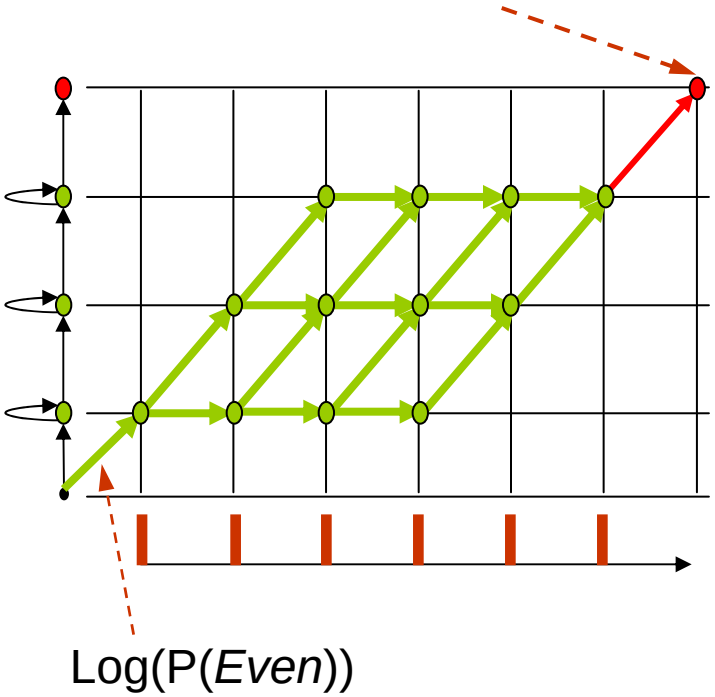


Log(P(*Odd*))+ log(P(*X*|*Odd*))



Log(P(*Odd*))

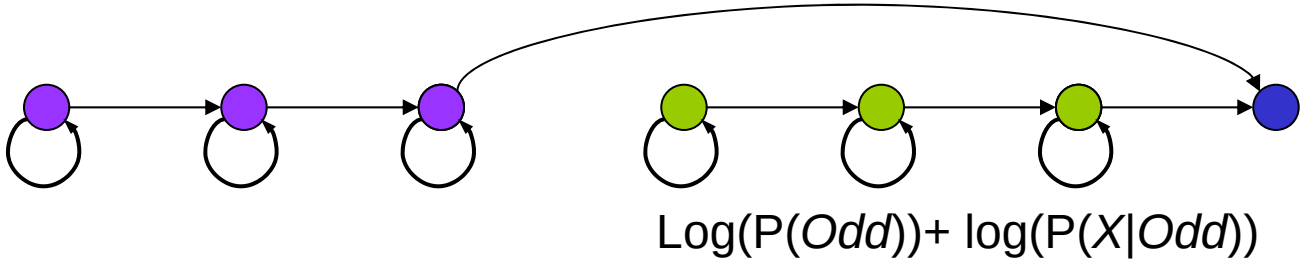
Log(P(*Even*))+log(P(*X*|*Even*))



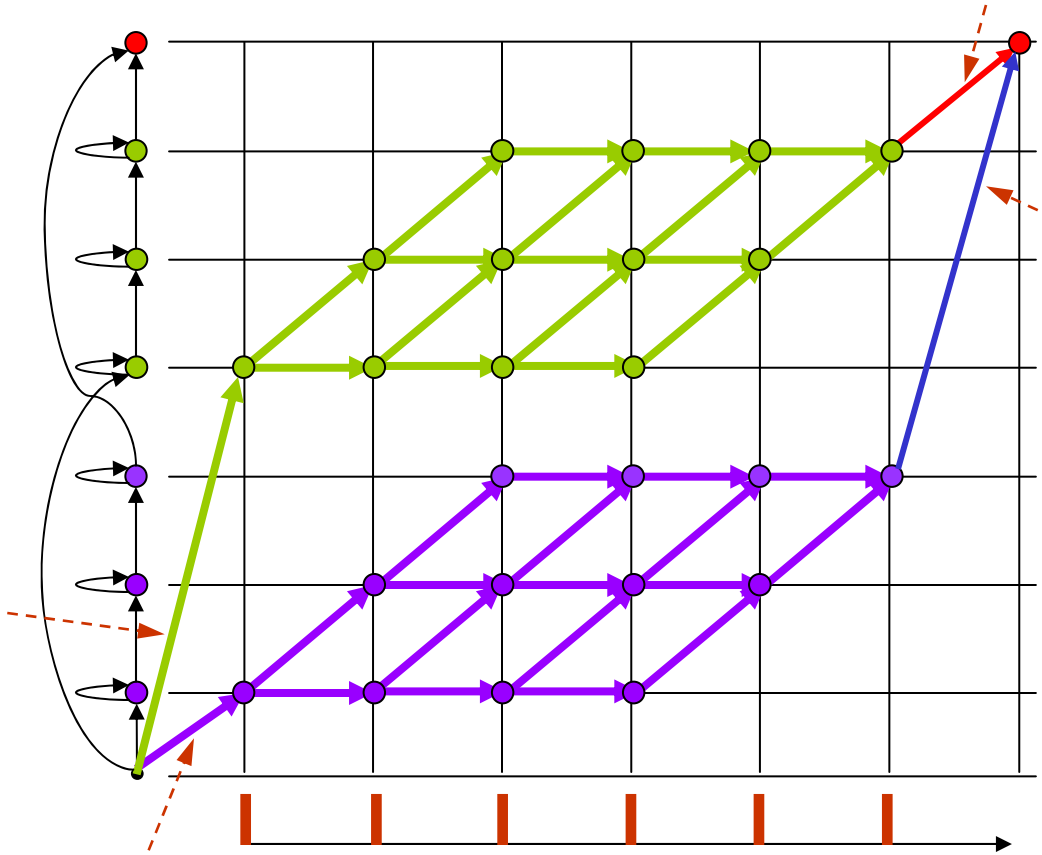
Log(P(*Even*))

- **The prior bias is factored in as the edge penalty at the entry to the trellis**

Classifying between two words: *Odd* and *Even*



Log(P(*Odd*))+ log(P(*X*|*Odd*))



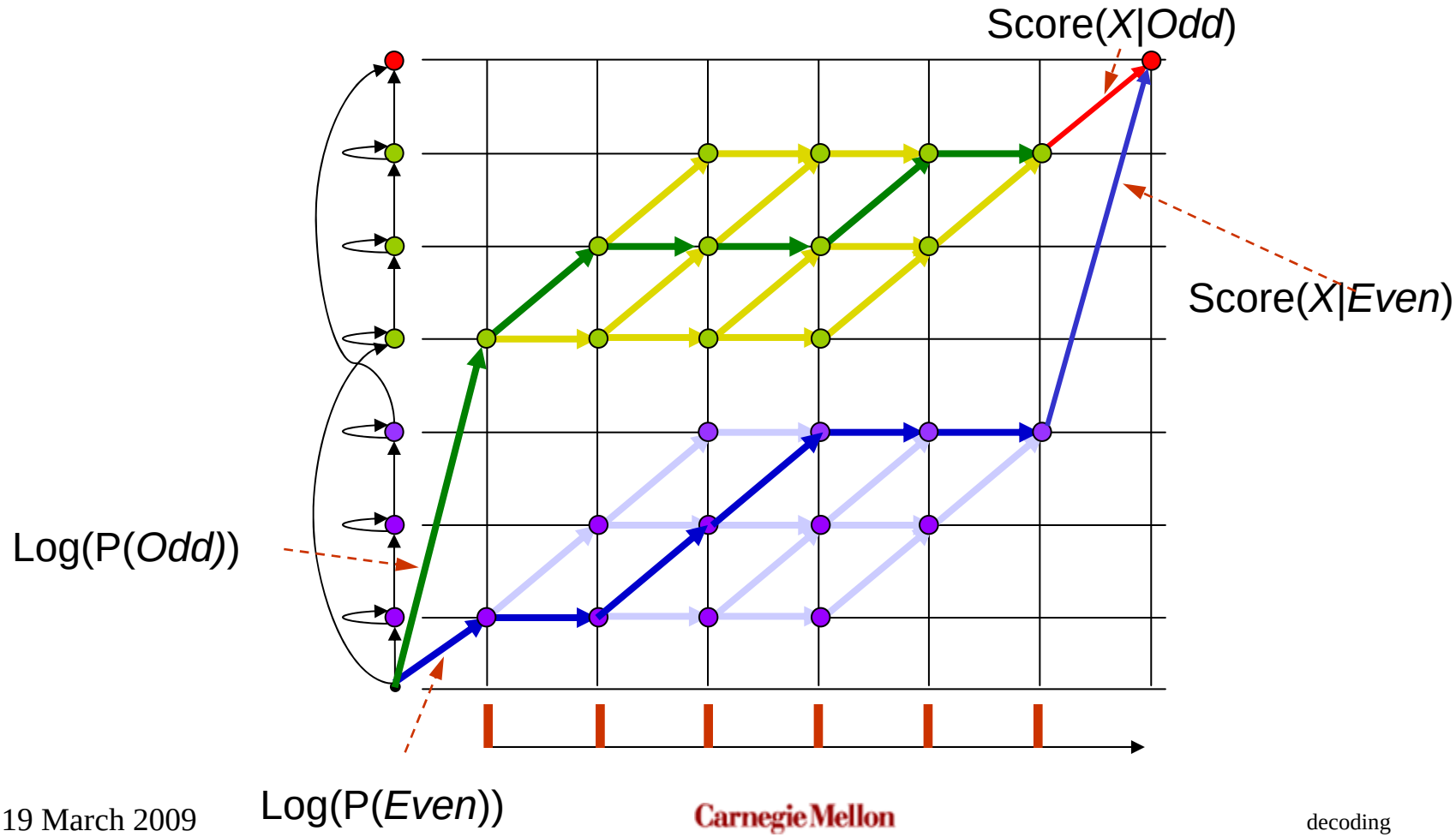
Log(P(*Even*))
+log(P(*X*|*Even*))

Log(P(*Odd*))

Log(P(*Even*))

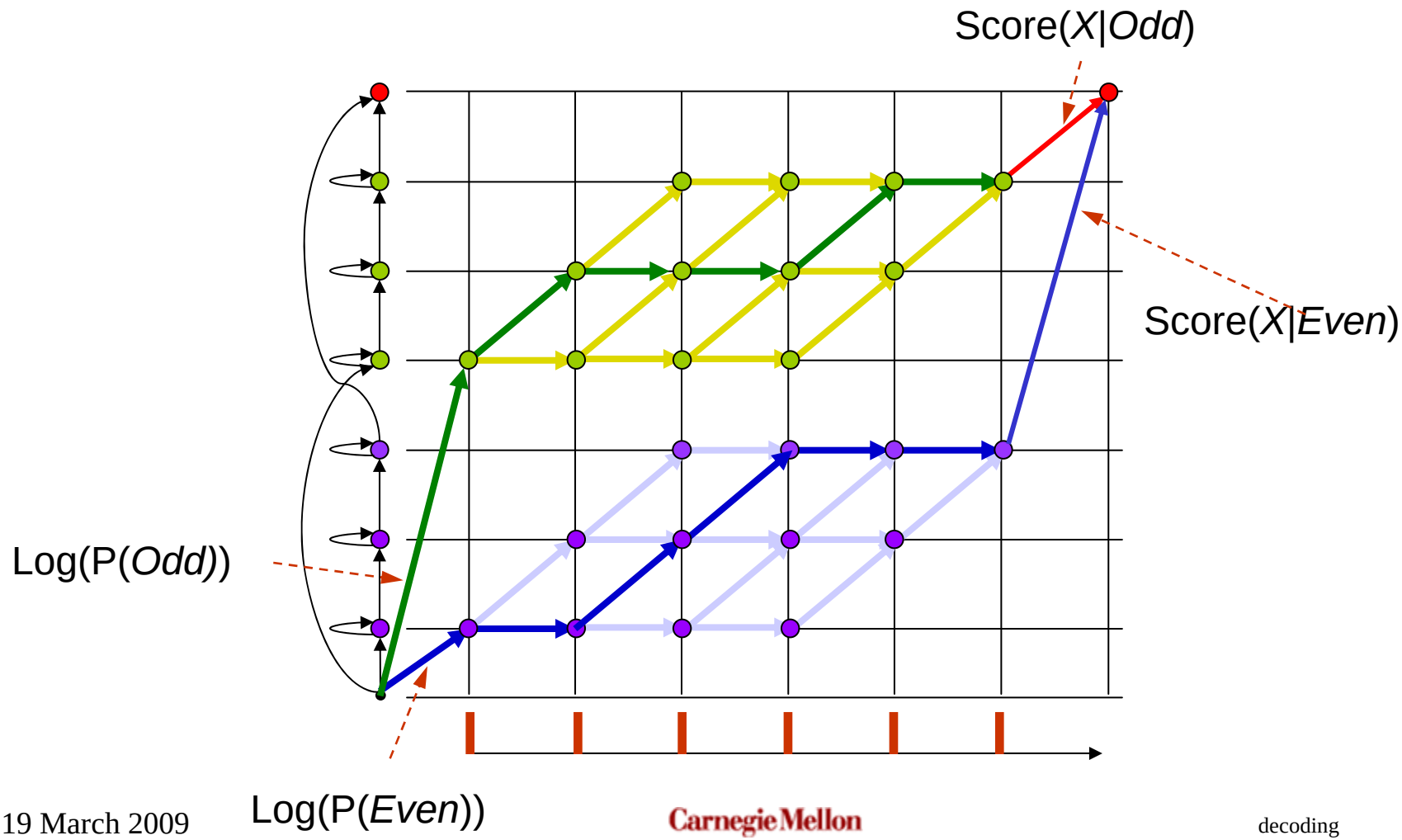
Decoding to classify between *Odd* and *Even*

- Compute the probability of best path
  - Computations can be done in the log domain. Only additions and comparisons are required



Decoding to classify between *Odd* and *Even*

- Compare scores (best state sequence probabilities) of all competing words
- Select the word sequence corresponding to the path with the best score



Decoding isolated words with word HMMs

- Construct a trellis (search graph) based on the HMM for each word
  - Alternately construct a single, common trellis
- Select the word corresponding to the best scoring path through the combined trellis

A change of notation

- Thus far we have been talking about *Scores*, that are in fact *Log Probabilities*
- In the following slides we will talk in terms of Probabilities and not Log probabilities
  - A matter of convenience
  - This does not change the basic procedures – what used to be summation will not become multiplication
    - Ie. We multiply the probabilities along the best path, rather than to add them

- Bayesian classification:

  $\text{Recognized\_Word} = \text{argmax}_{word} \log(P(word)) + \log(P(X|\ word))$

- $\text{Recognized\_Word} = \text{argmax}_{word} P(word)\ P(X|\ word)$

Statistical classification of word sequences

- Given data *X*, find which of a number of classes C_1, C_2,…C_N it belongs to, based on known distributions of data from C_1, C_2, etc.
- Bayesian Classification:

$$\text{Class} = C_i : i = \text{argmax}_j \, P(C_j)P(X|C_j)$$

- Classes are word sequences
- Data are spoken recordings of word sequences
- Bayesian classification

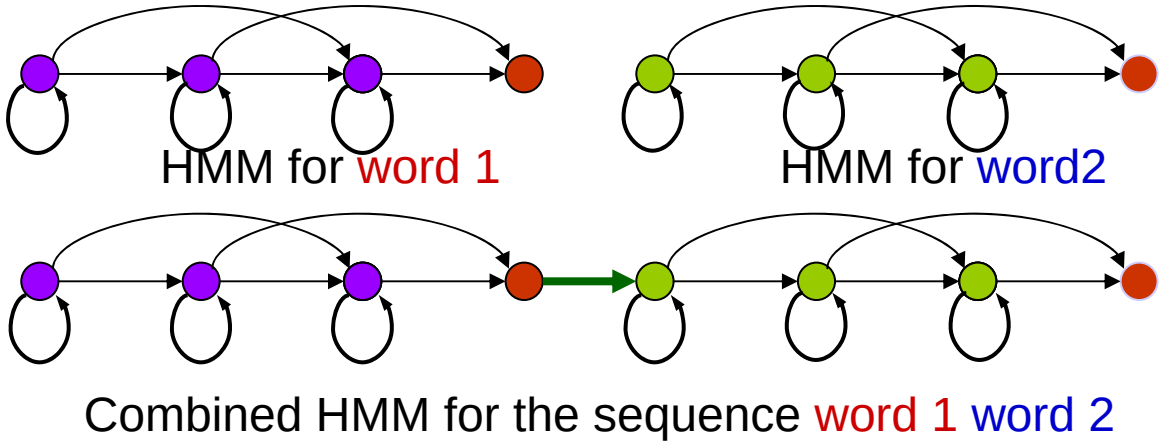
$$word_1, word_2, ..., word_N =$$
$$\arg\max_{wd_1, wd_2, ..., wd_N} \{P(X \mid wd_1, wd_2, ..., wd_N)P(wd_1, wd_2, ..., wd_N)\}$$

- $P(wd_1,wd_2,wd_3..)$ is *a priori* probability of word sequence $wd_1,wd_2,wd_3..$
  - Obtained from a model of the language

- $P(X| wd_1,wd_2,wd_3..)$ is the probability of *X* computed on the probability distribution function of the word sequence $wd_1,wd_2,wd_3..$
  - HMMs now represent probability distributions of word sequences

Decoding continuous speech

First step: construct an HMM for each possible word sequence

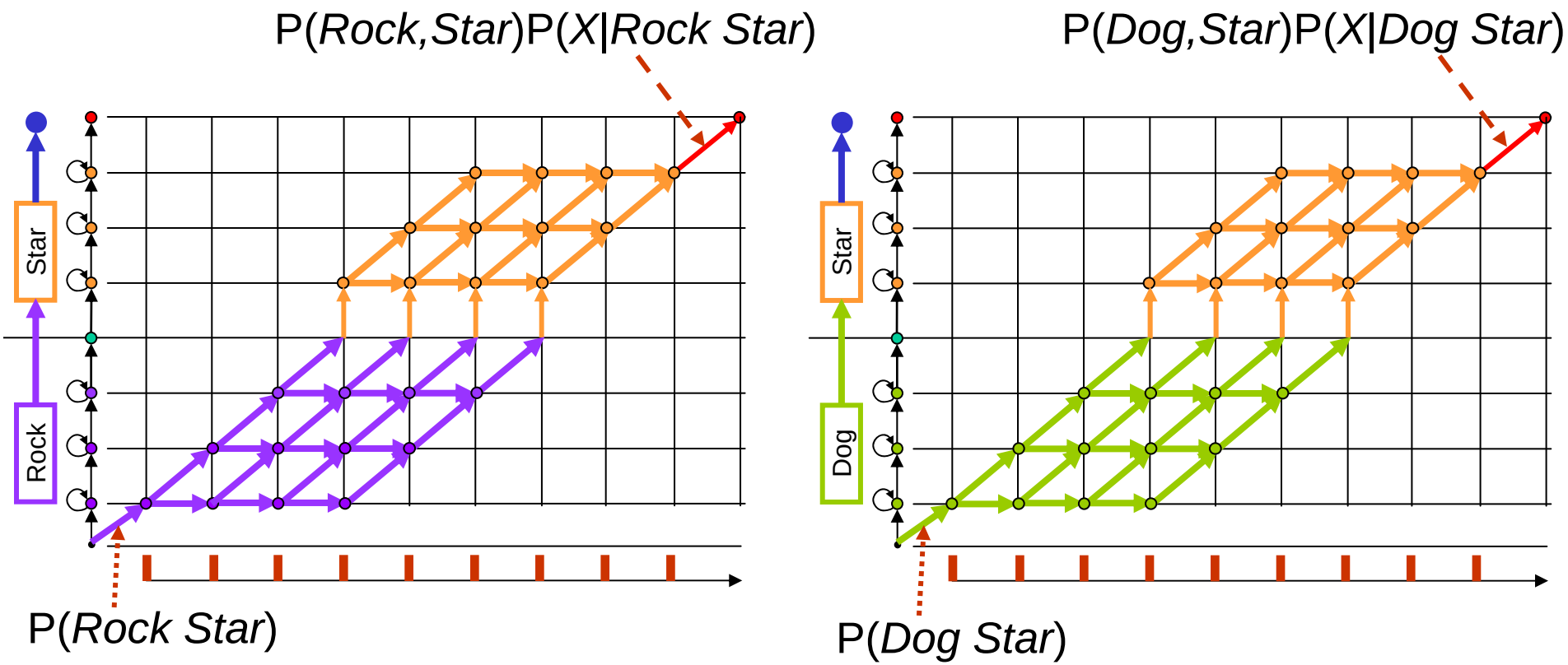


Second step: find the probability of the given utterance on the HMM for each possible word sequence

- P(X| wd_1,wd_2,wd_3..) is the probability of X computed on the probability distribution function of the word sequence wd_1,wd_2,wd_3..
  - HMMs now represent probability distributions of word sequences

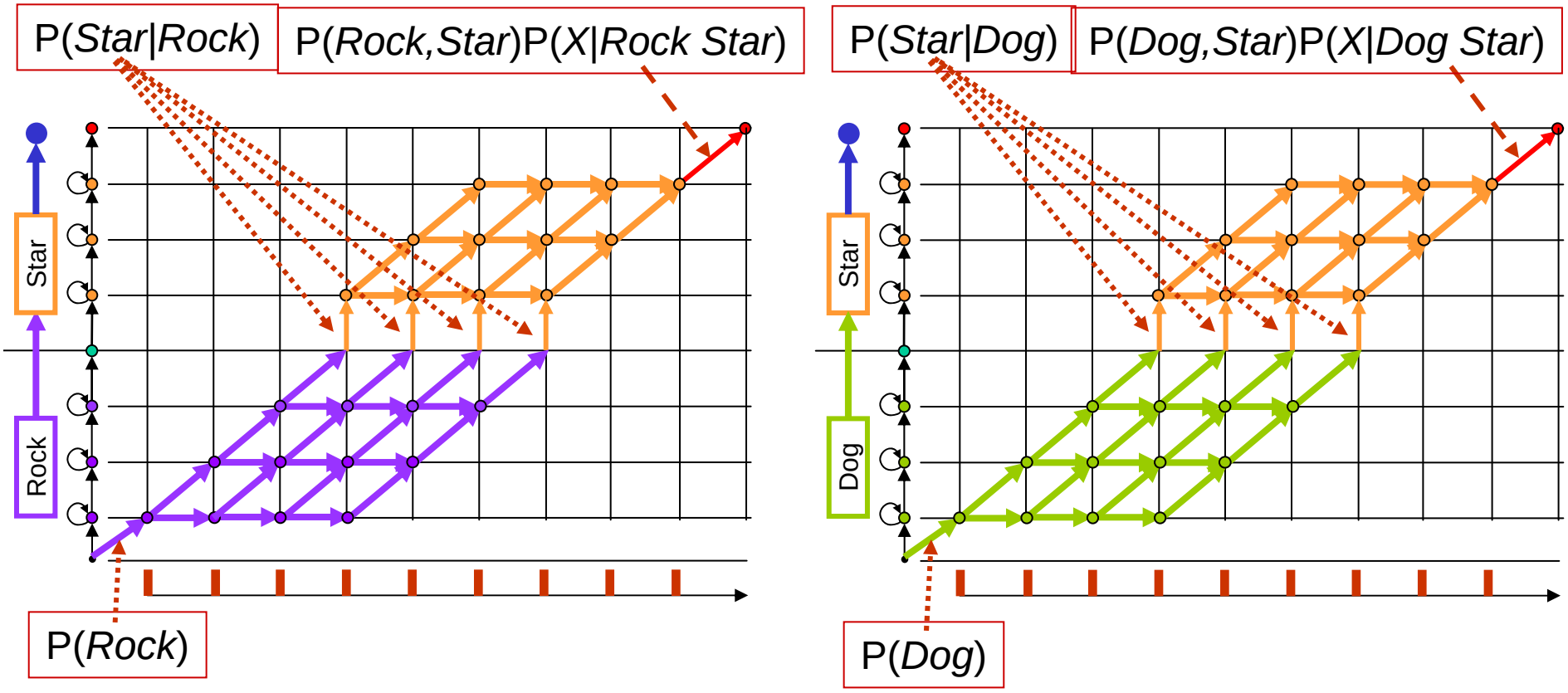
Bayesian Classification between word sequences

- Classifying an utterance as either "Rock Star" or "Dog Star"
  - Must compare P(Rock,Star)P(X|Rock Star) with P(Dog,Star)P(X|Dog Star)

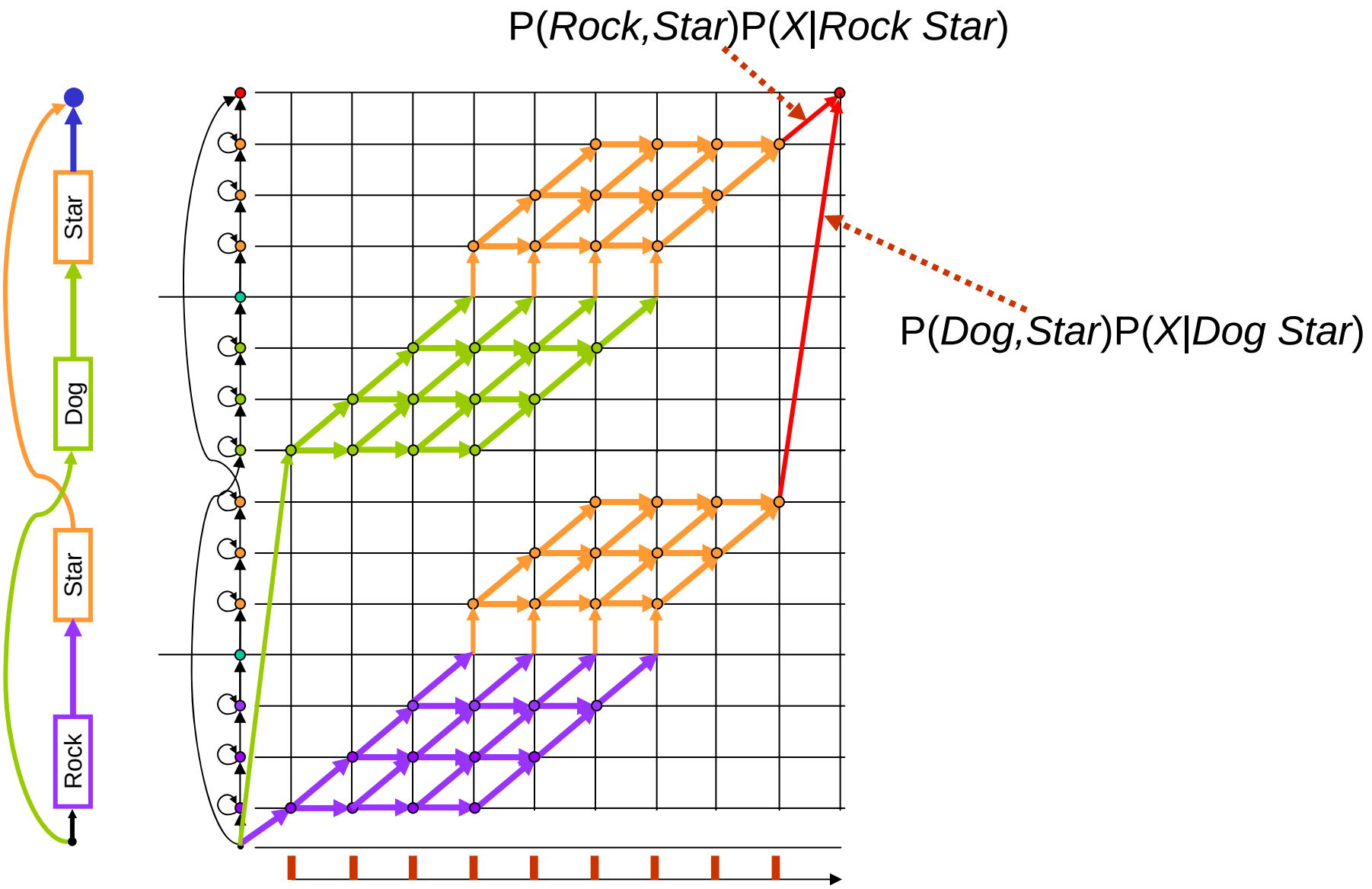


Bayesian Classification between word sequences

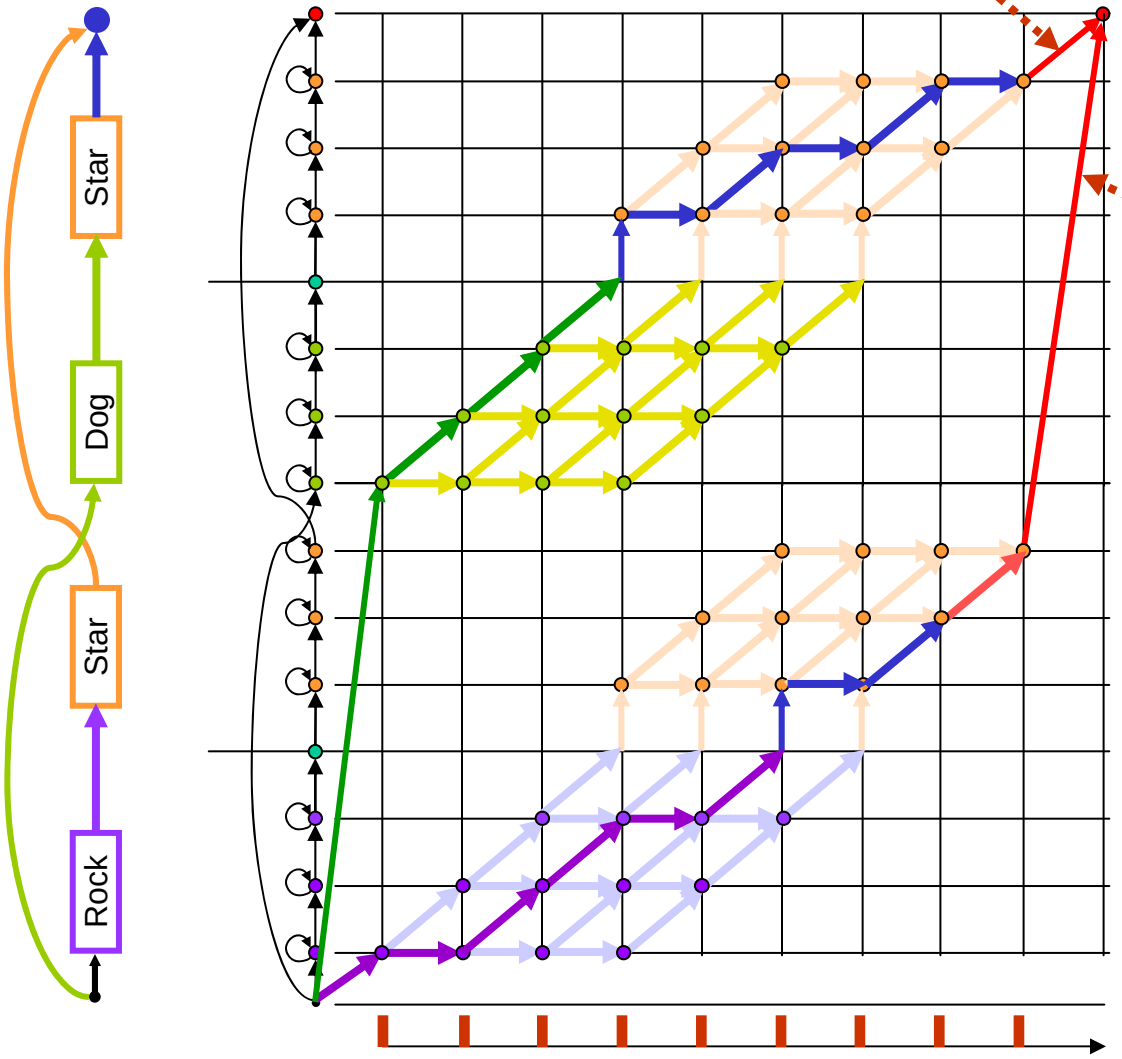
- Classifying an utterance as either “Rock Star” or “Dog Star”
  - Must compare P(Rock,Star)P(X|Rock Star) with P(Dog,Star)P(X|Dog Star)



Bayesian Classification between word sequences

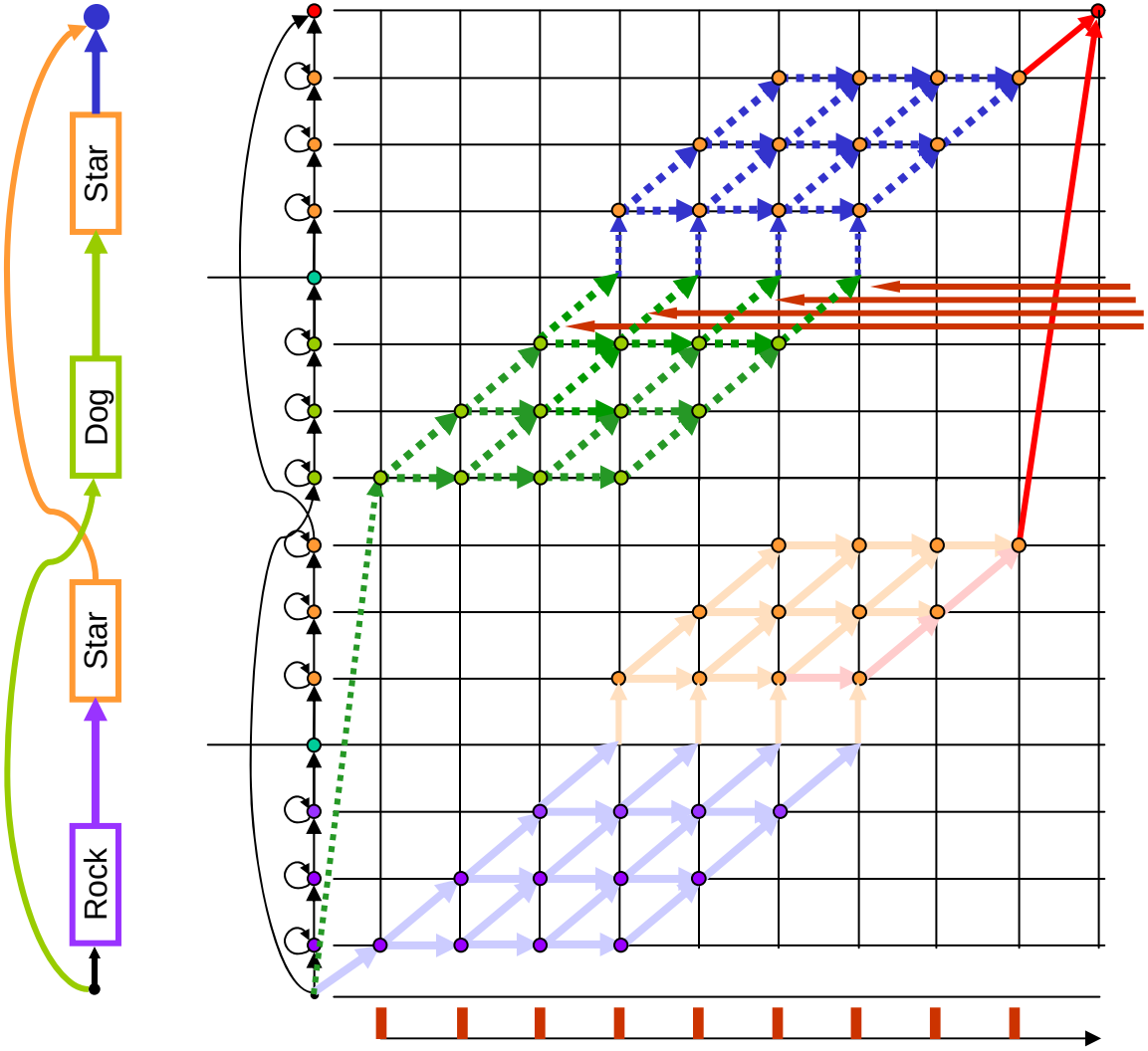


Decoding to classify between word sequences



Approximate total probability with best path score

Decoding to classify between word sequences



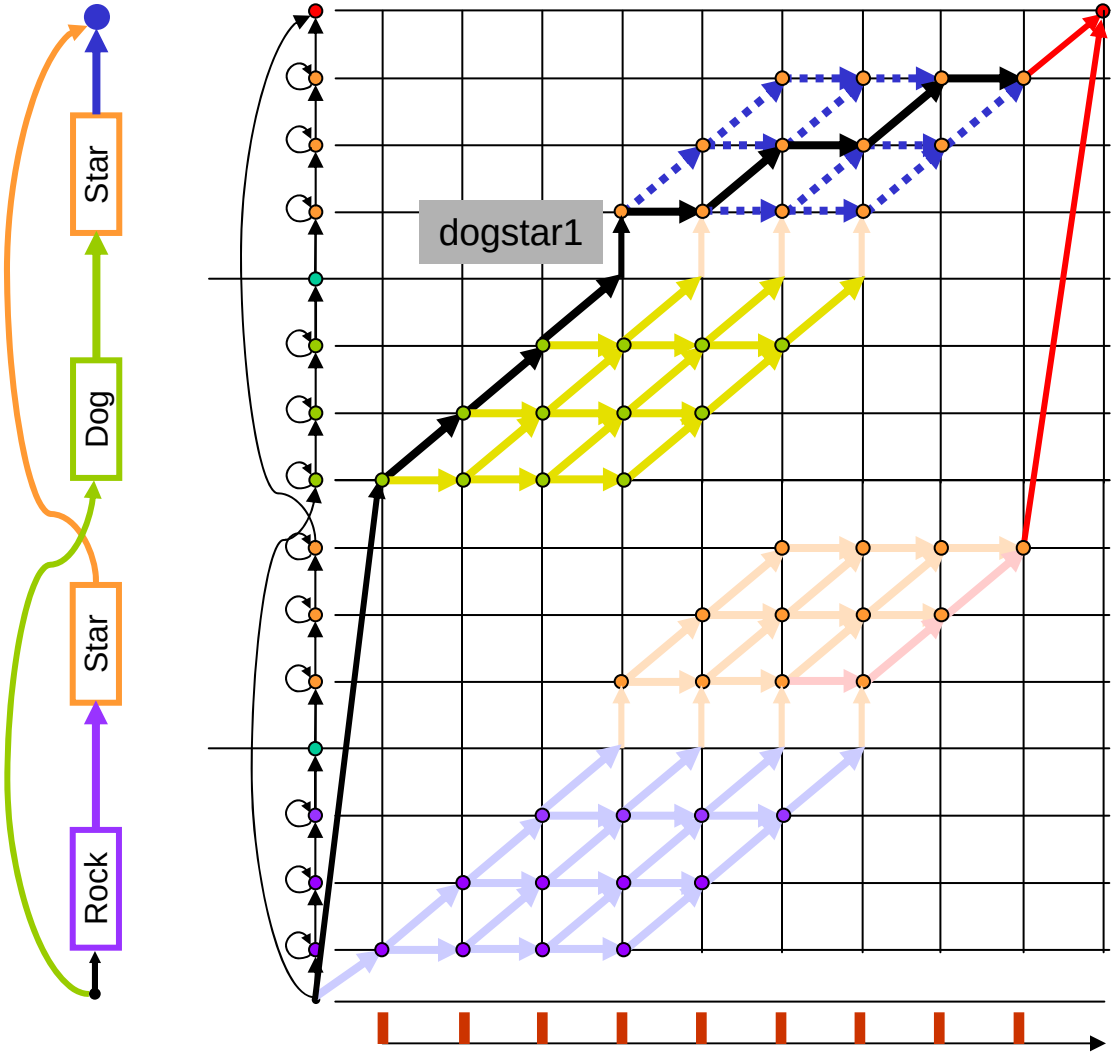
The best path through *Dog Star* lies within the dotted portions of the trellis

There are four transition points from *Dog* to *Star* in this trellis

There are four different sets paths through the dotted trellis, each with its own best path

Decoding to classify between word sequences

SET 1 and its best path



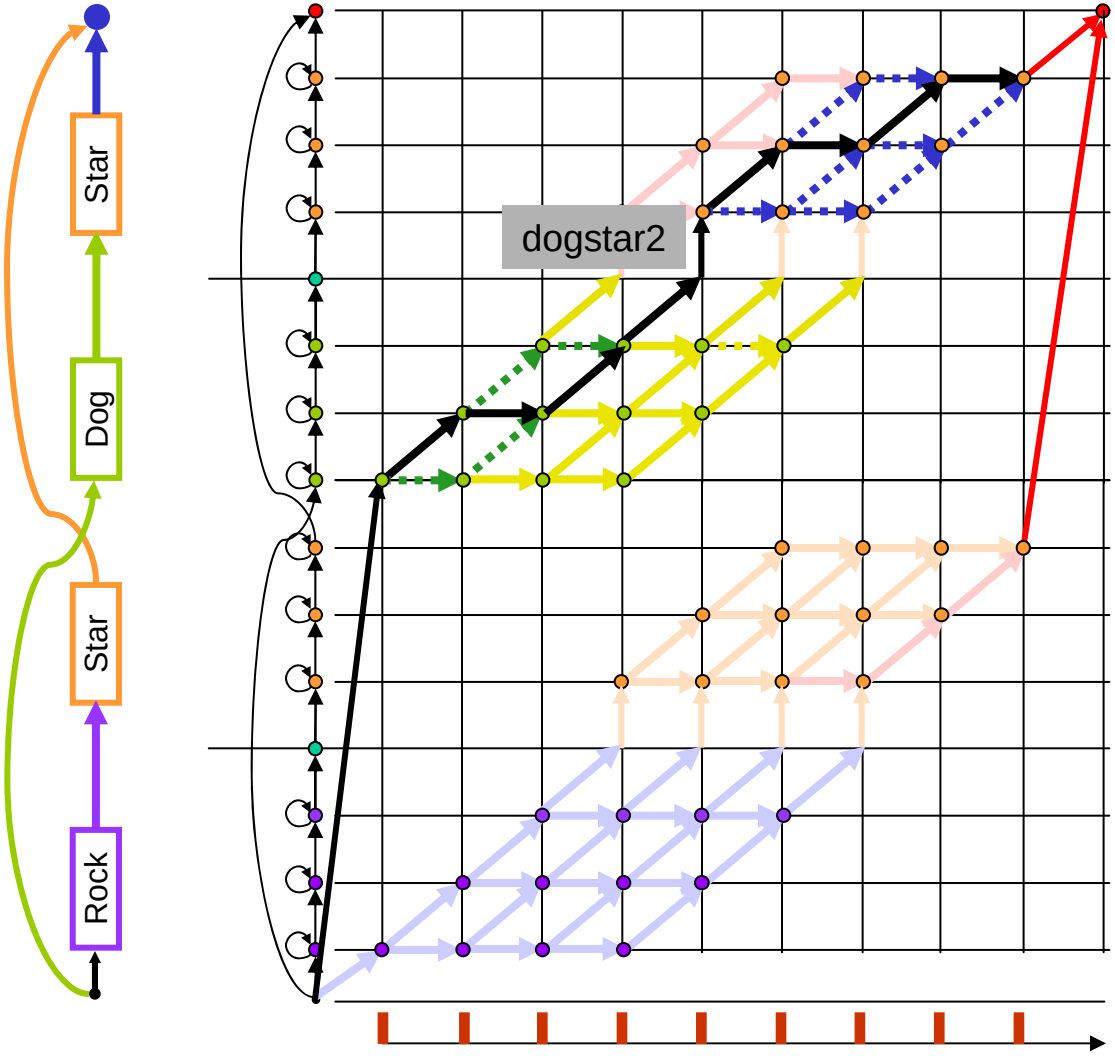
The best path through *Dog Star* lies within the dotted portions of the trellis

There are four transition points from Dog to Star in this trellis

There are four different sets paths through the dotted trellis, each with its own best path

Decoding to classify between word sequences

SET 2 and its best path



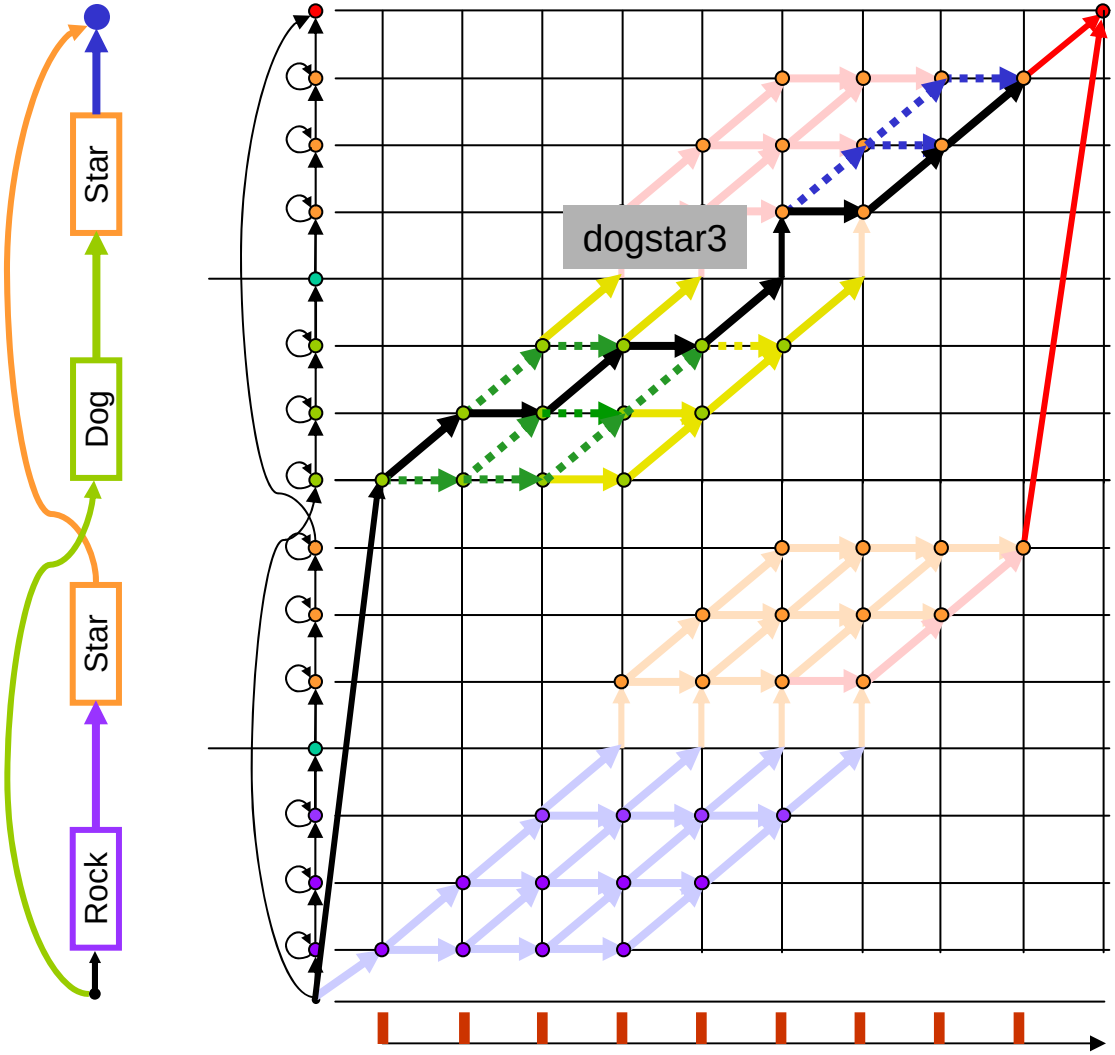
The best path through *Dog Star* lies within the dotted portions of the trellis

There are four transition points from Dog to Star in this trellis

There are four different sets paths through the dotted trellis, each with its own best path

Decoding to classify between word sequences

SET 3 and its best path



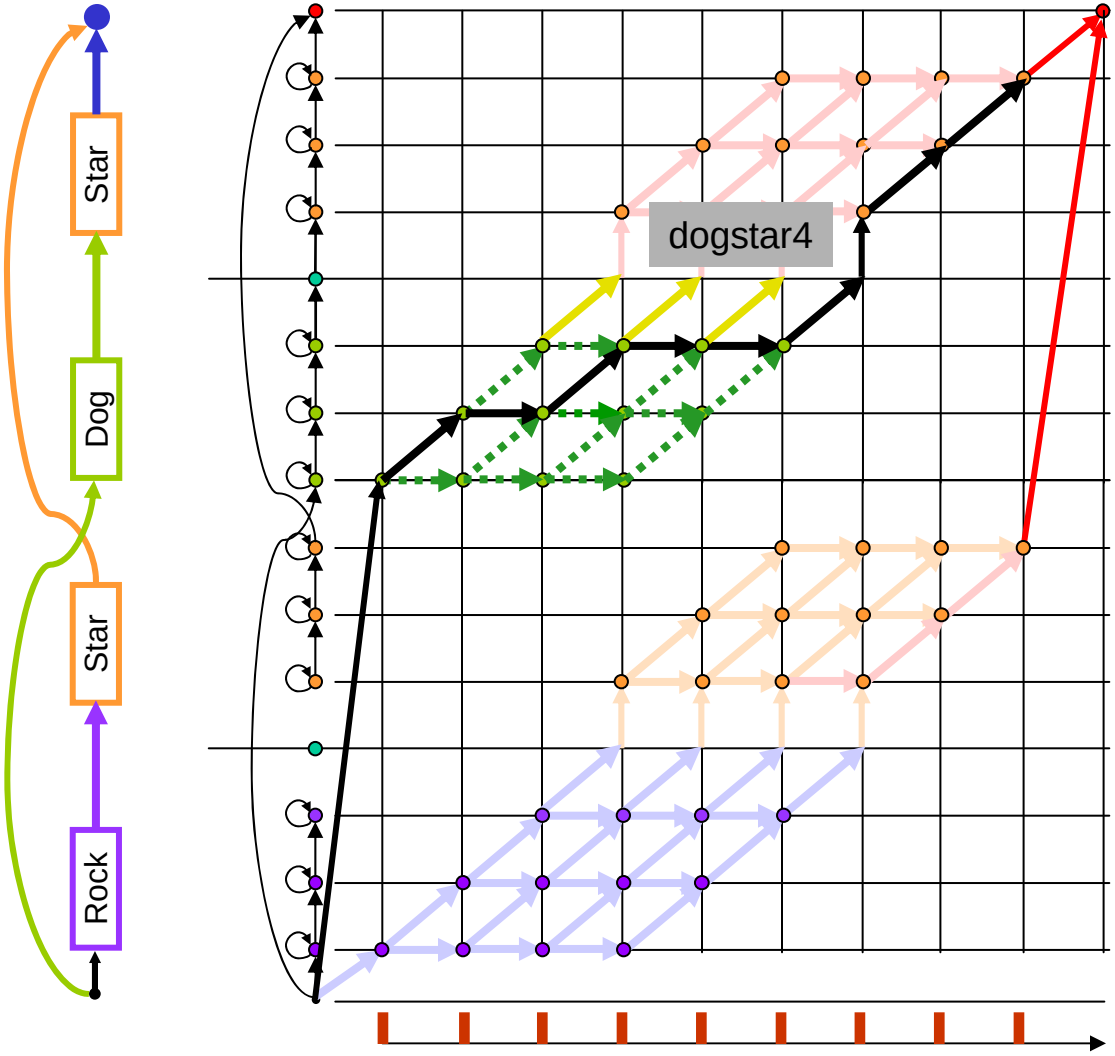
The best path through *Dog Star* lies within the dotted portions of the trellis

There are four transition points from Dog to Star in this trellis

There are four different sets paths through the dotted trellis, each with its own best path

Decoding to classify between word sequences

SET 4 and its best path

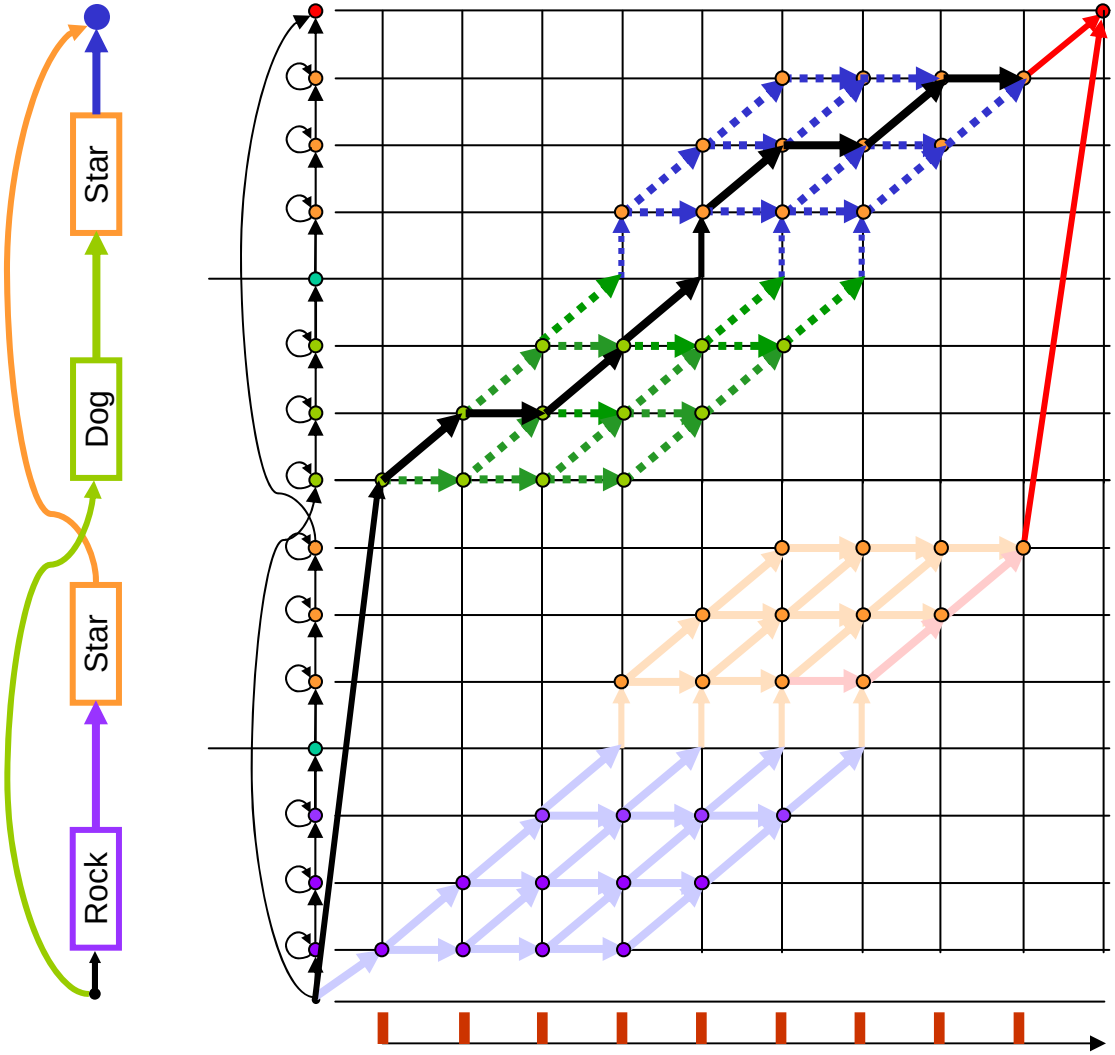


The best path through *Dog Star* lies within the dotted portions of the trellis

There are four transition points from Dog to Star in this trellis

There are four different sets paths through the dotted trellis, each with its own best path

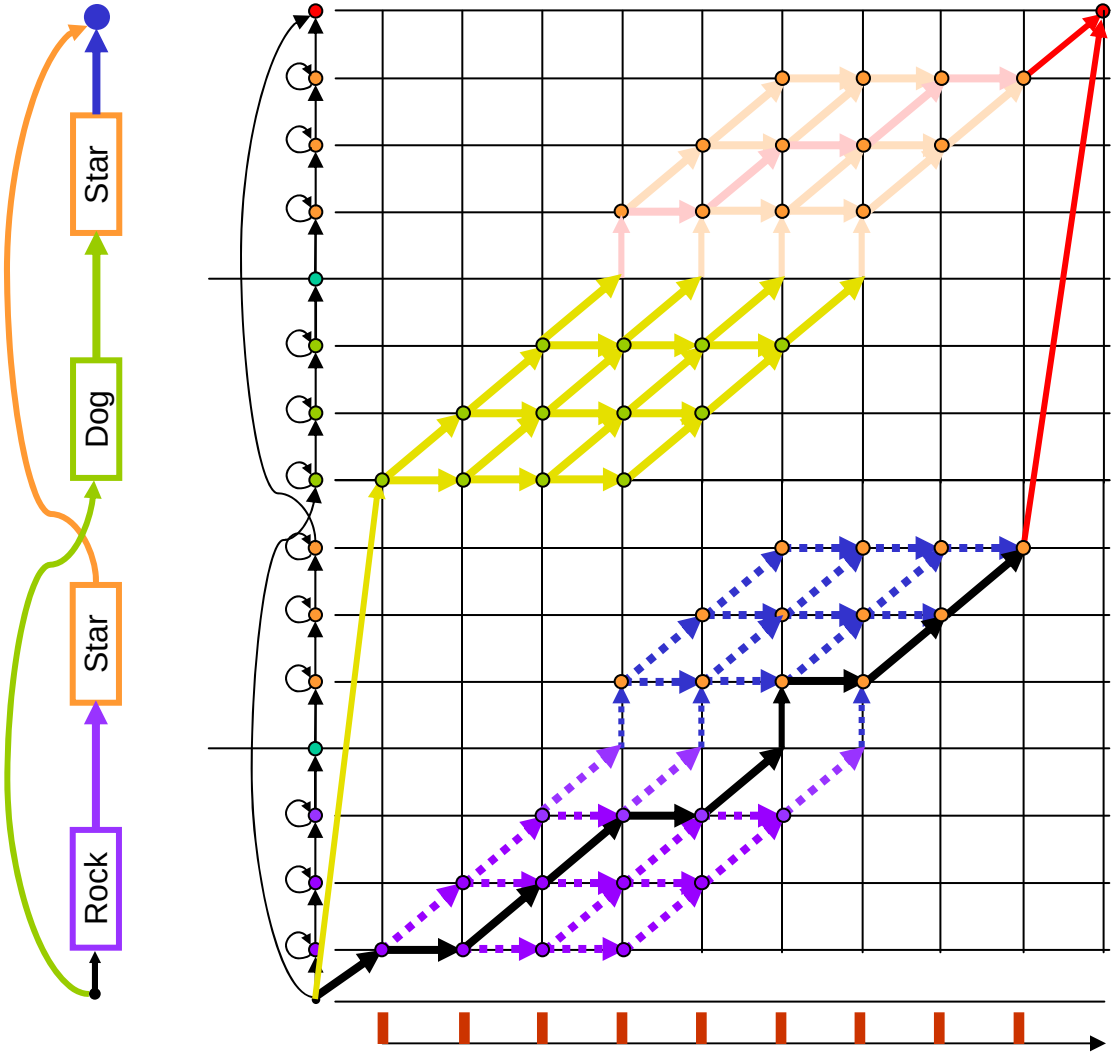
Decoding to classify between word sequences



The best path through *Dog Star* is the best of the four transition-specific best paths

max(dogstar) =
max (dogstar1, dogstar2,
dogstar3, dogstar4)

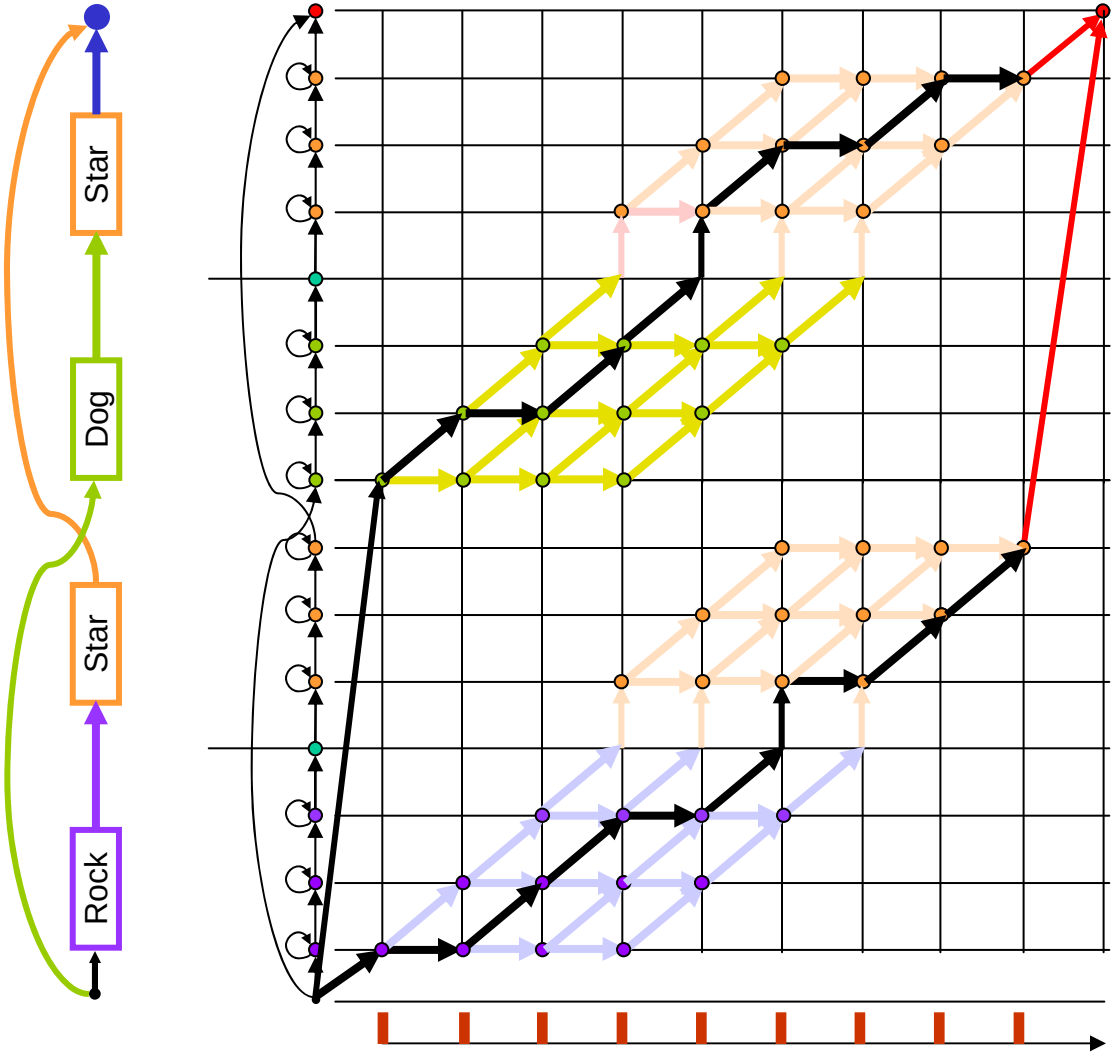
Decoding to classify between word sequences



Similarly, for *Rock Star* the best path through the trellis is the best of the four transition-specific best paths

max(rockstar) =
max (rockstar1, rockstar2,
rockstar3, rockstar4)

Decoding to classify between word sequences



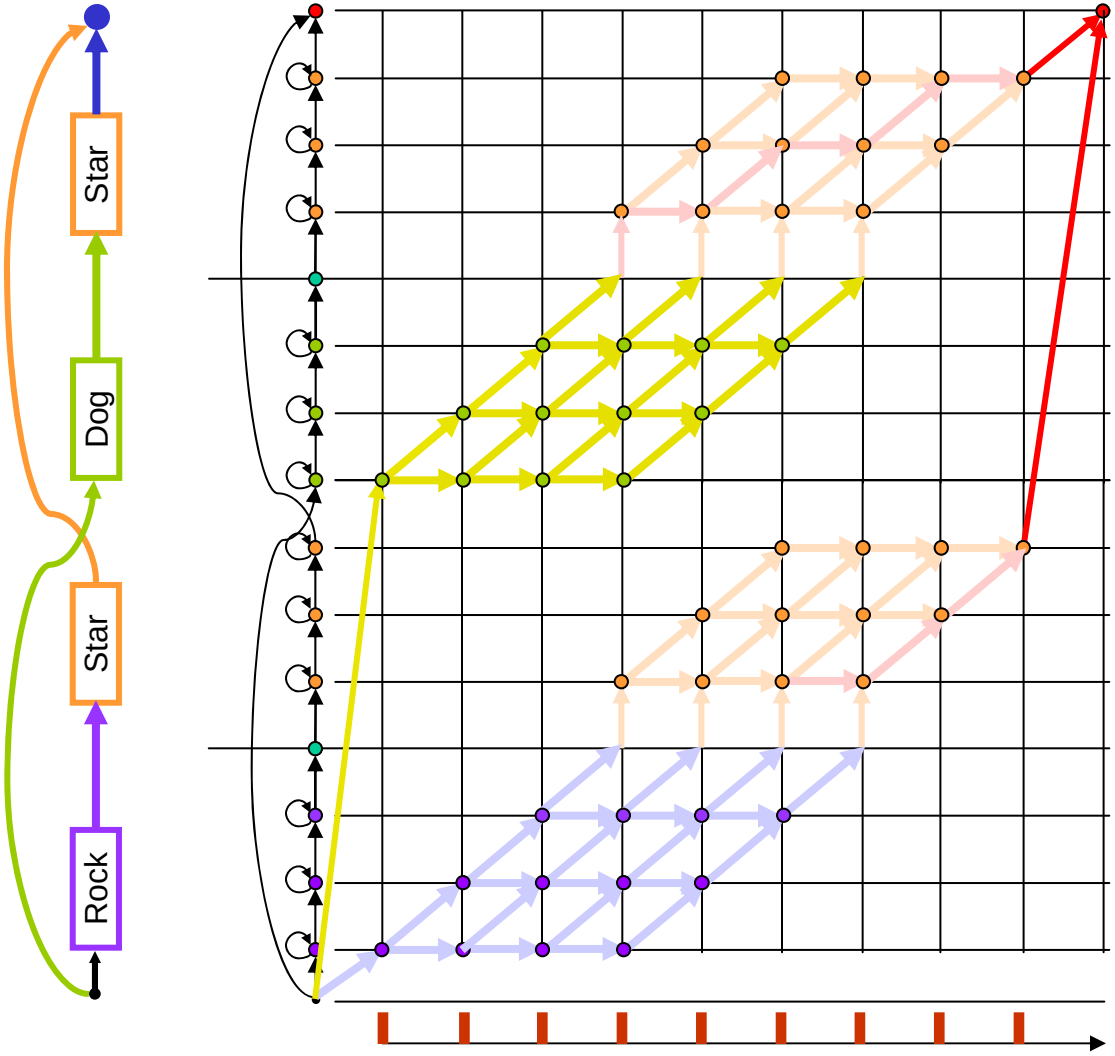
Then we'd compare the best paths through *Dog Star* and *Rock Star*

max(dogstar) =
max (dogstar1, dogstar2, dogstar3, dogstar4)

max(rockstar) =
max (rockstar1, rockstar2, rockstar3, rockstar4)

Viterbi =
max(max(dogstar), max(rockstar))

Decoding to classify between word sequences



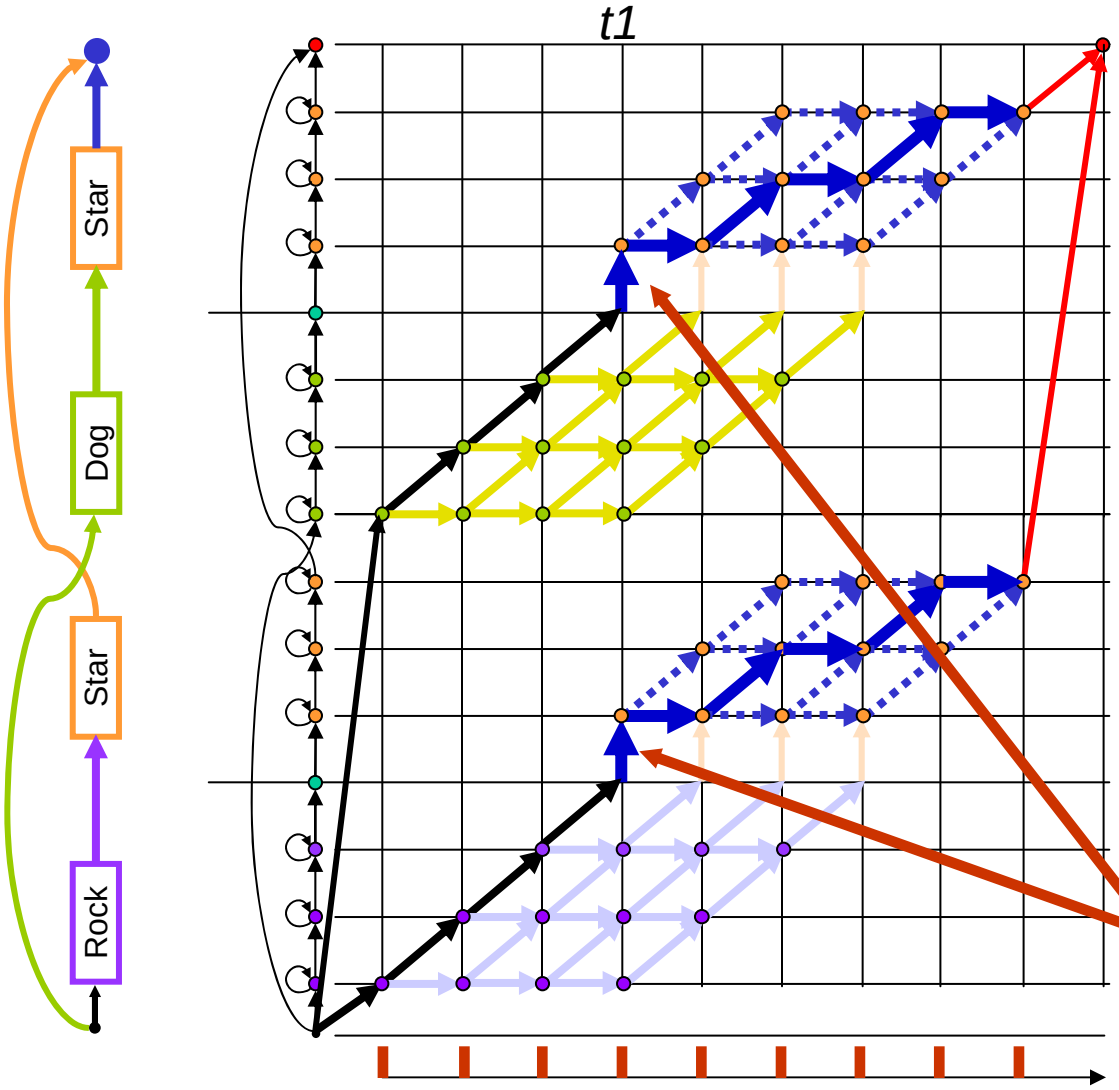
argmax is commutative:

max(max(dogstar), max(rockstar))

=

max (
  max(dogstar1, rockstar1),
  max (dogstar2, rockstar2),
   max (dogstar3,rockstar3),
   max(dogstar4,rockstar4)
)

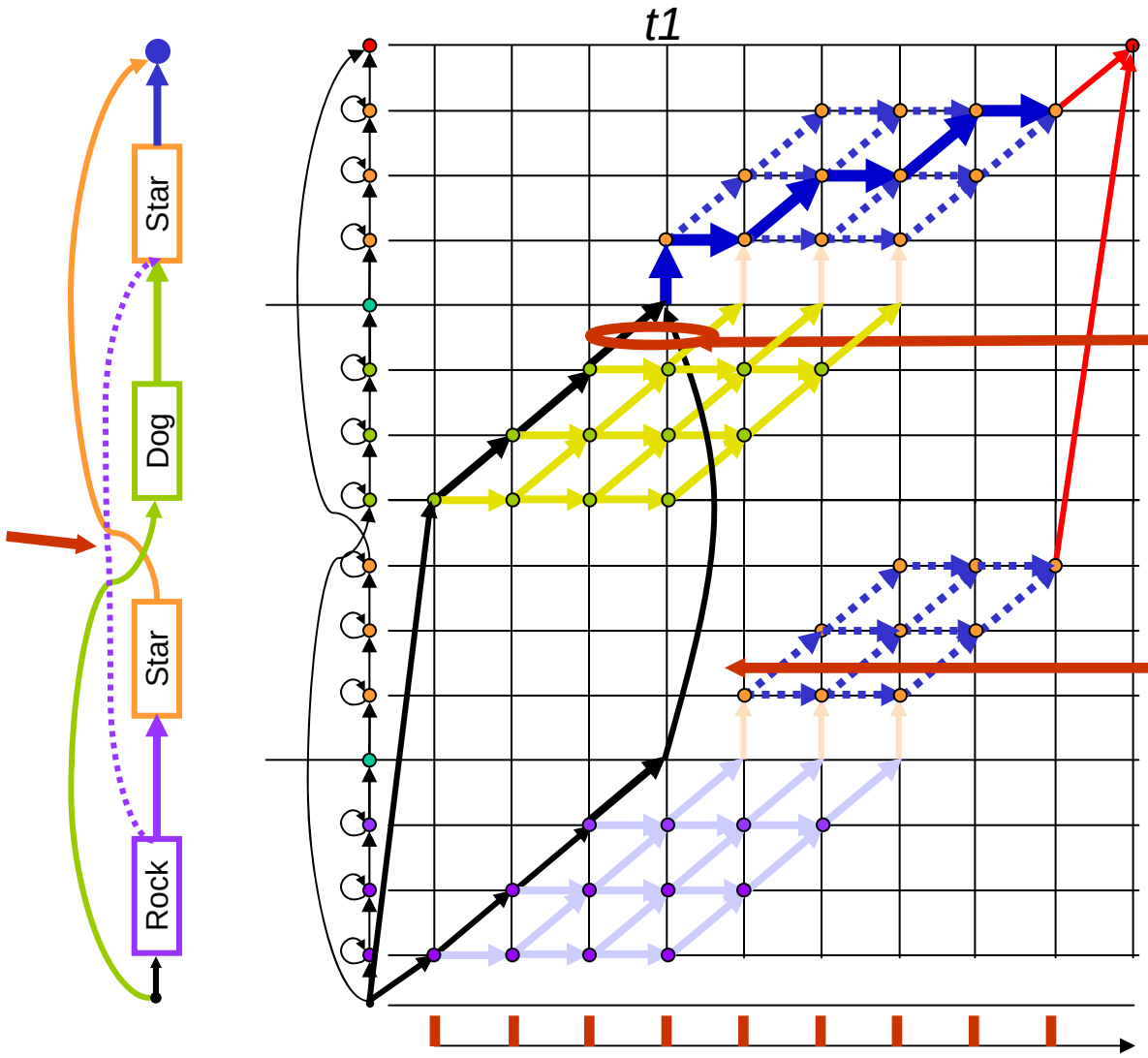
Decoding to classify between word sequences



For a given entry point the best path through STAR is the same for both trellises

We can choose between Dog and Rock right here because the futures of these paths are identical

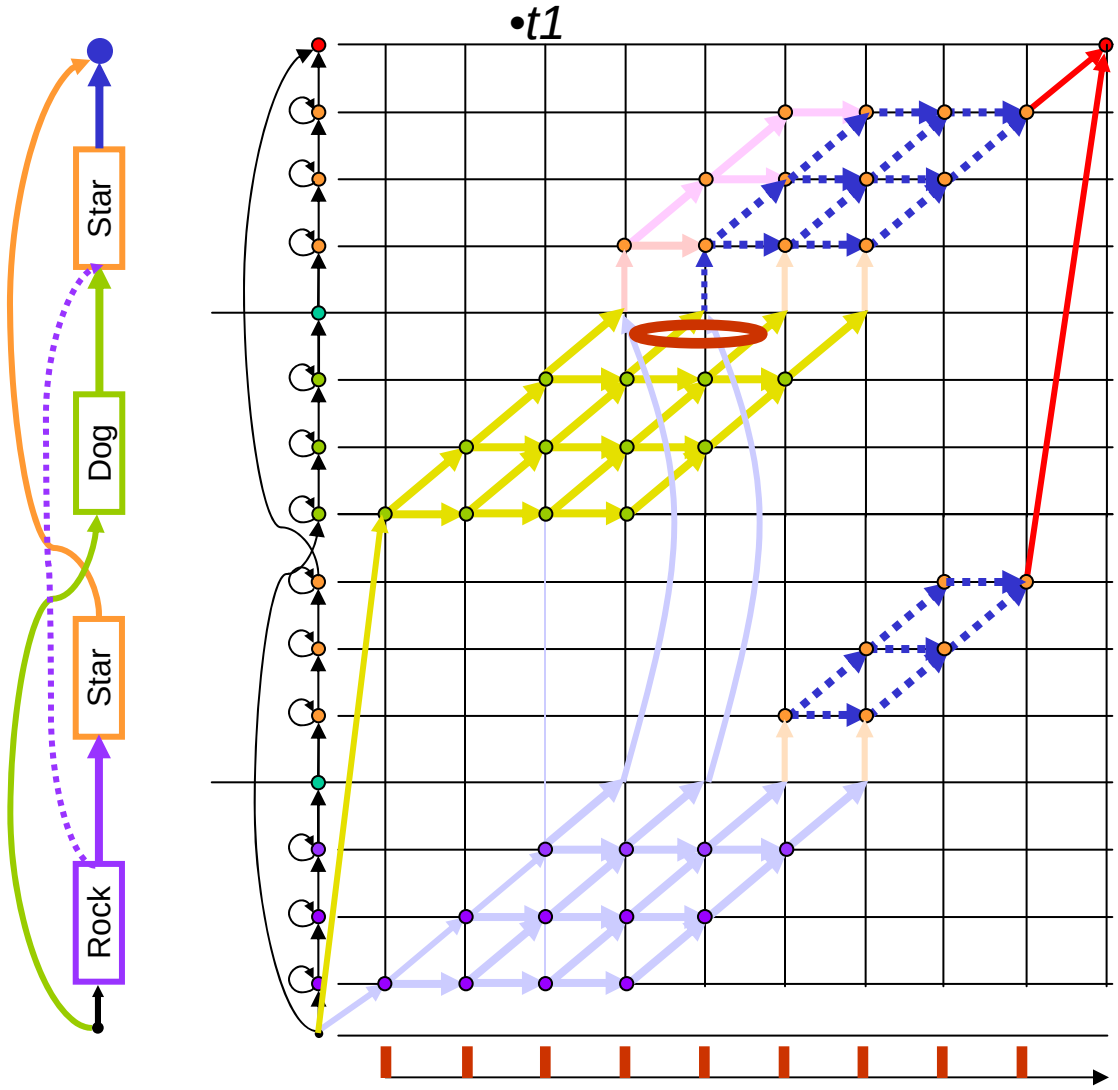
Decoding to classify between word sequences



We select the higher scoring of the two incoming edges here

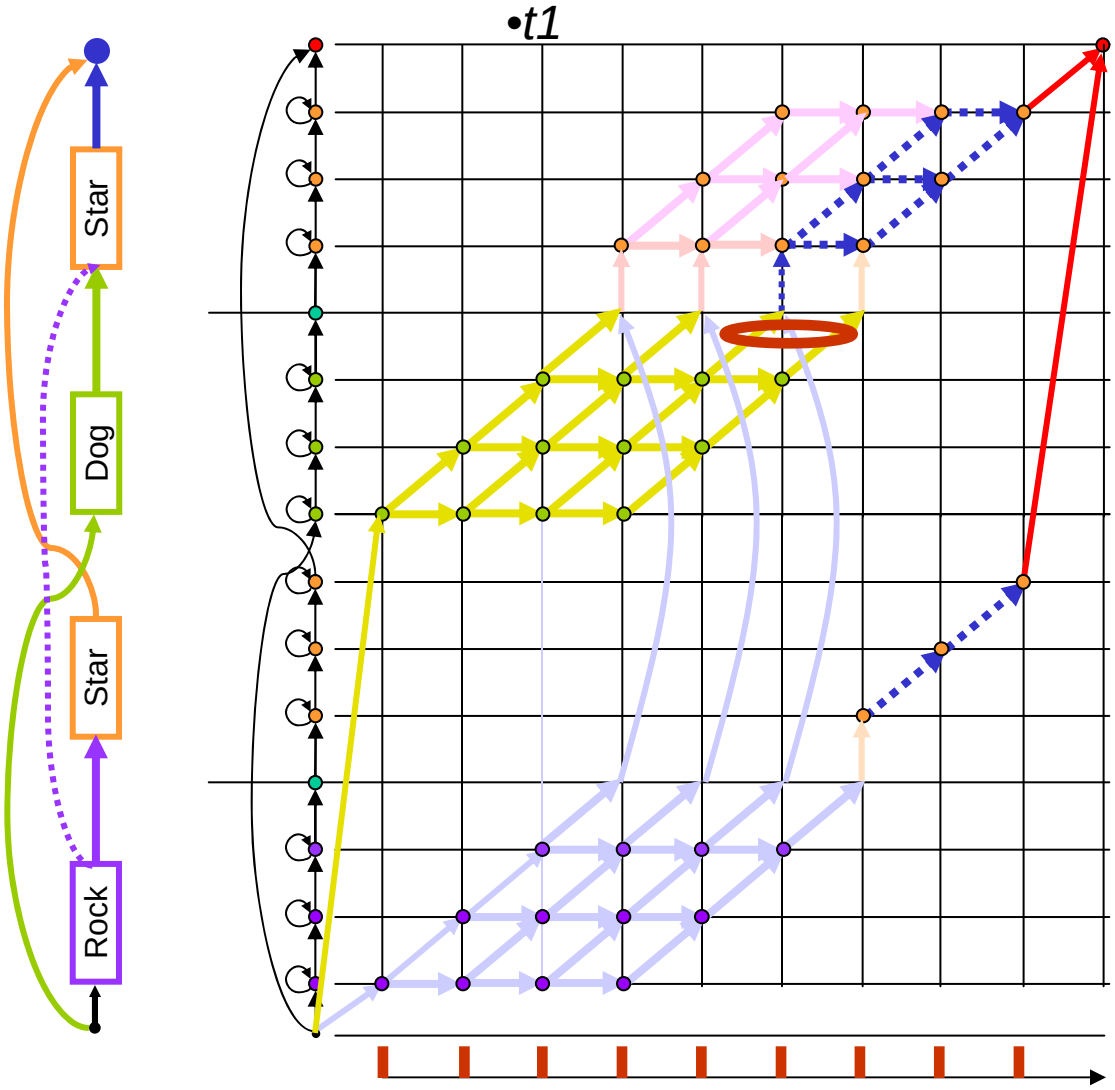
This portion of the trellis is now deleted

Decoding to classify between word sequences



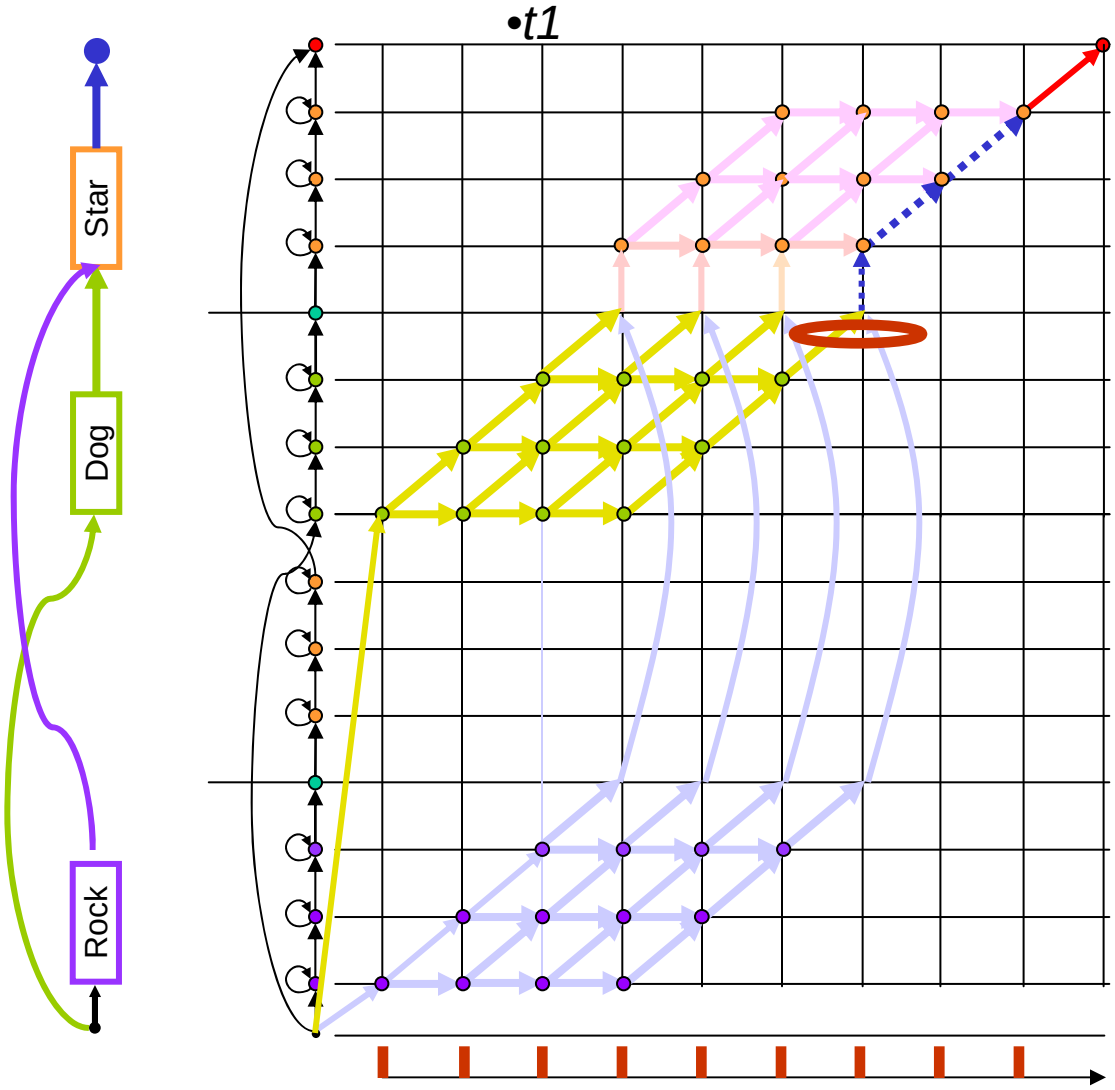
Similar logic can be applied at other entry points to *Star*

Decoding to classify between word sequences



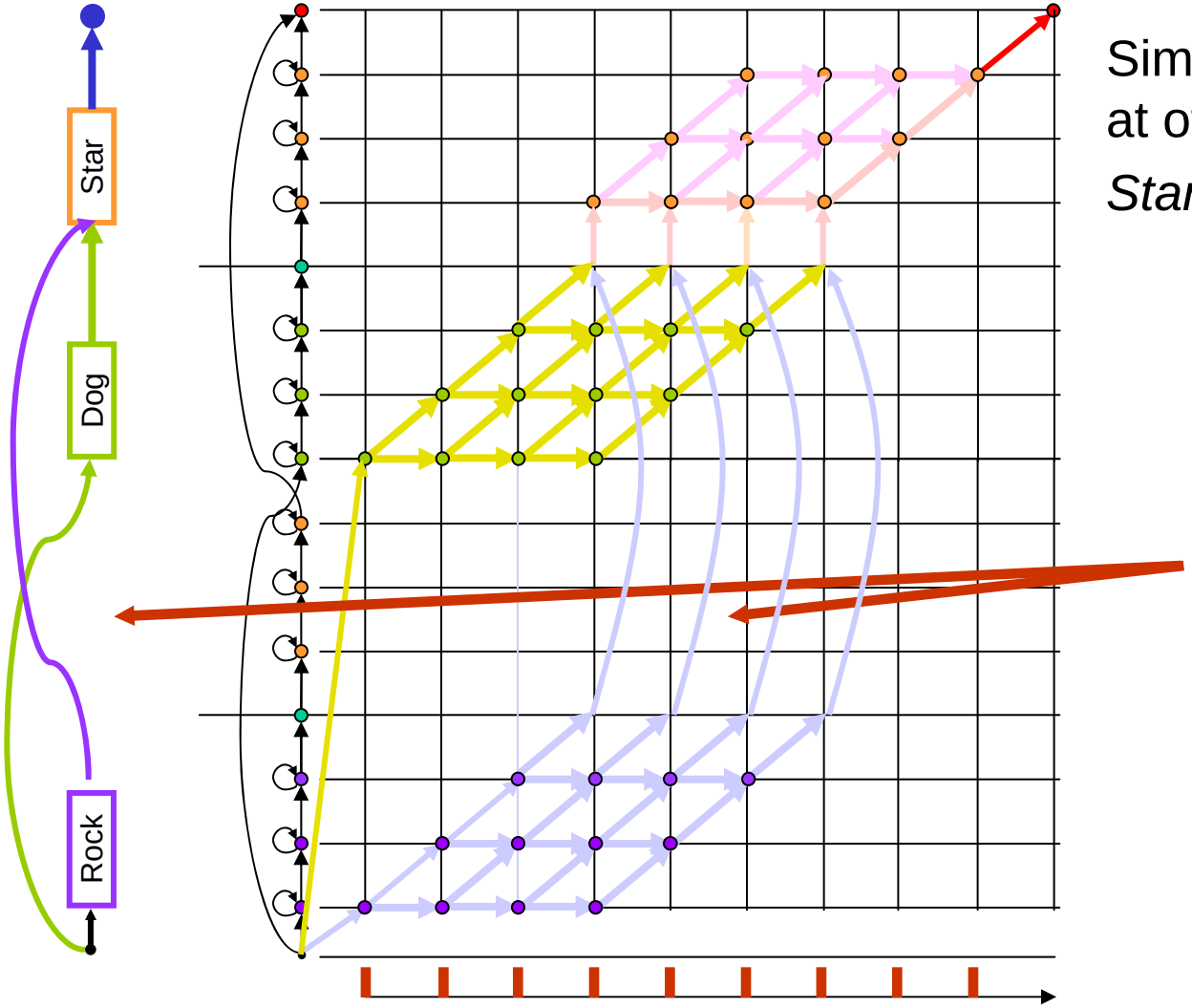
Similar logic can be applied at other entry points to *Star*

Decoding to classify between word sequences



Similar logic can be applied at other entry points to *Star*

Decoding to classify between word sequences

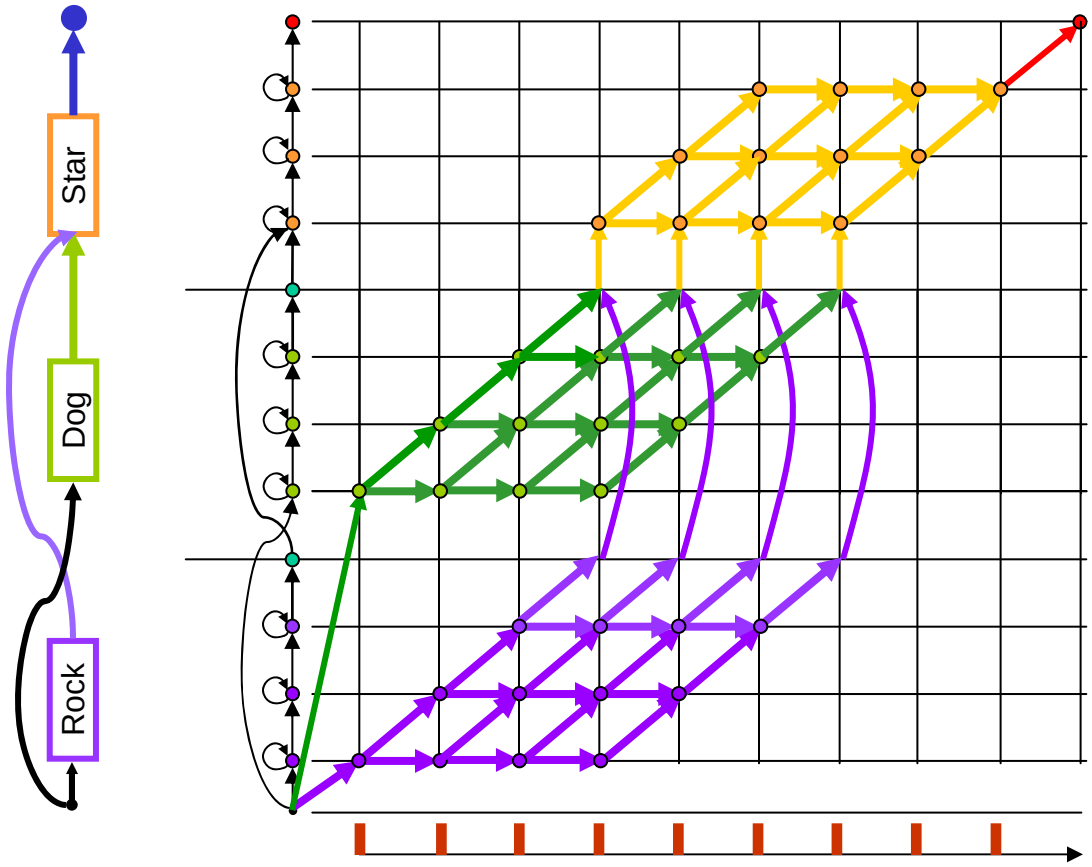


Similar logic can be applied at other entry points to *Star*

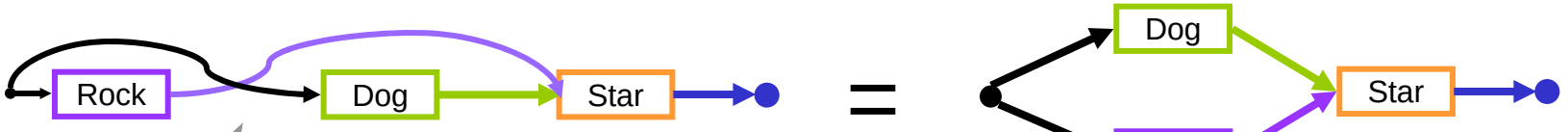
This copy of the trellis for *STAR* is completely removed

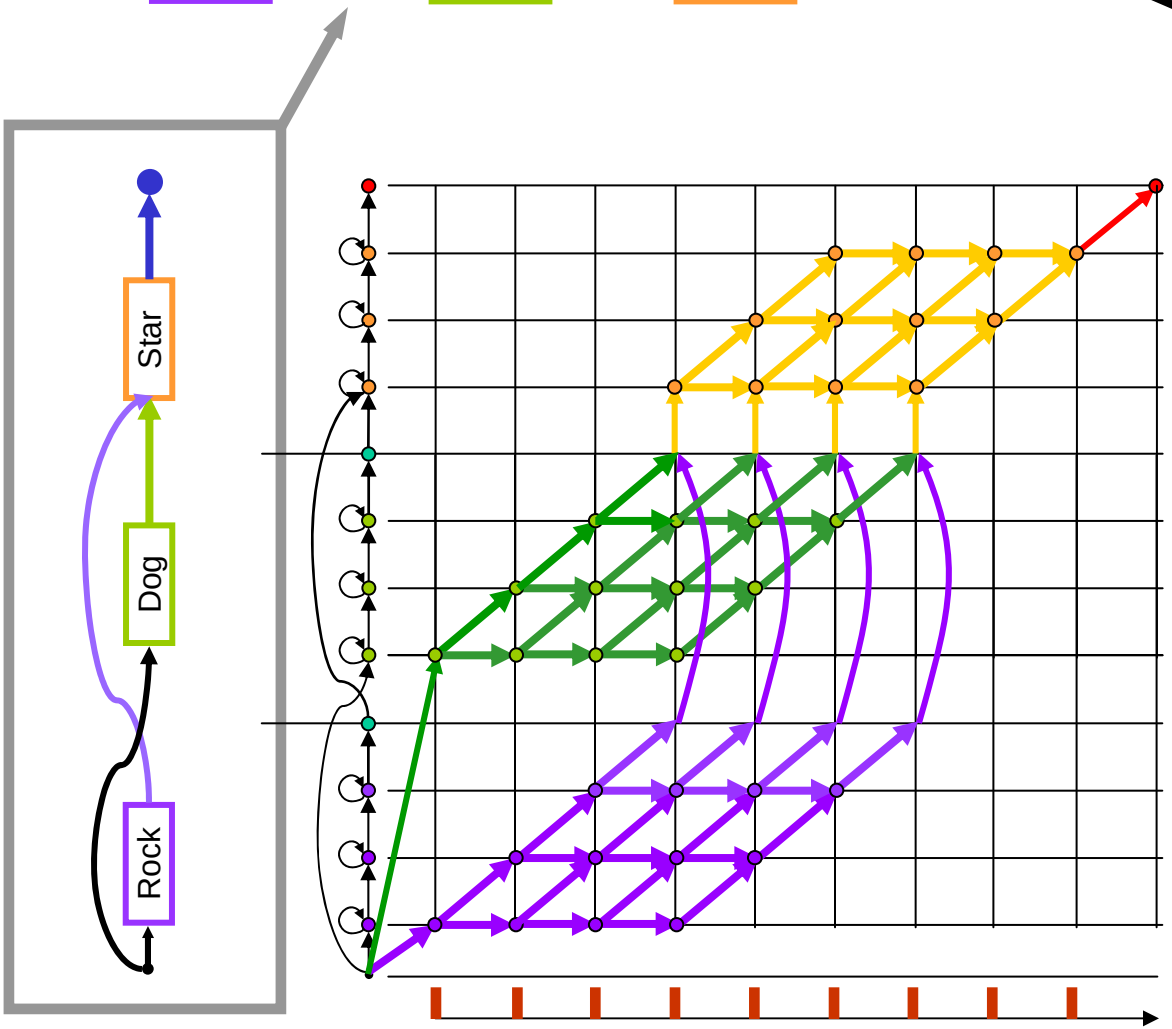
Decoding to classify between word sequences

- The two instances of *Star* can be collapsed into one to form a smaller trellis



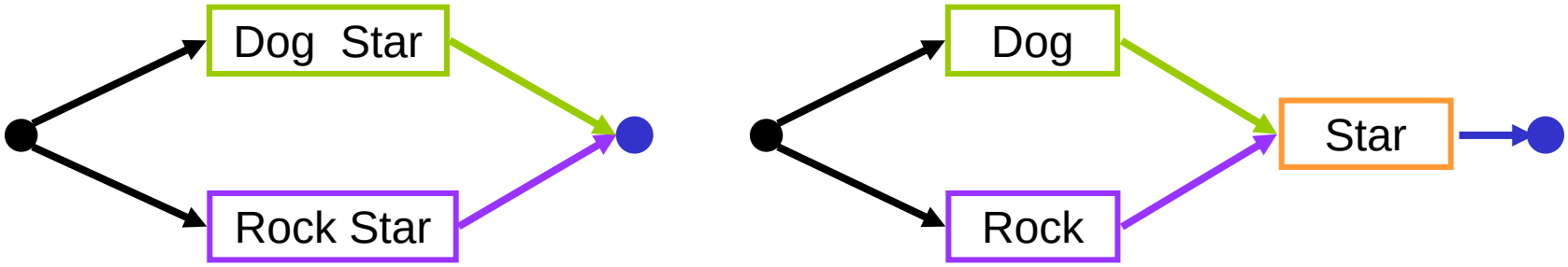
Language-HMMs for fixed length word sequences





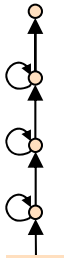
We will represent the vertical axis of the trellis in this simplified manner

The Real "Classes"

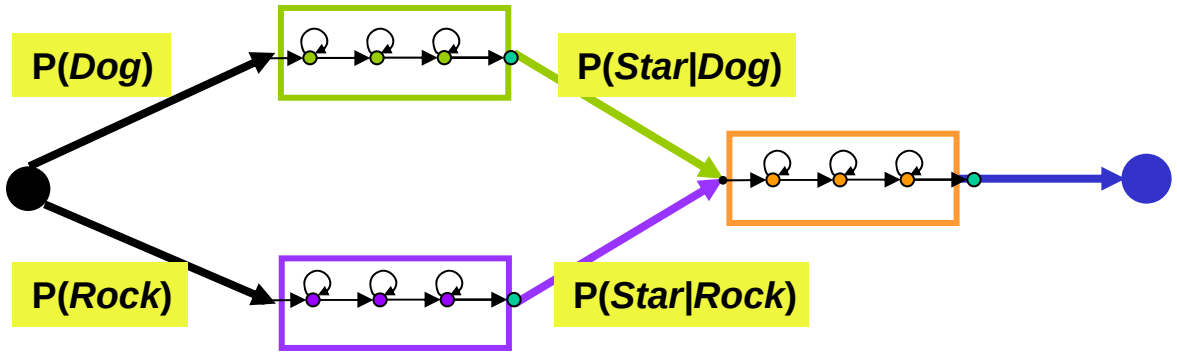


- The actual recognition is DOG STAR vs. ROCK STAR
  - i.e. the two items that form our "classes" are entire phrases
- The reduced graph to the right is merely an engineering reduction obtained by utilizing commonalities in the two phrases (STAR)
- This distinction affects the design of the recognition system

Language-HMMs for fixed length word sequences

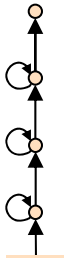


Each word is an HMM



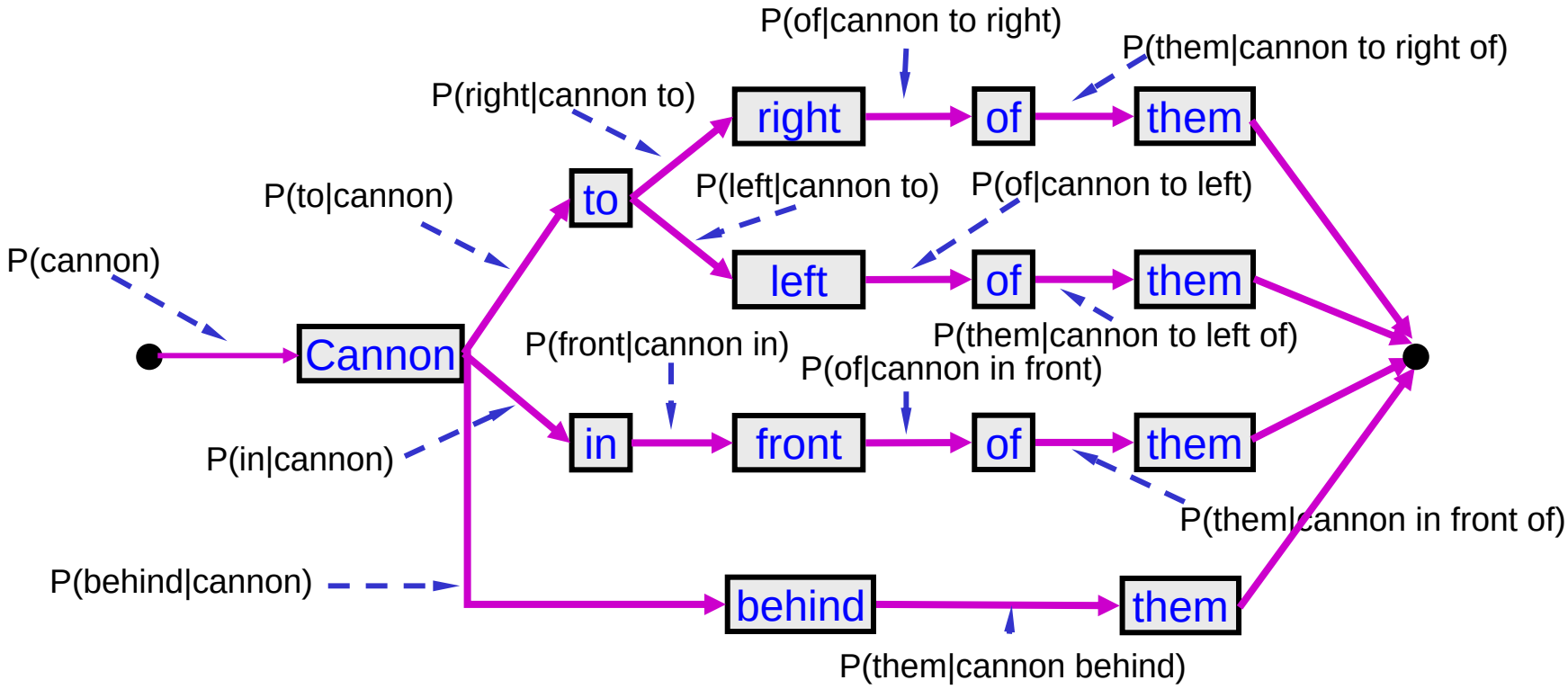
- The word graph represents all allowed word sequences in our example
  - The set of all allowed word sequences represents the allowed "language"
- At a more detailed level, the figure represents an HMM composed of the HMMs for all words in the word graph
  - This is the "Language HMM" – the HMM for the entire allowed language
- The language HMM represents the vertical axis of the trellis
  - It is the trellis, and NOT the language HMM, that is searched for the best path

Language-HMMs for fixed length word sequences



Each word is an HMM

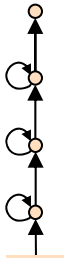
- Recognizing one of four lines from "charge of the light brigade"
  - Cannon to right of them
  - Cannon to left of them
  - Cannon in front of them
  - Cannon behind them



Where does the graph come from

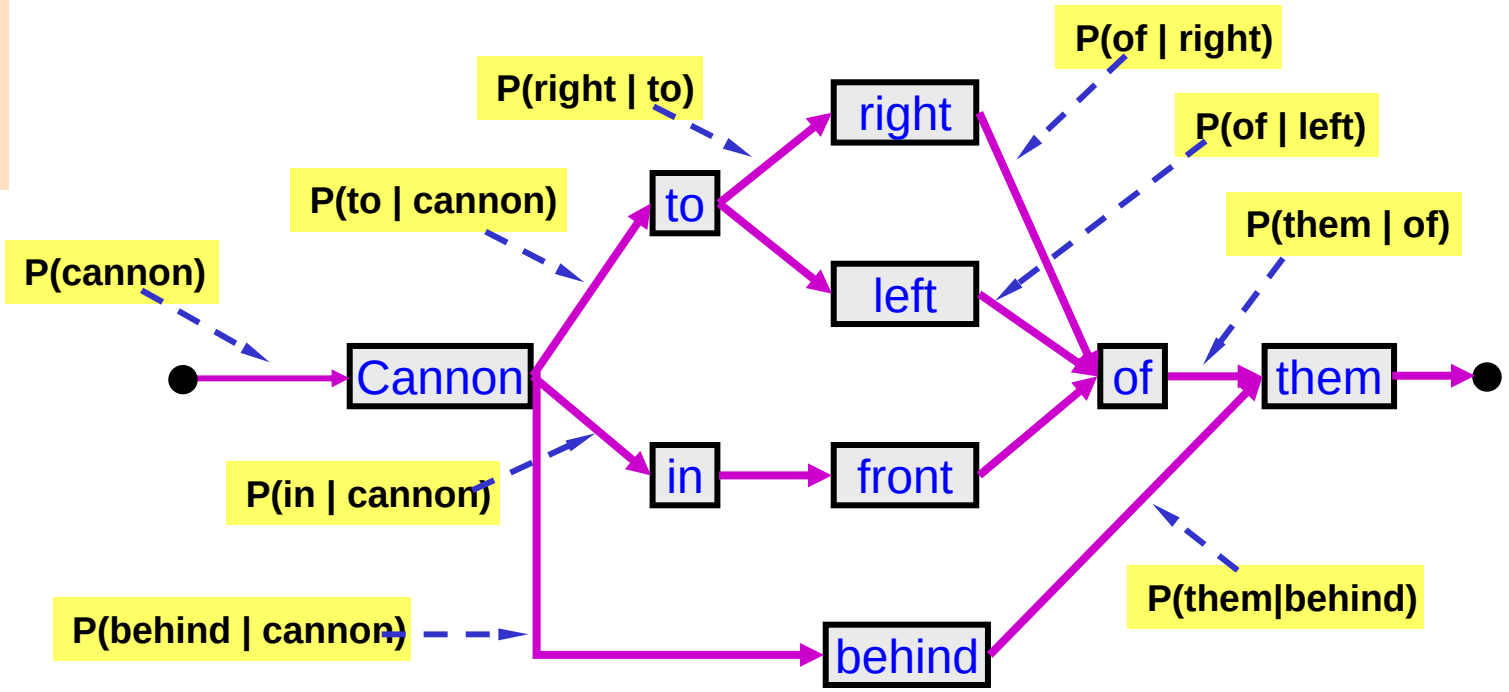
- The graph must be specified to the recognizer
  - What we are actually doing is to specify the complete set of “allowed” sentences in graph form
- May be specified as an FSG or a Context-Free Grammar
  - CFGs and FSG do not have probabilities associated with them
  - We could factor in prior biases through *probabilistic* FSG/CFGs
  - In probabilistic variants of FSGs and CFGs we associate probabilities with options
    - E.g. in the last graph

Simplification of the language HMM through lower context language models

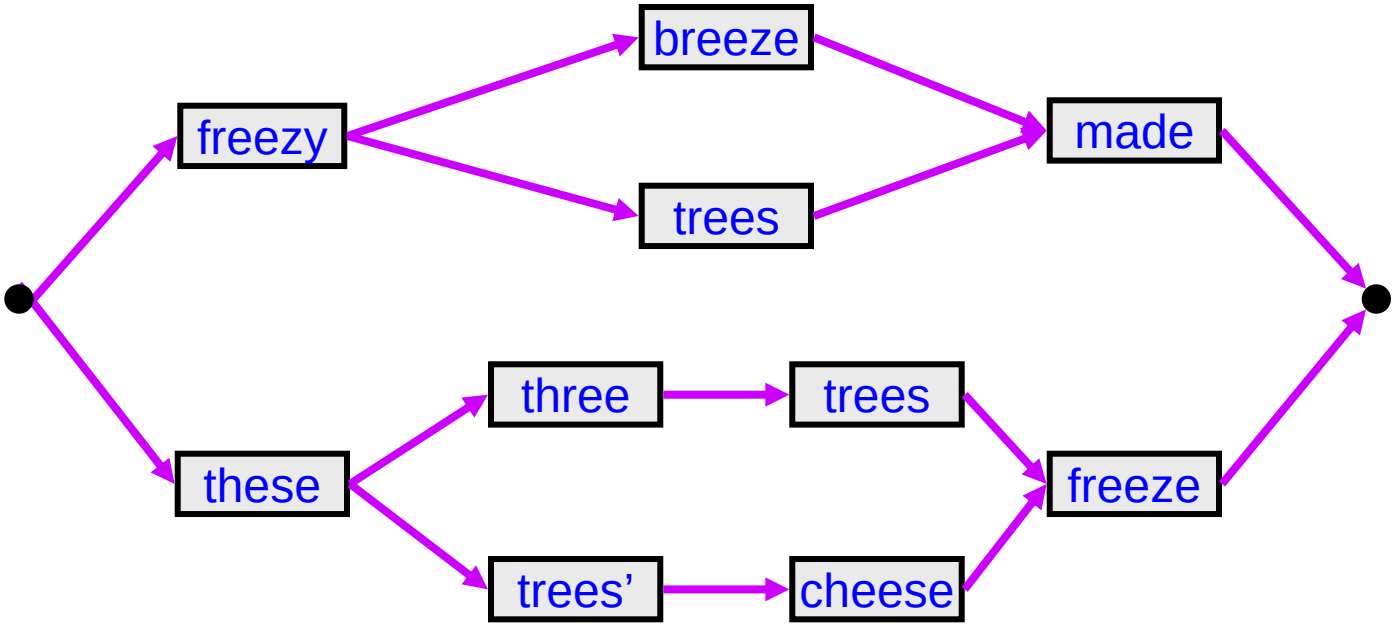


Each word is an HMM

- Recognizing one of four lines from “charge of the light brigade”
- If the probability of a word only depends on the preceding word, the graph can be collapsed:
  - **e.g. P(them | cannon to right of) = P(them | cannon to left of) = P(cannon | of)**

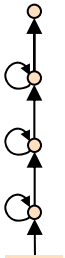


Language HMMs for fixed-length word sequences: based on a grammar for Dr. Seuss

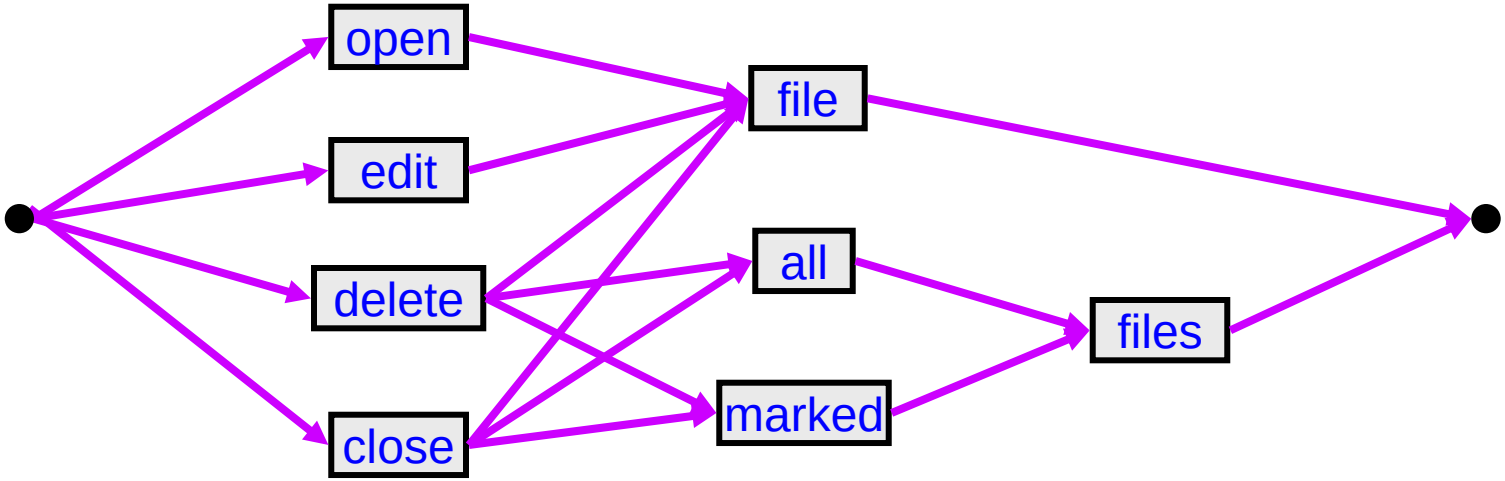


No probabilities specified – a person may utter any of these phrases at any time

Language HMMs for fixed-length word sequences: command and control grammar



Each word is an HMM



No probabilities specified – a person may utter any of these phrases at any time

Language HMMs for arbitrarily long word sequences

- Previous examples chose between a finite set of known word sequences
- Word sequences can be of arbitrary length
  - E.g. set of all word sequences that consist of an arbitrary number of repetitions of the word bang

    bang

    bang bang

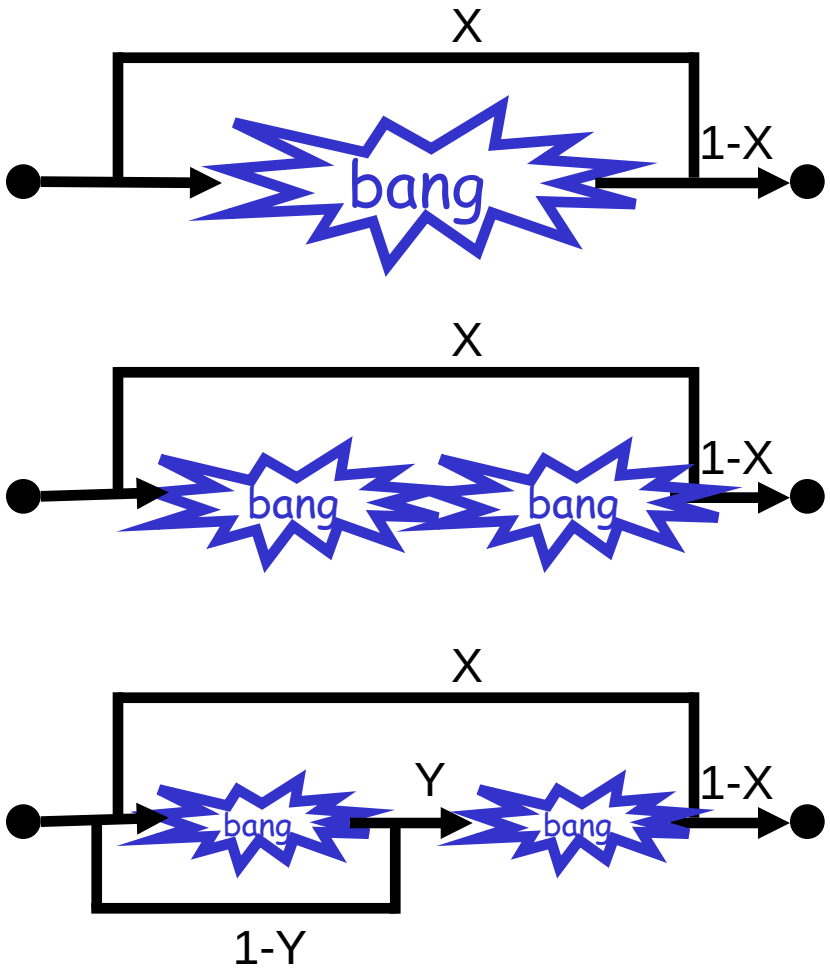
    bang bang bang

    bang bang bang bang

    ……
  - Forming explicit word-sequence graphs of the type we've seen so far is not possible
    - The number of possible sequences (with non-zero *a-priori* probability) is potentially infinite
    - Even if the longest sequence length is restricted, the graph will still be large

Language HMMs for arbitrarily long word sequences

- Arbitrary word sequences can be modeled with loops under some assumptions. E.g.:
- A "bang" can be followed by another "bang" with probability P("bang").
  - P("bang") = X;
    P(Termination) = 1-X;
- Bangs can occur only in pairs with probability X
- A more complex graph allows more complicated patterns
- You can extend this logic to other vocabularies where the speaker says other words in addition to "bang"
  - e.g. "bang bang you're dead"



Language HMMs for arbitrarily long word sequences

- Constrained set of word sequences with constrained vocabulary are realistic
  - Typically in command-and-control situations
    - Example: operating TV remote
  - Simple dialog systems
    - When the set of permitted responses to a query is restricted
- Unconstrained word sequences : NATURAL LANGUAGE
  - State-of-art large vocabulary decoders
  - Later in the program..

QUESTIONS?

- Next up: Pruning and the Backpointer table
- Any questions on topics so far?

Pruning

- The search for the best path can become very expensive
  - As model size or utterance length increase, the trellis size increases
  - Number of paths to evaluate increases unmanageable
- Need to reduce computation somehow
  - Eliminating parts of the trellis from consideration altogether
    - This approach is called *search pruning*, or just *pruning*
- Basic consideration in pruning: *As long as the best cost path is not eliminated by pruning, we obtain the same result*

Pruning

- Pruning is a *heuristic*: typically, there is a *threshold* on some measured quantity, and anything above or below is eliminated

- It is all about choosing the right measure, and the right threshold

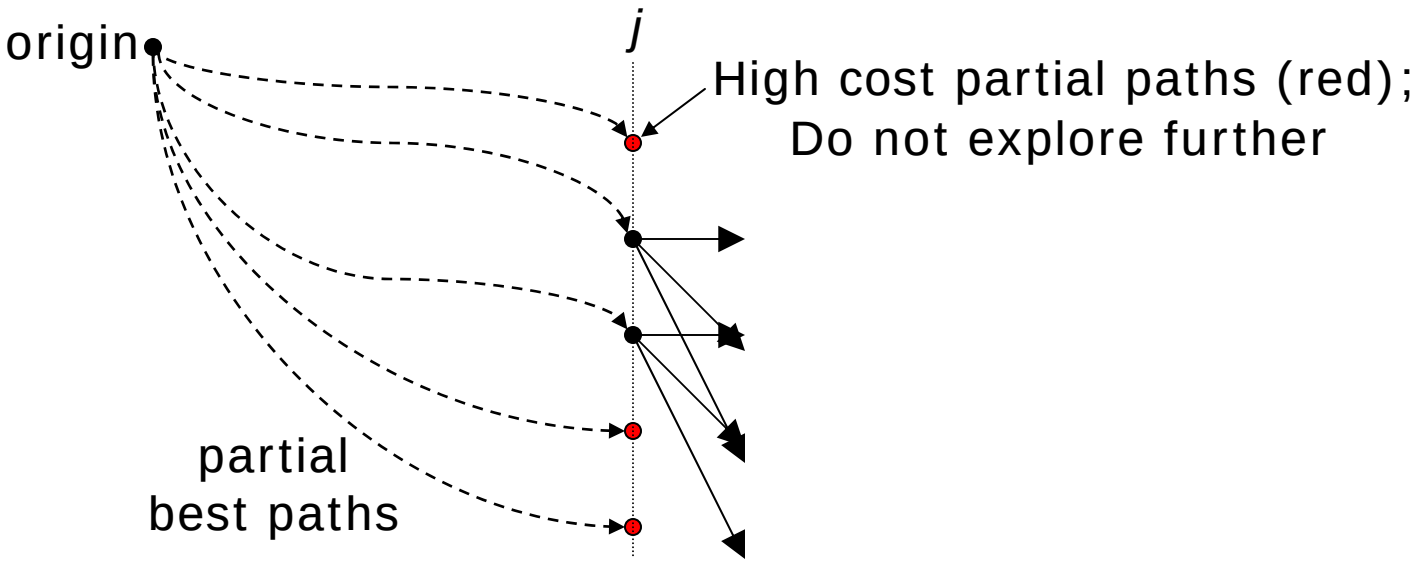
- Let us see two different pruning methods:
  - Fixed-width pruning
  - Relative-score pruning

Fixed-width Pruning

- At each time find the highest scoring node and retain only the N closest nodes
- The rest of the nodes are not considered for further propagation

Pruning by Limiting Path Cost

- *Observation*: Partial paths with "very high" costs will rarely recover to win
- Hence, poor partial paths can be eliminated from the search:
  - For each frame *j*, after computing all the trellis nodes path costs, determine which nodes have too high costs
  - Eliminate them from further exploration
  - (*Assumption*: In any frame, the best partial path has low cost)

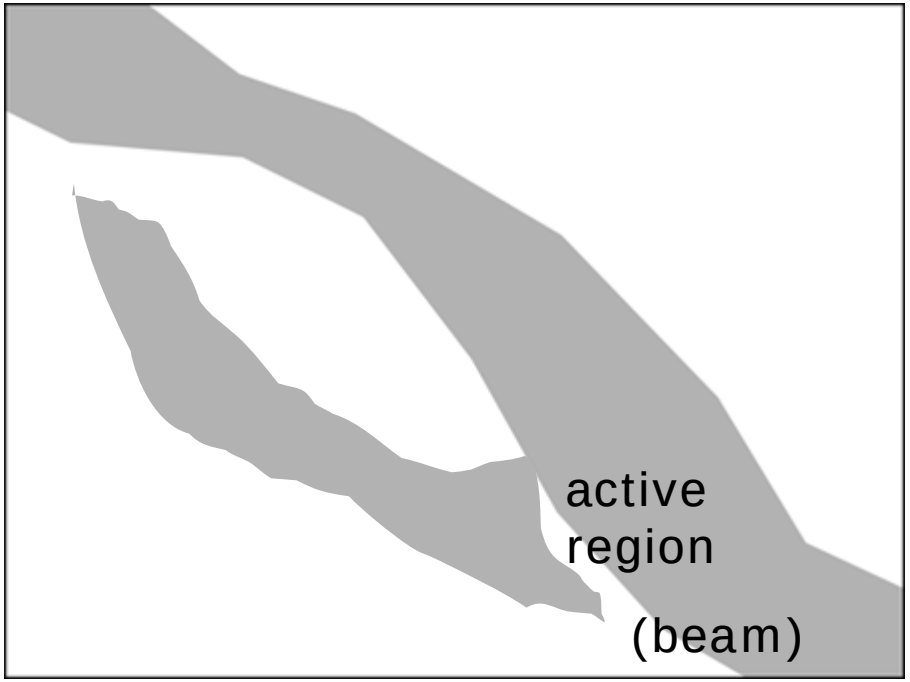


Pruning: Beam Search

- *Solution*: In each frame *j*, set the pruning threshold by a fixed amount *T relative to the best cost in that frame*
  - *I.e.* if the best partial path cost achieved in the frame is *X*, prune away all nodes with partial path cost > *X*+*T*
  - Note that *time synchronous* search is very efficient for implementing the above
- Advantages:
  - Unreliability of absolute path costs is eliminated
  - Monotonic growth of path costs with time is also irrelevant
- Search that uses such pruning is called *beam search*
  - This is the most widely used search optimization strategy
- The relative threshold *T* is usually called *beam width* or just *beam*

Beam Search Visualization

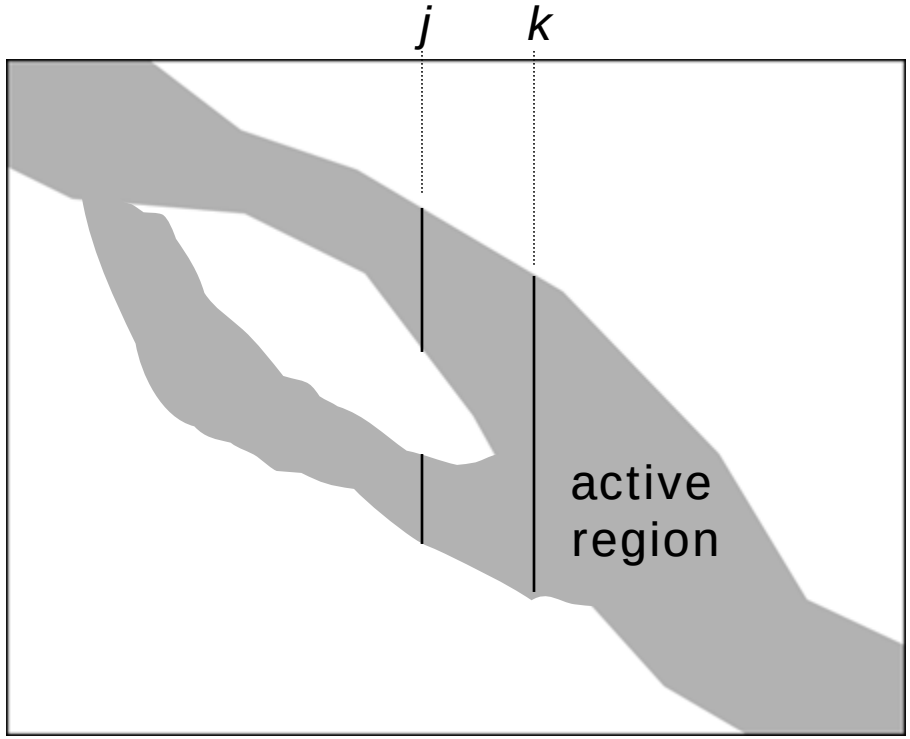
- The set of trellis nodes actually evaluated is the *active* set
- Here is a typical “map” of the *active region*, aka *beam* (confusingly)



- Presumably, the best path lies somewhere in the active region

Beam Search Efficiency

- Unlike the fixed width approach, the computation reduction with beam search is unpredictable
  - The set of *active nodes* at frames *j* and *k* is shown by the black lines
- However, since the active region can follow any *warping*, it is likely to be relatively more efficient than the fixed width approach

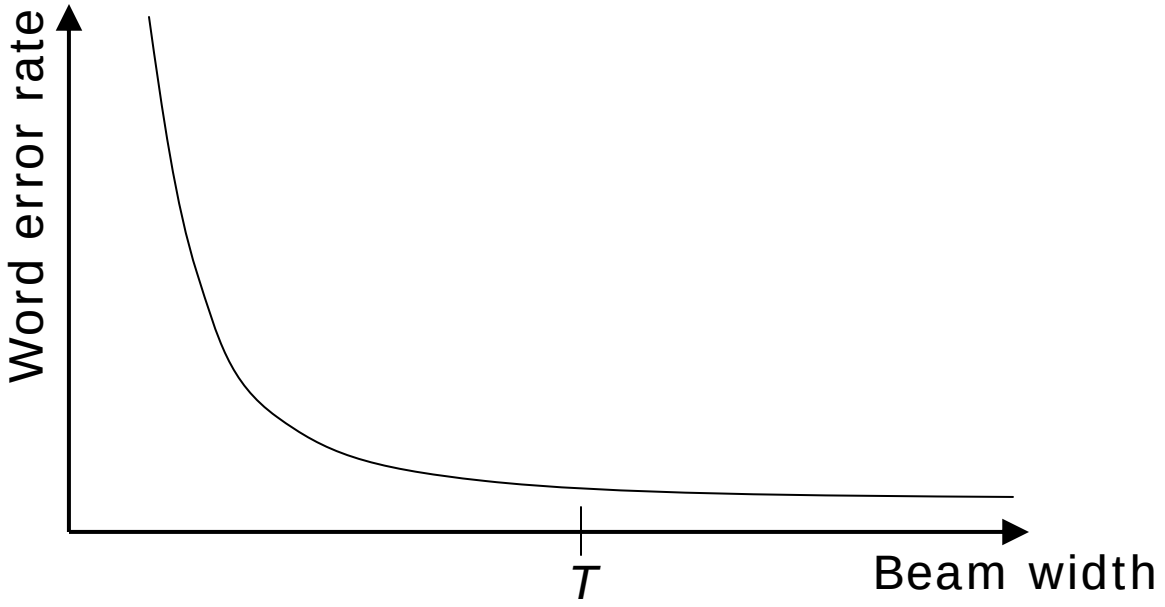


Determining the Optimal Beam Width

- Determining the optimal beam width to use is crucial
  - Using too *narrow* or *tight* a beam (too low *T*) can prune the best path and result in too high a match cost, and errors
  - Using too large a beam results in unnecessary computation in searching unlikely paths
  - One may also wish to set the beam to limit the computation (*e.g.* for real-time operation), regardless of recognition errors

- *Unfortunately, there is no mathematical solution to determining an optimal beam width*
- Common method: Try a wide range of beams on some test data until the desired operating point is found
  - Need to ensure that the test data are somehow representative of actual speech that will be encountered by the application
  - The operating point may be determined by some combination of recognition accuracy and computational efficiency

Determining the Optimal Beam Width

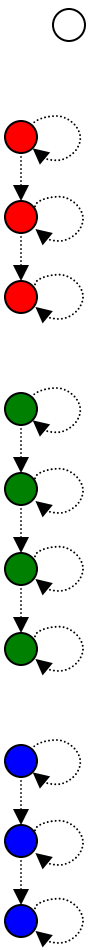


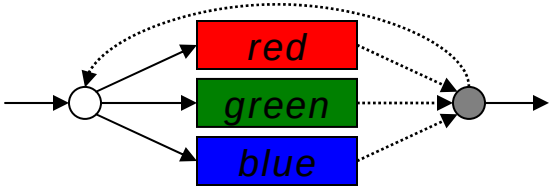
- Any value around the point marked *T* is a reasonable beam for minimizing *word error rate* (WER)
- A similar analysis may be performed based on average CPU usage (instead of WER)

Backpointer Table

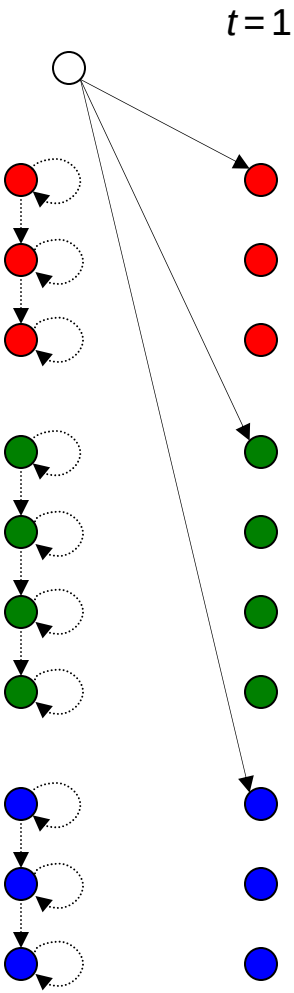
- The Viterbi decoding process stores backpointers from every state to its best predecessor at every time.
- For HMMs of large word graphs, this set of backpointers can quickly become very large
- Better to store “backpointers” efficiently, so that only necessary information is stored
- We use a “BackPointer Table”

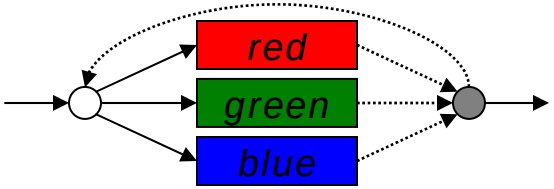
Trellis with Complete Set of Backpointers



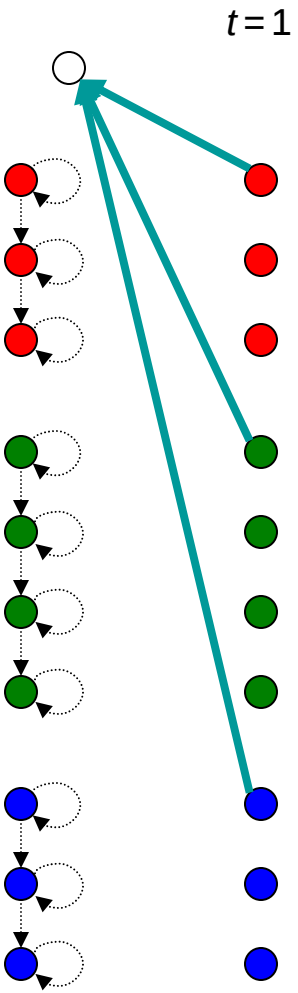


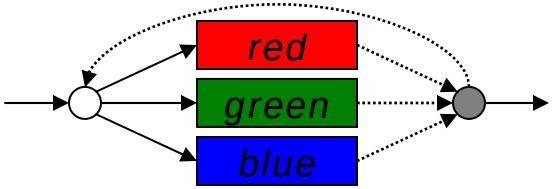
Trellis with Complete Set of Backpointers



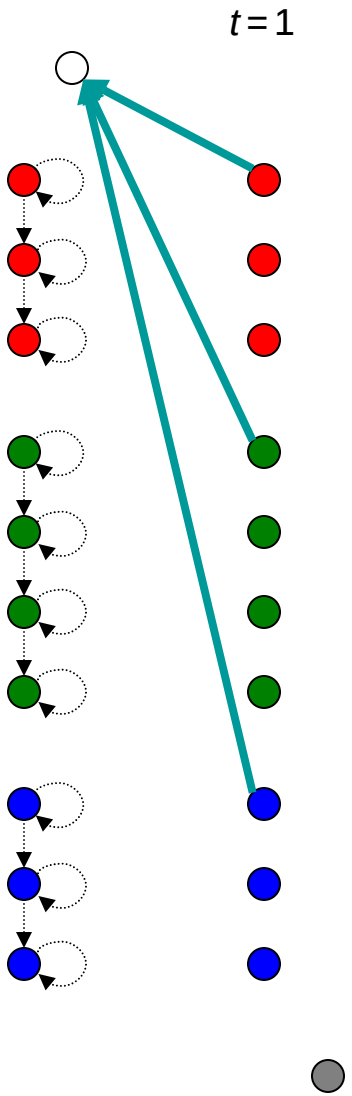


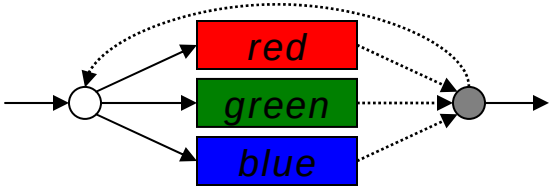
Trellis with Complete Set of Backpointers



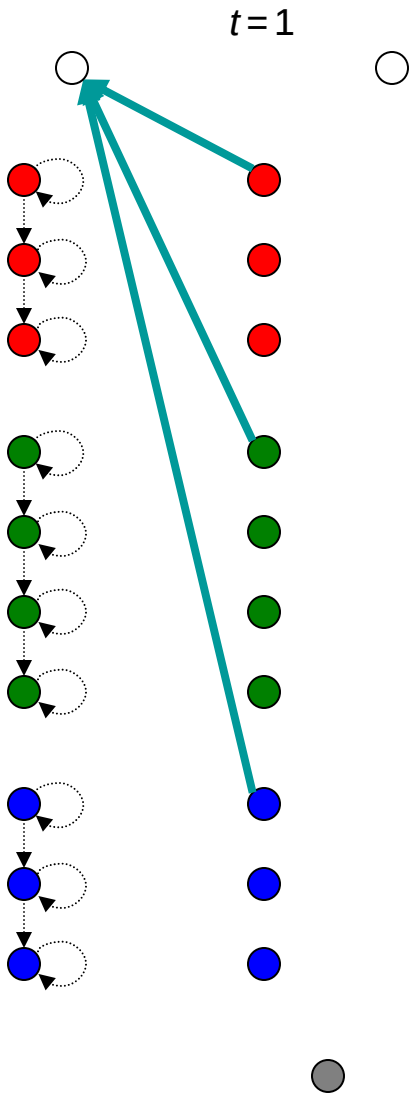


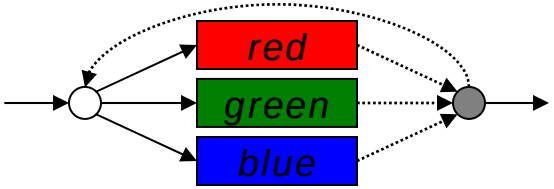
Trellis with Complete Set of Backpointers



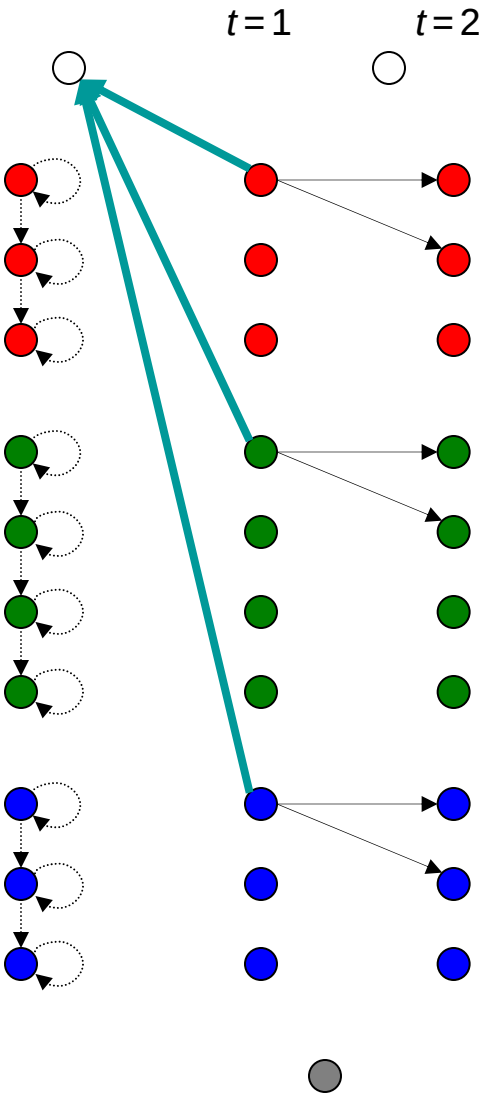


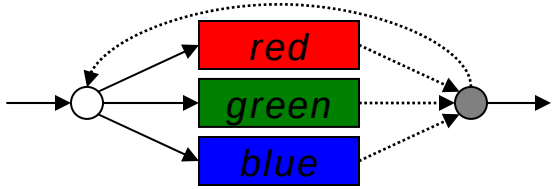
Trellis with Complete Set of Backpointers



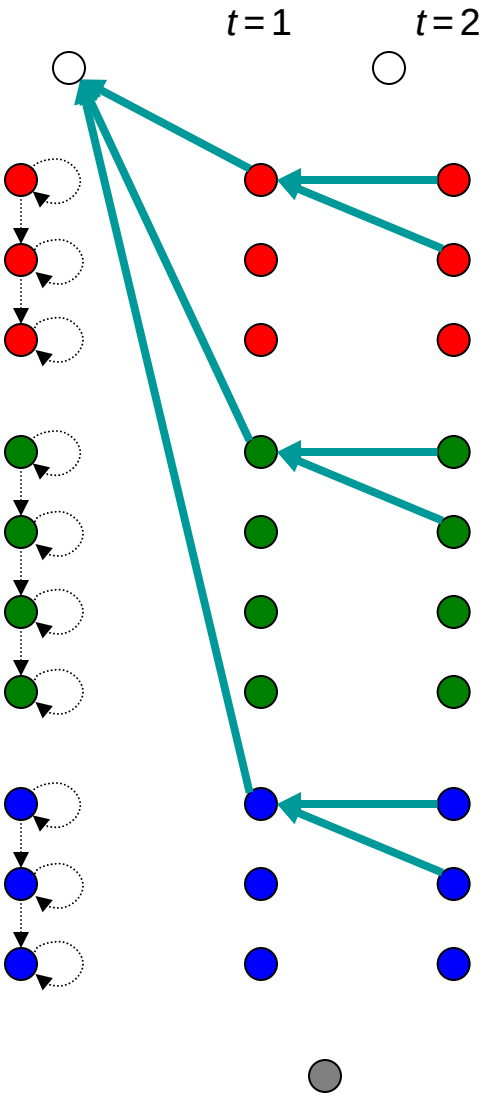


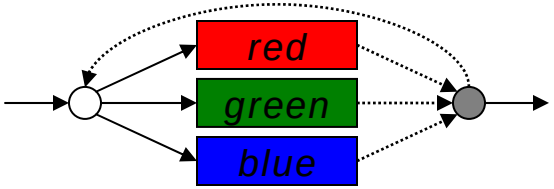
Trellis with Complete Set of Backpointers



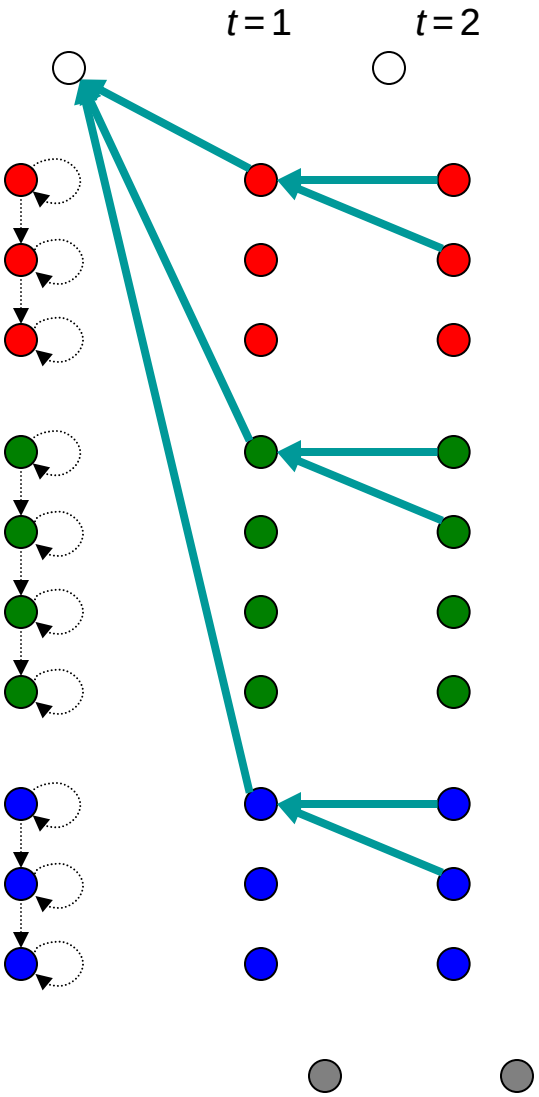


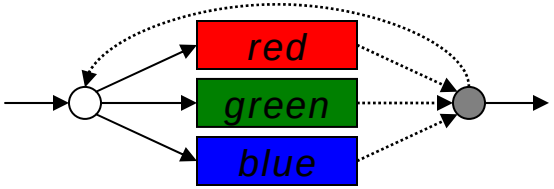
Trellis with Complete Set of Backpointers



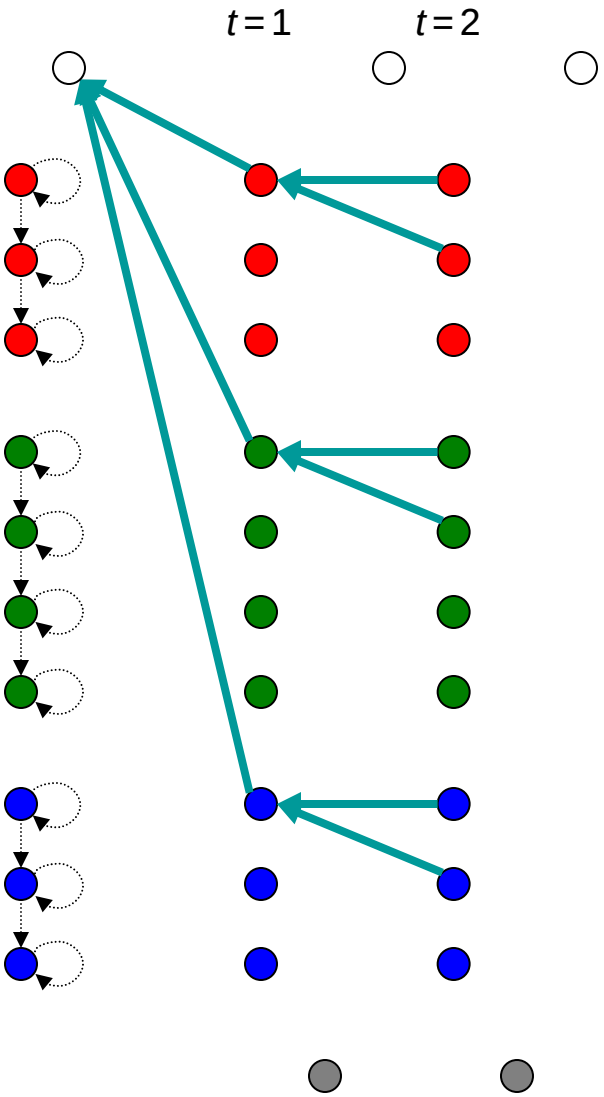


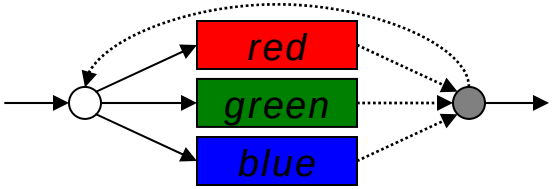
Trellis with Complete Set of Backpointers



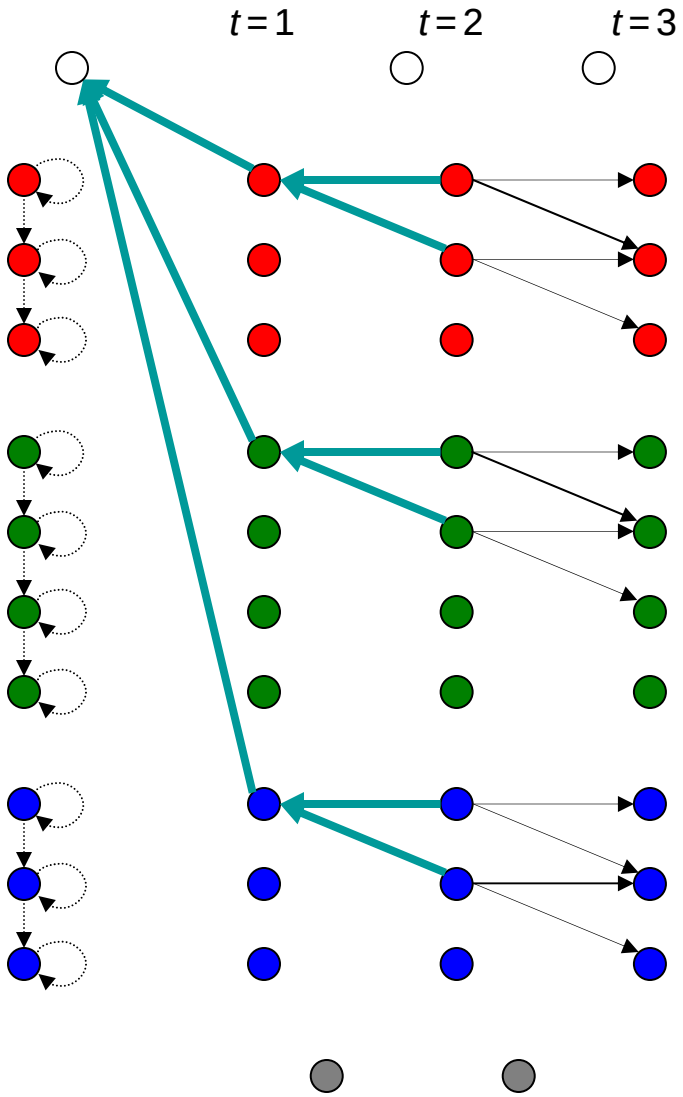


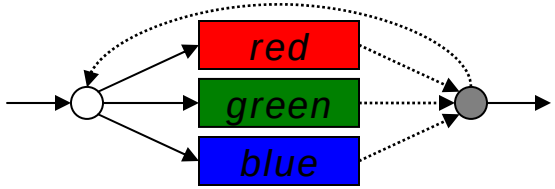
Trellis with Complete Set of Backpointers



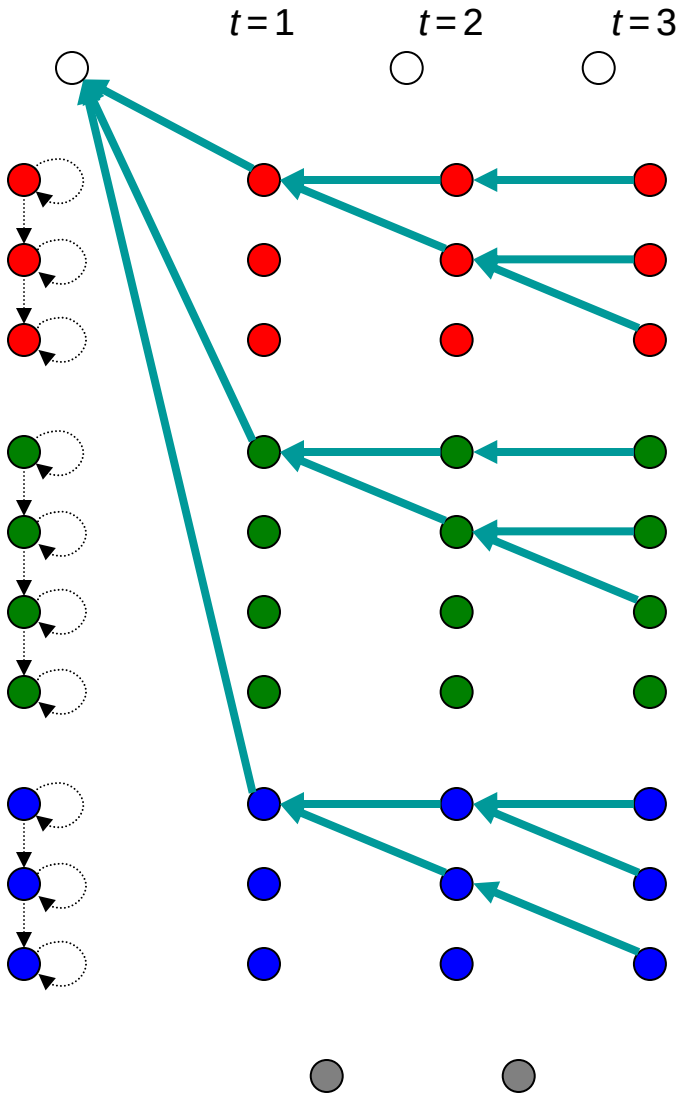


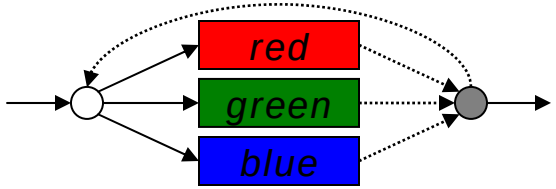
Trellis with Complete Set of Backpointers



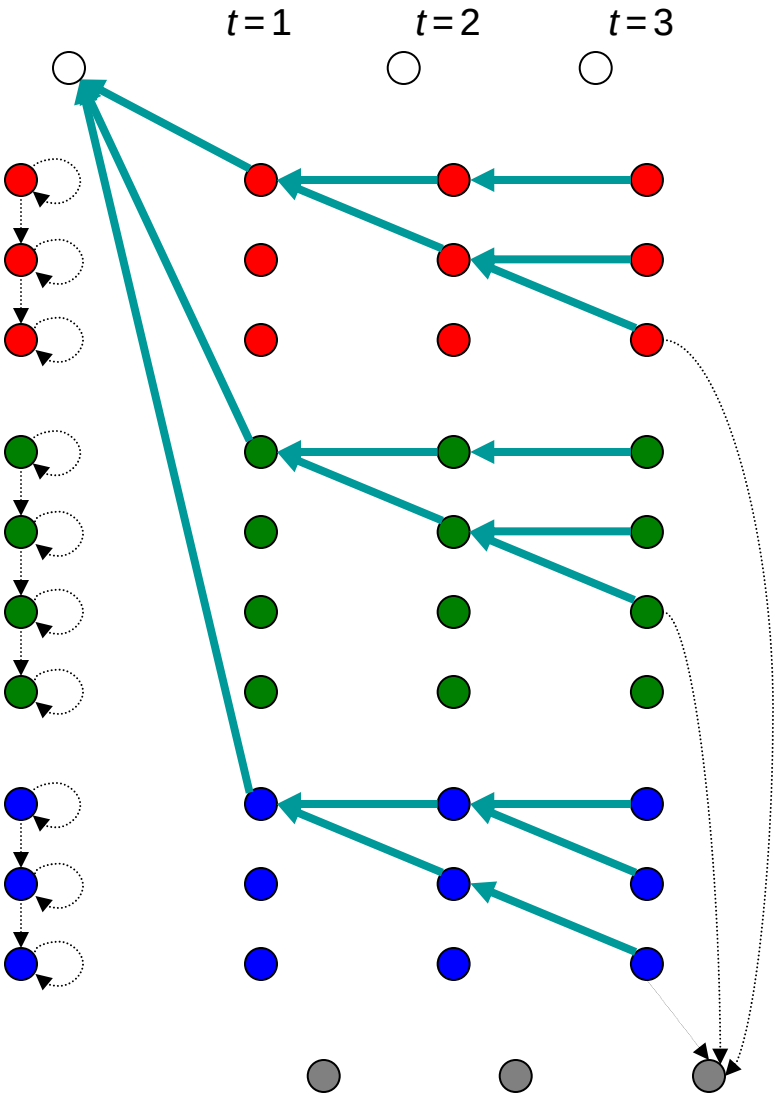


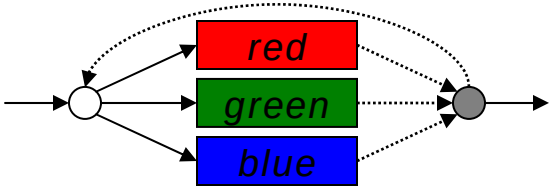
Trellis with Complete Set of Backpointers



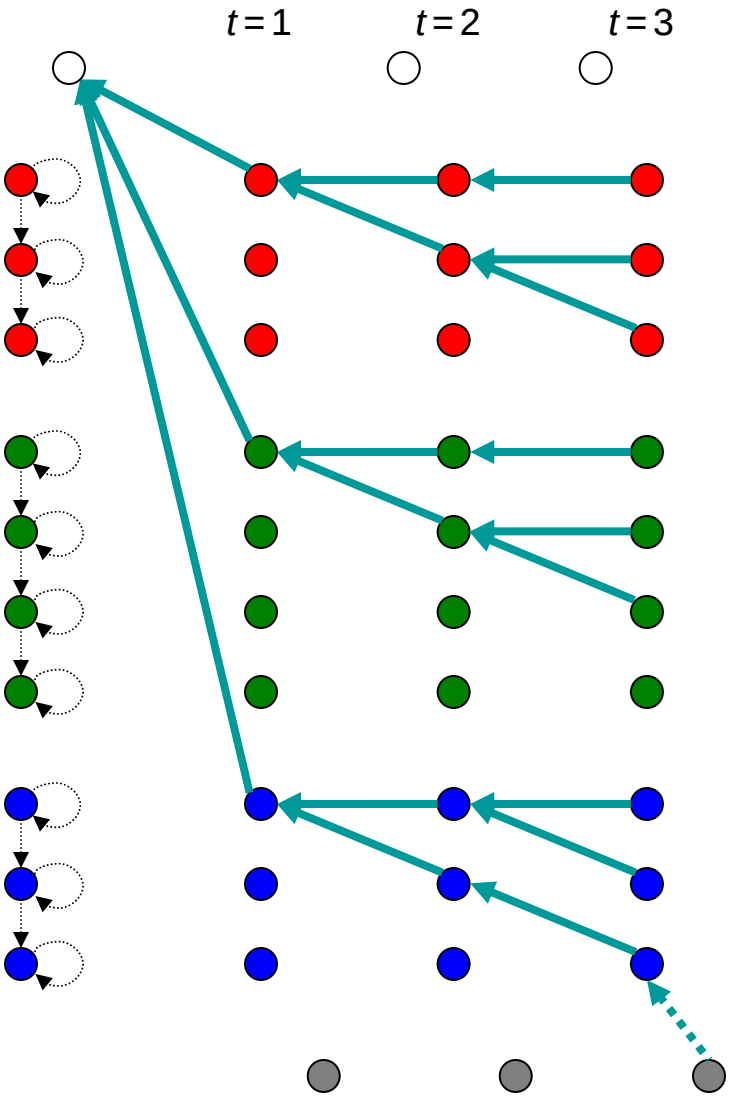


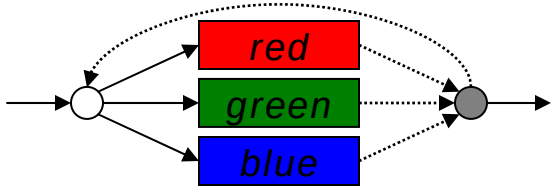
Trellis with Complete Set of Backpointers



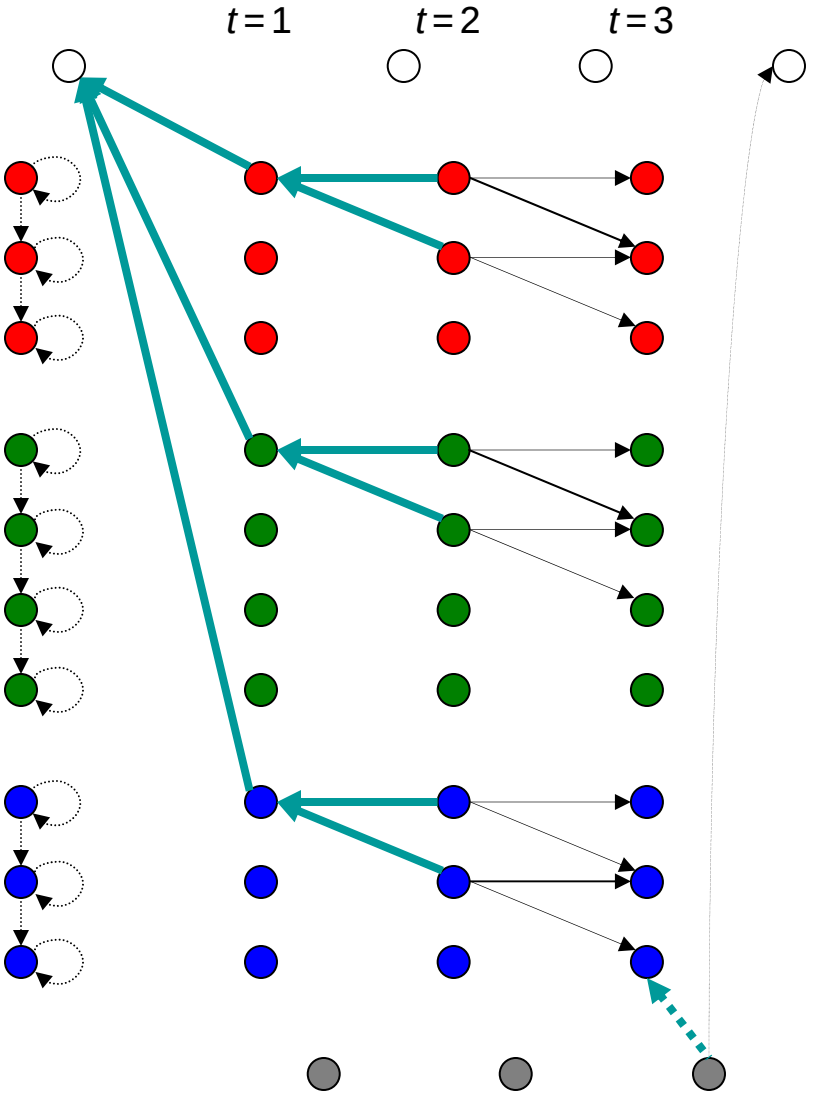


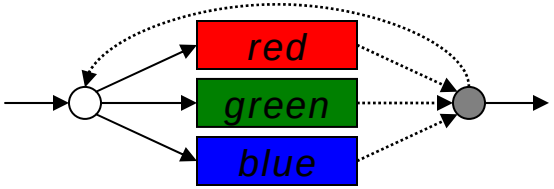
Trellis with Complete Set of Backpointers



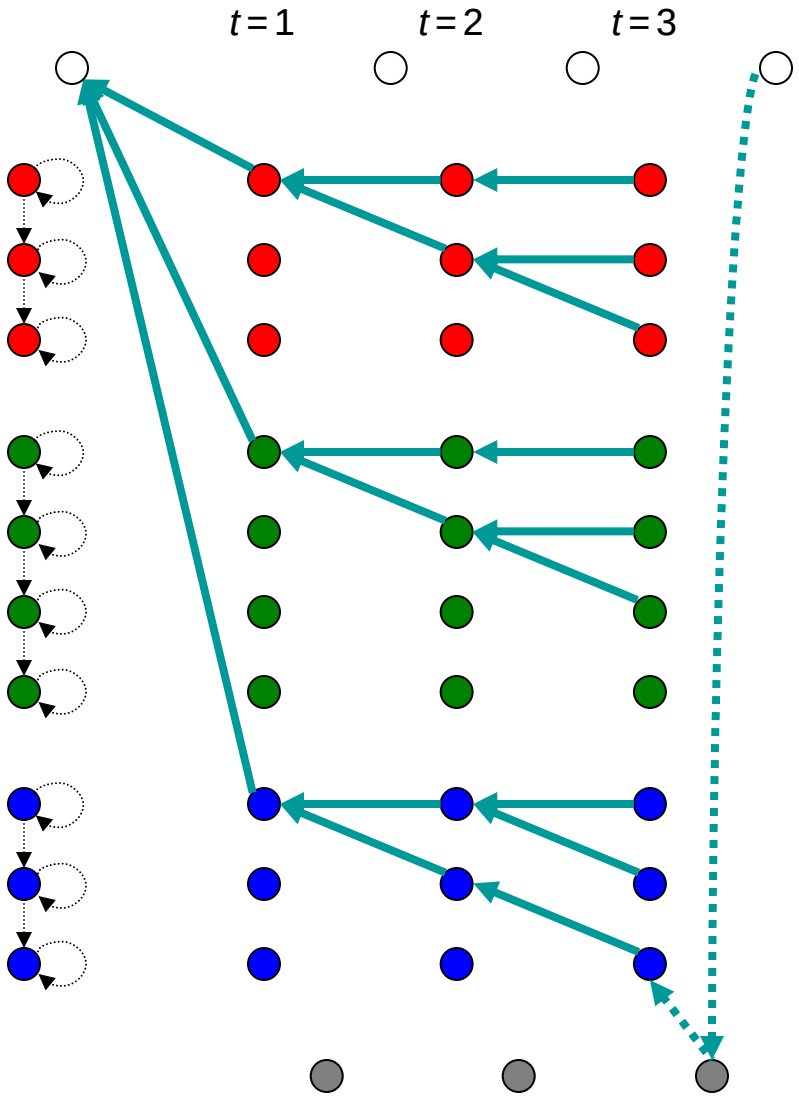


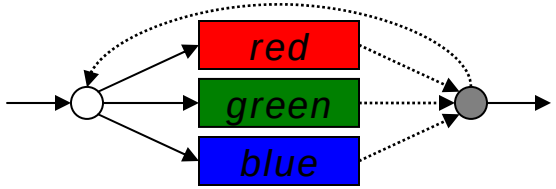
Trellis with Complete Set of Backpointers



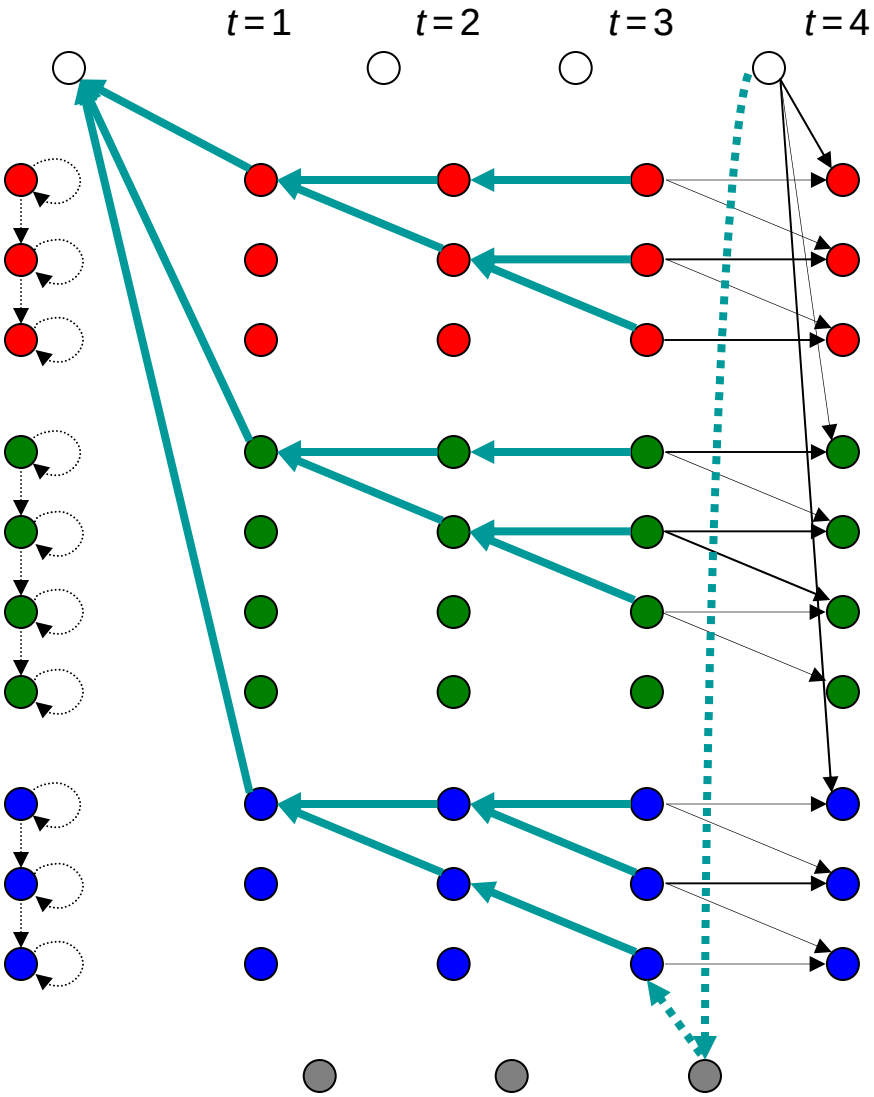


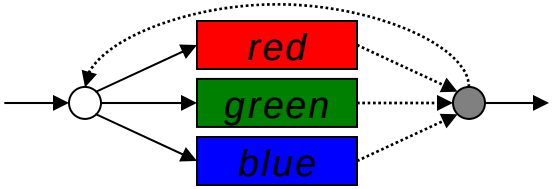
Trellis with Complete Set of Backpointers



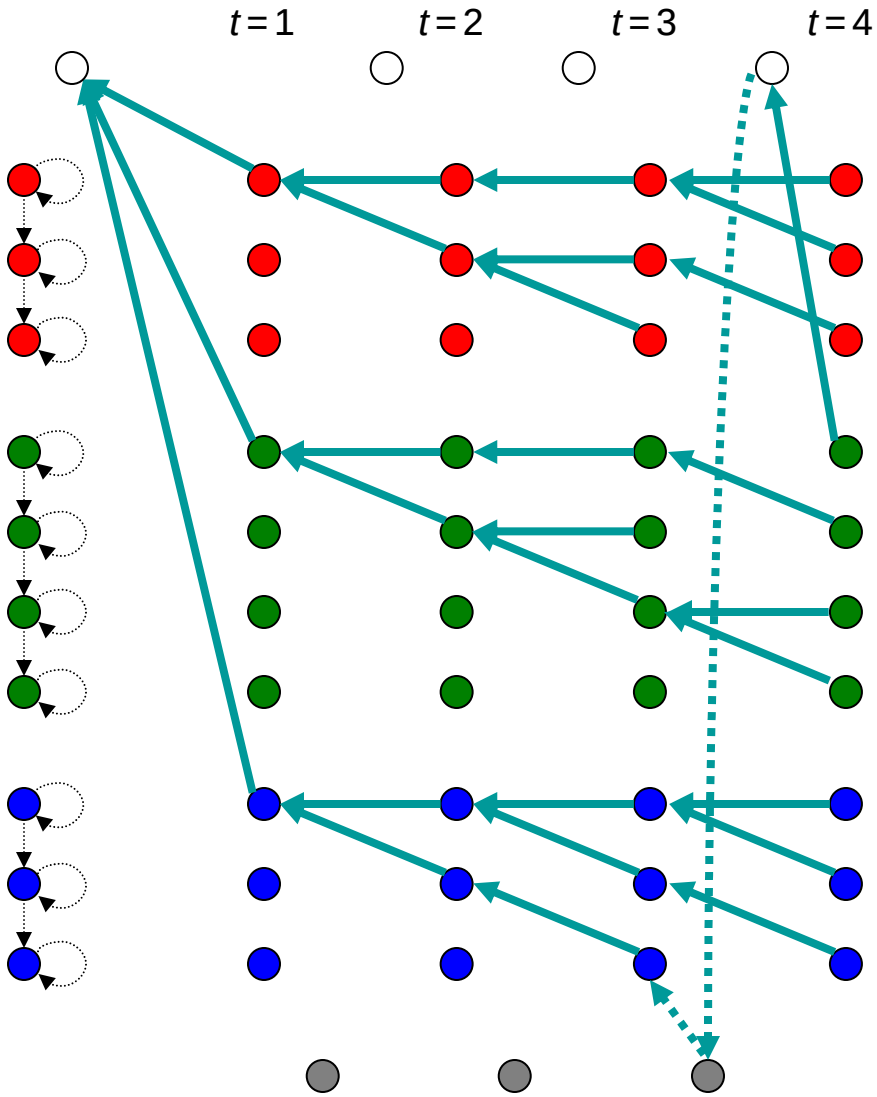


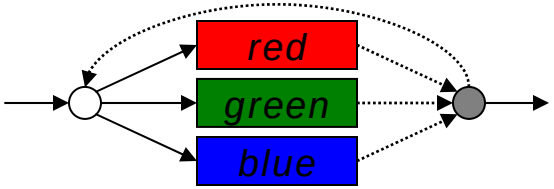
Trellis with Complete Set of Backpointers



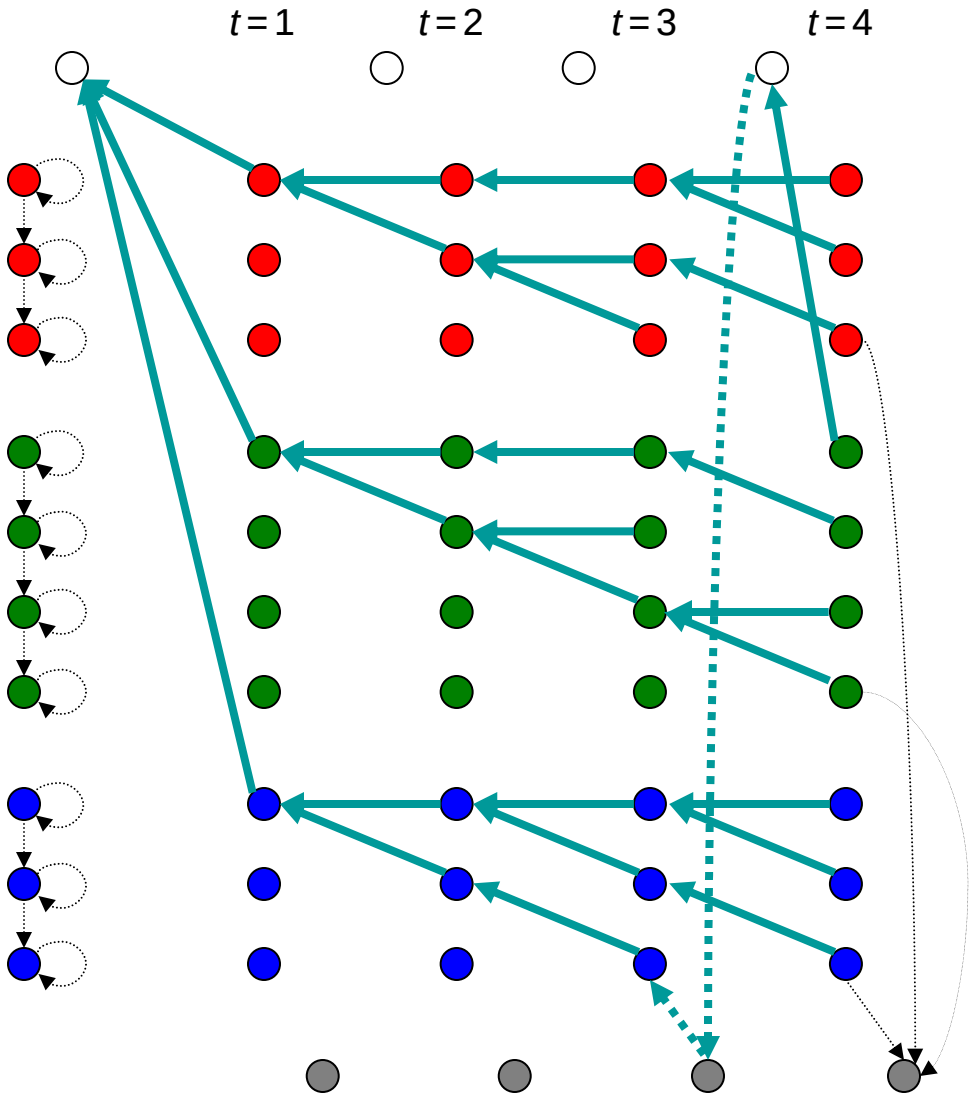


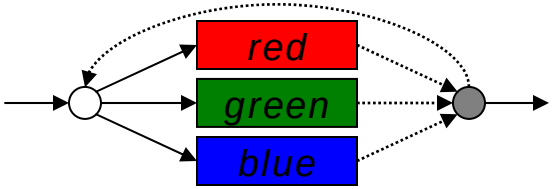
Trellis with Complete Set of Backpointers



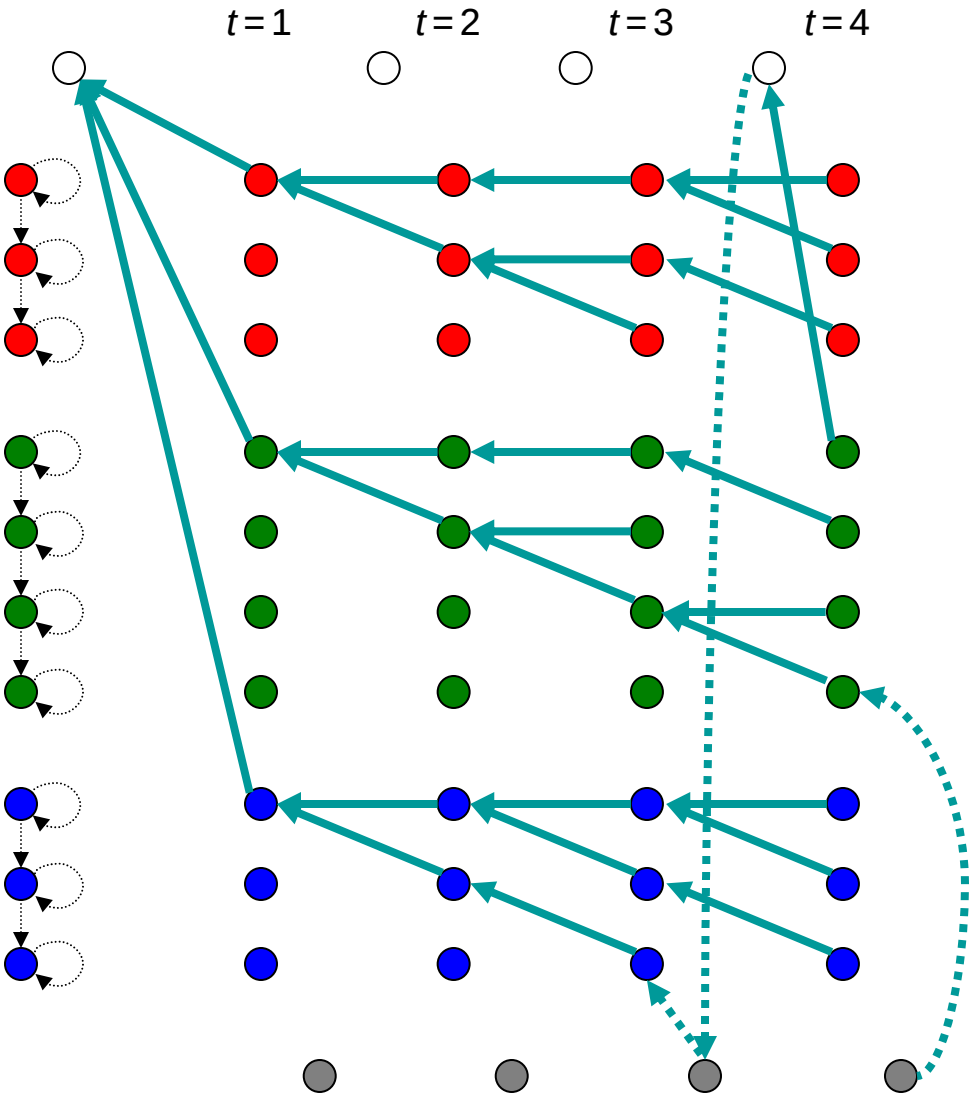


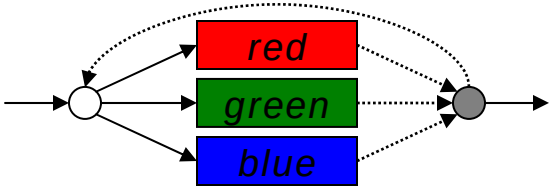
Trellis with Complete Set of Backpointers



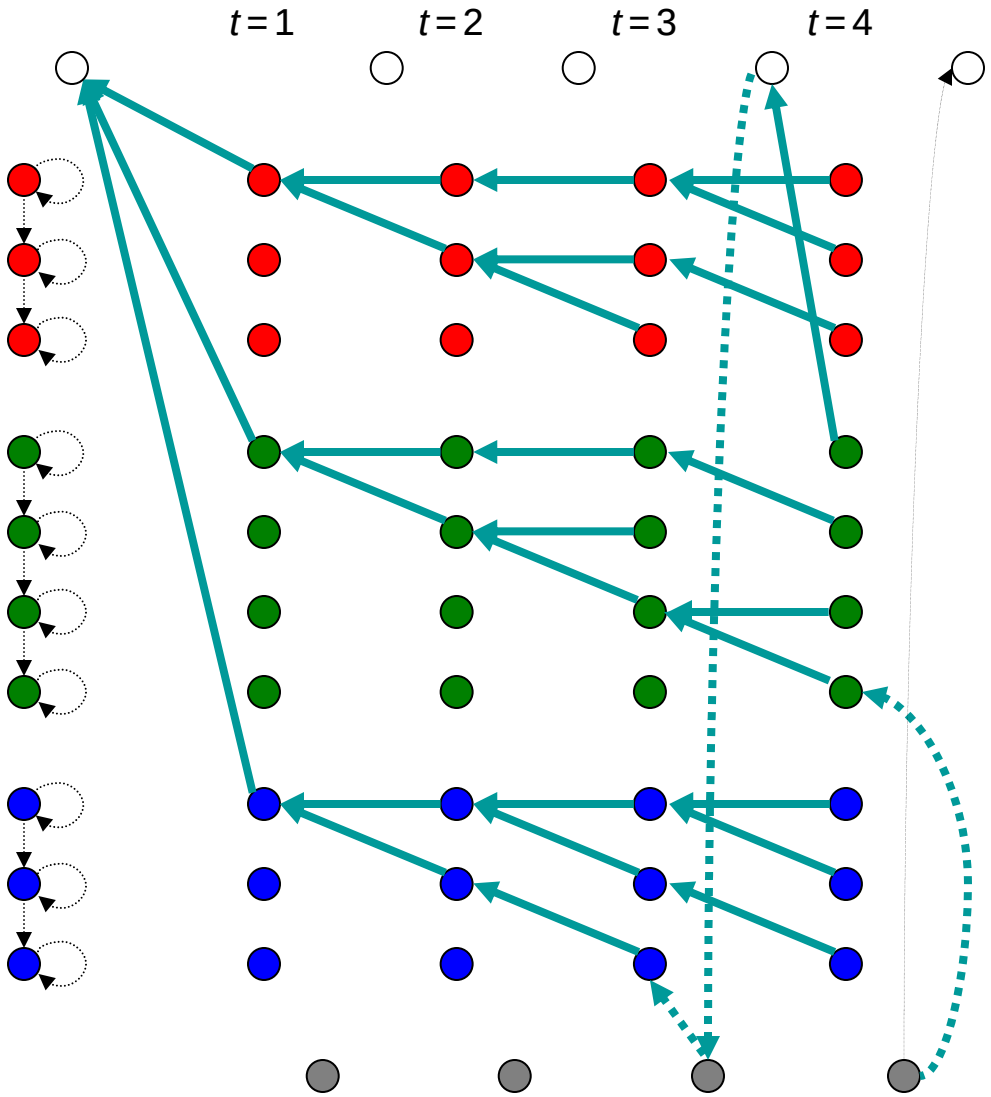


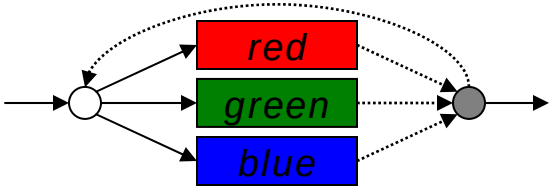
Trellis with Complete Set of Backpointers



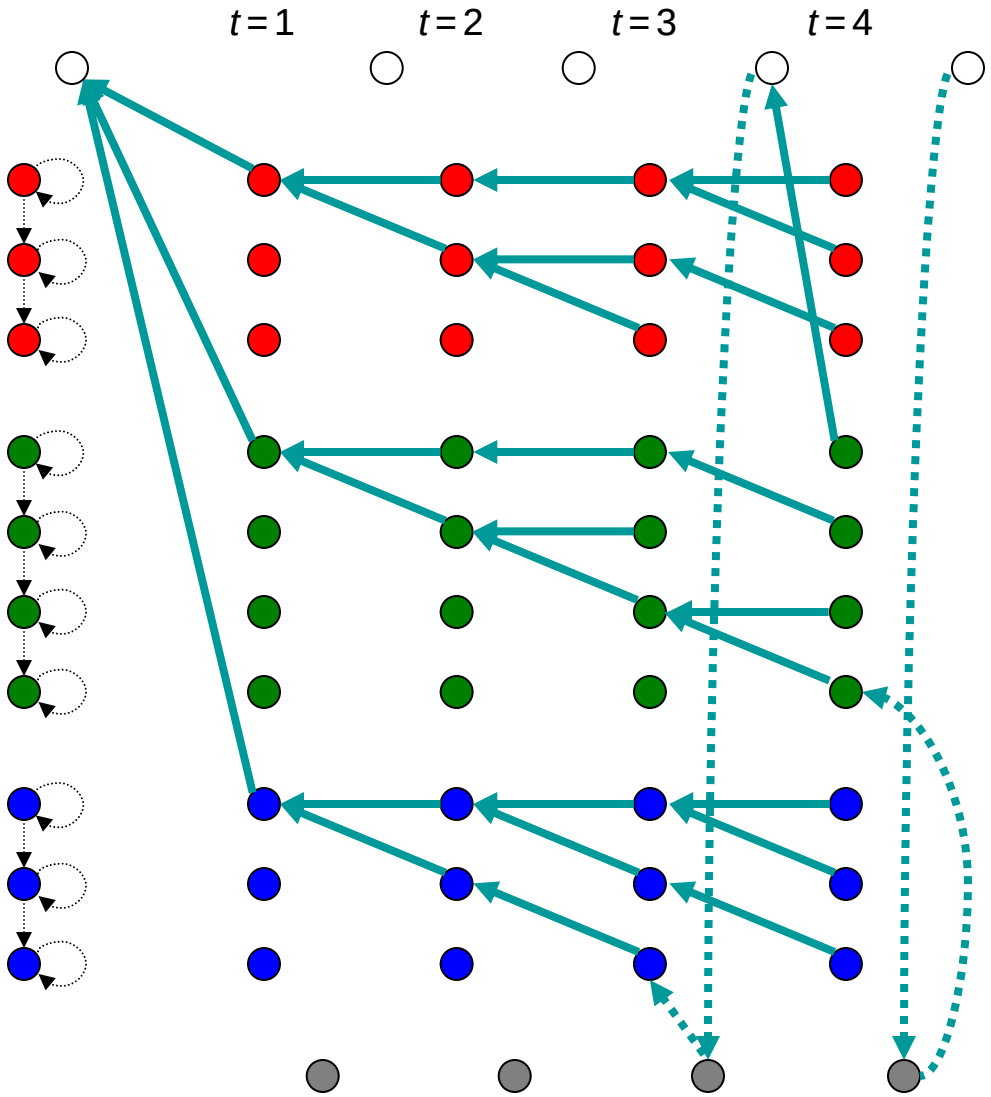


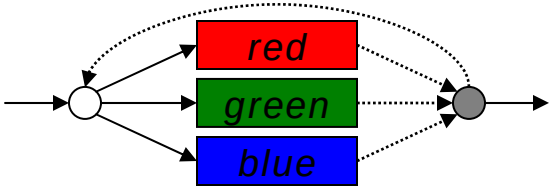
Trellis with Complete Set of Backpointers



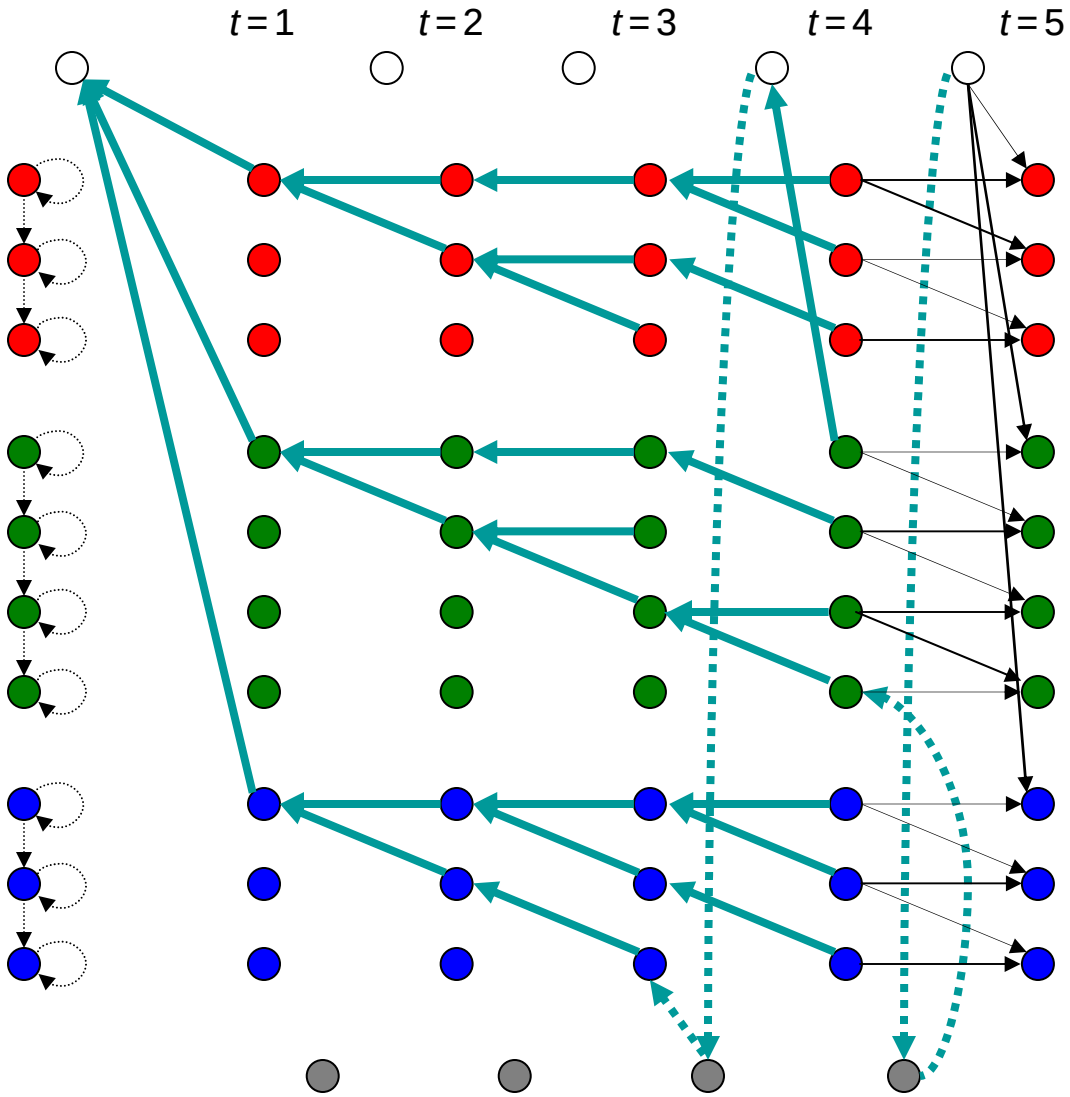


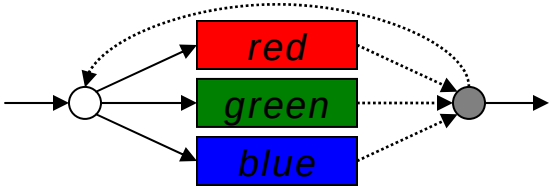
Trellis with Complete Set of Backpointers



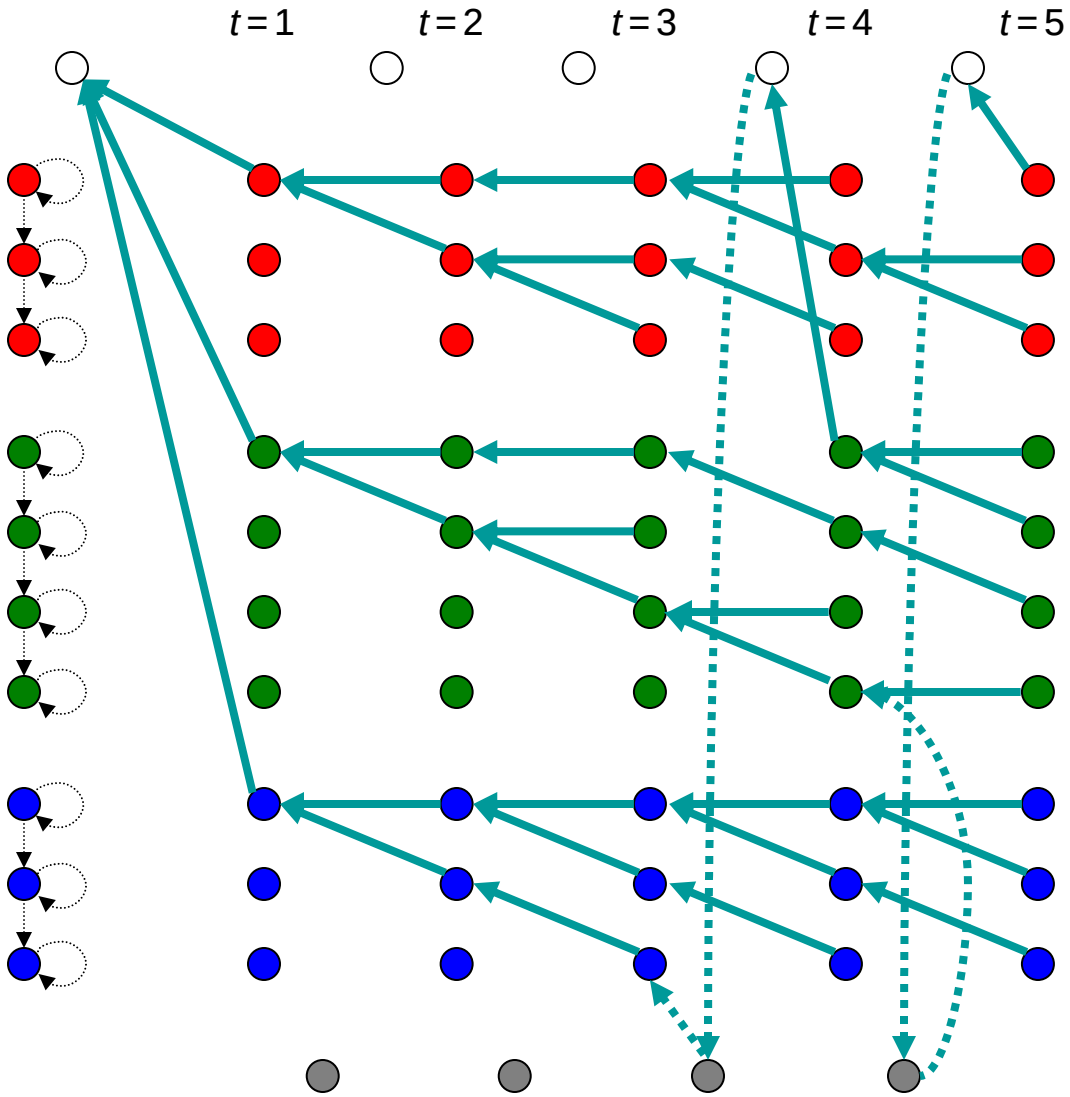


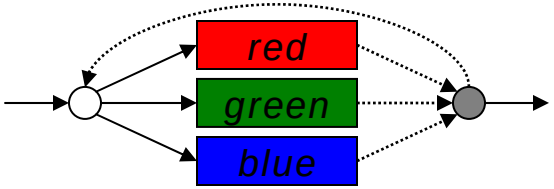
Trellis with Complete Set of Backpointers



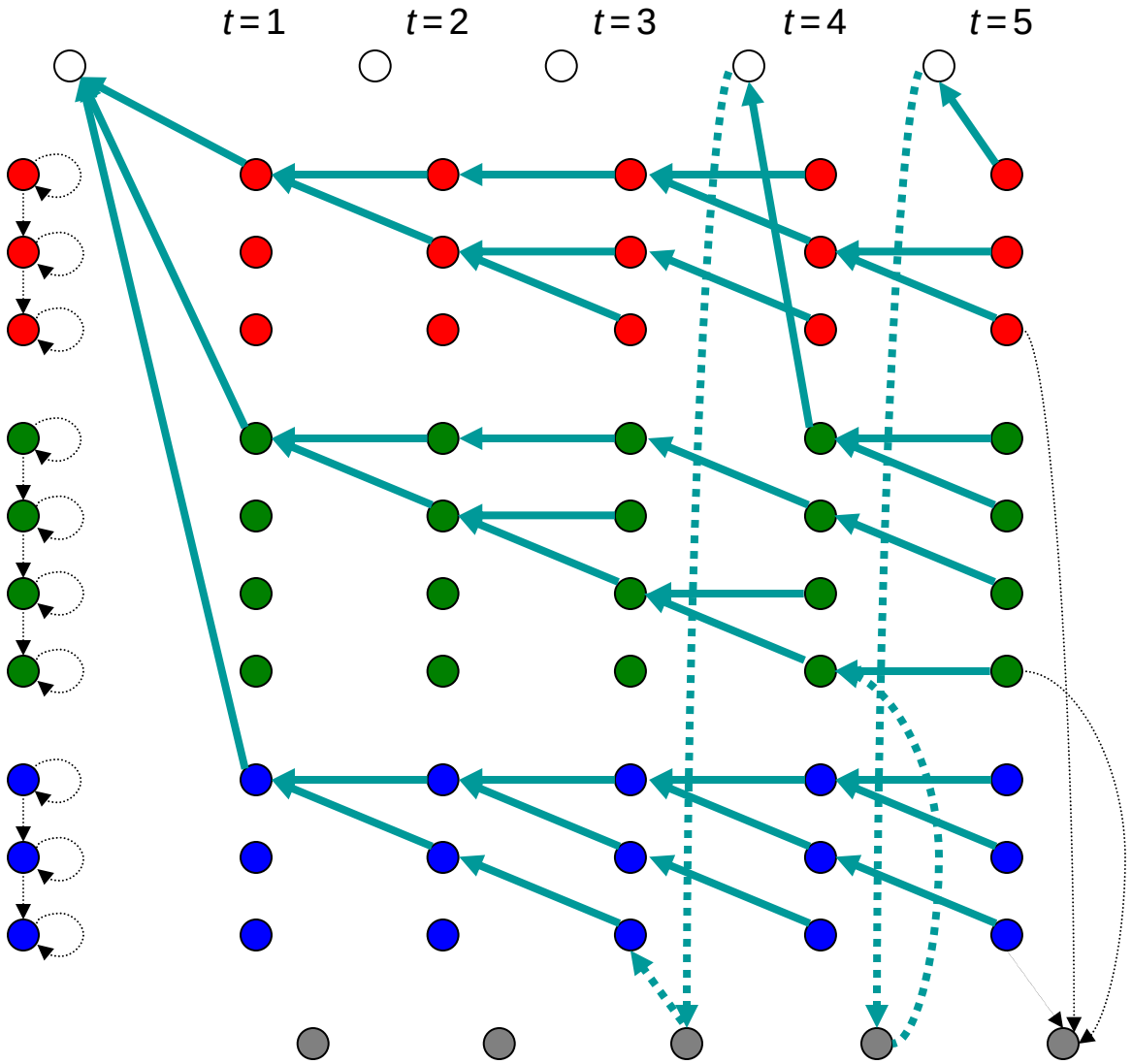


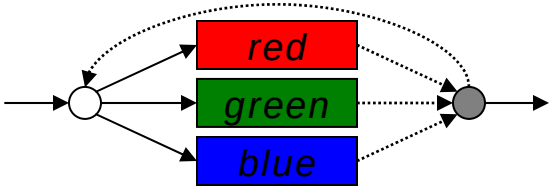
Trellis with Complete Set of Backpointers



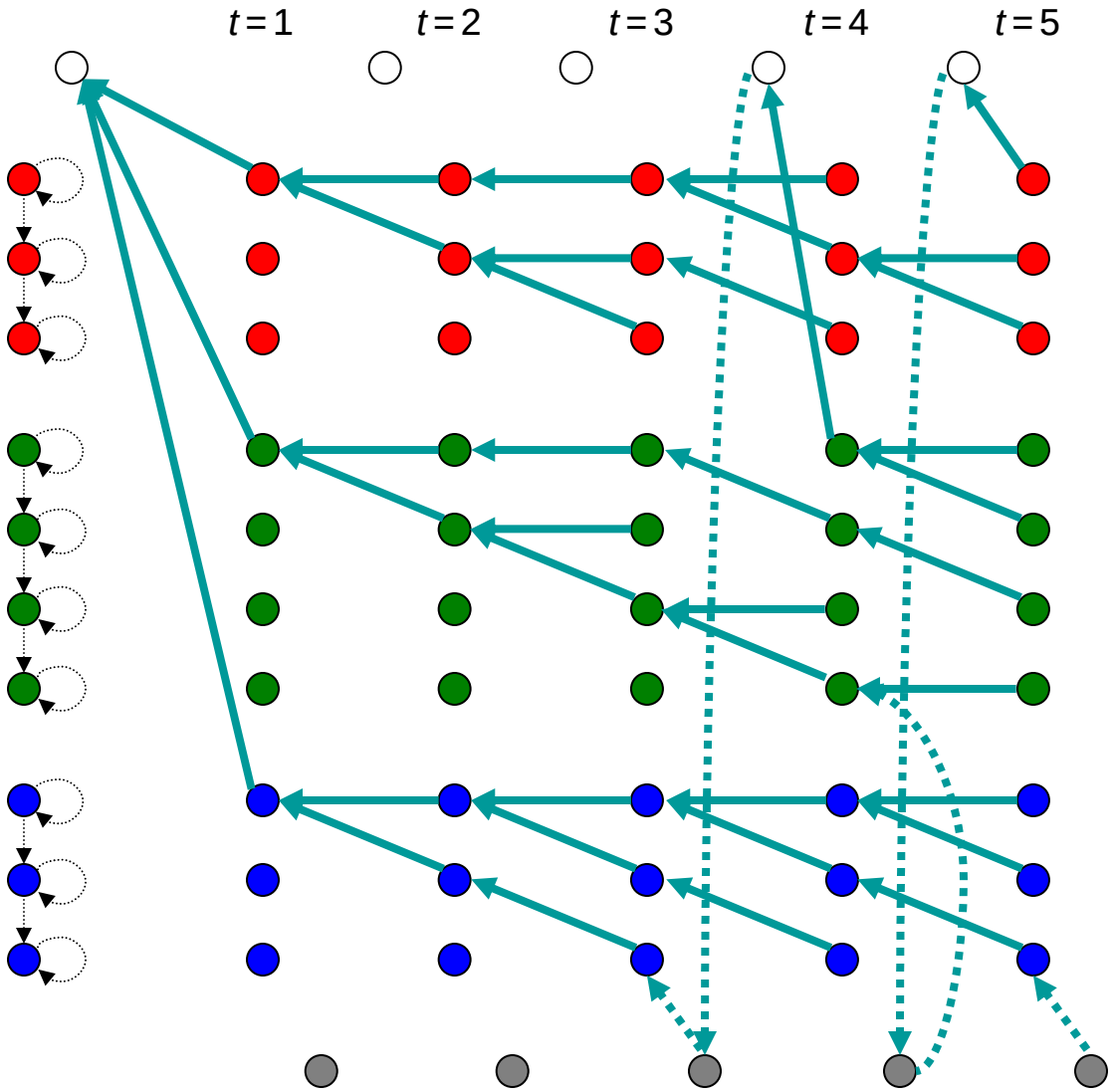


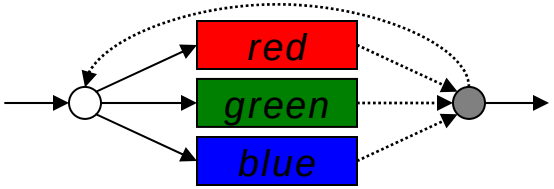
Trellis with Complete Set of Backpointers



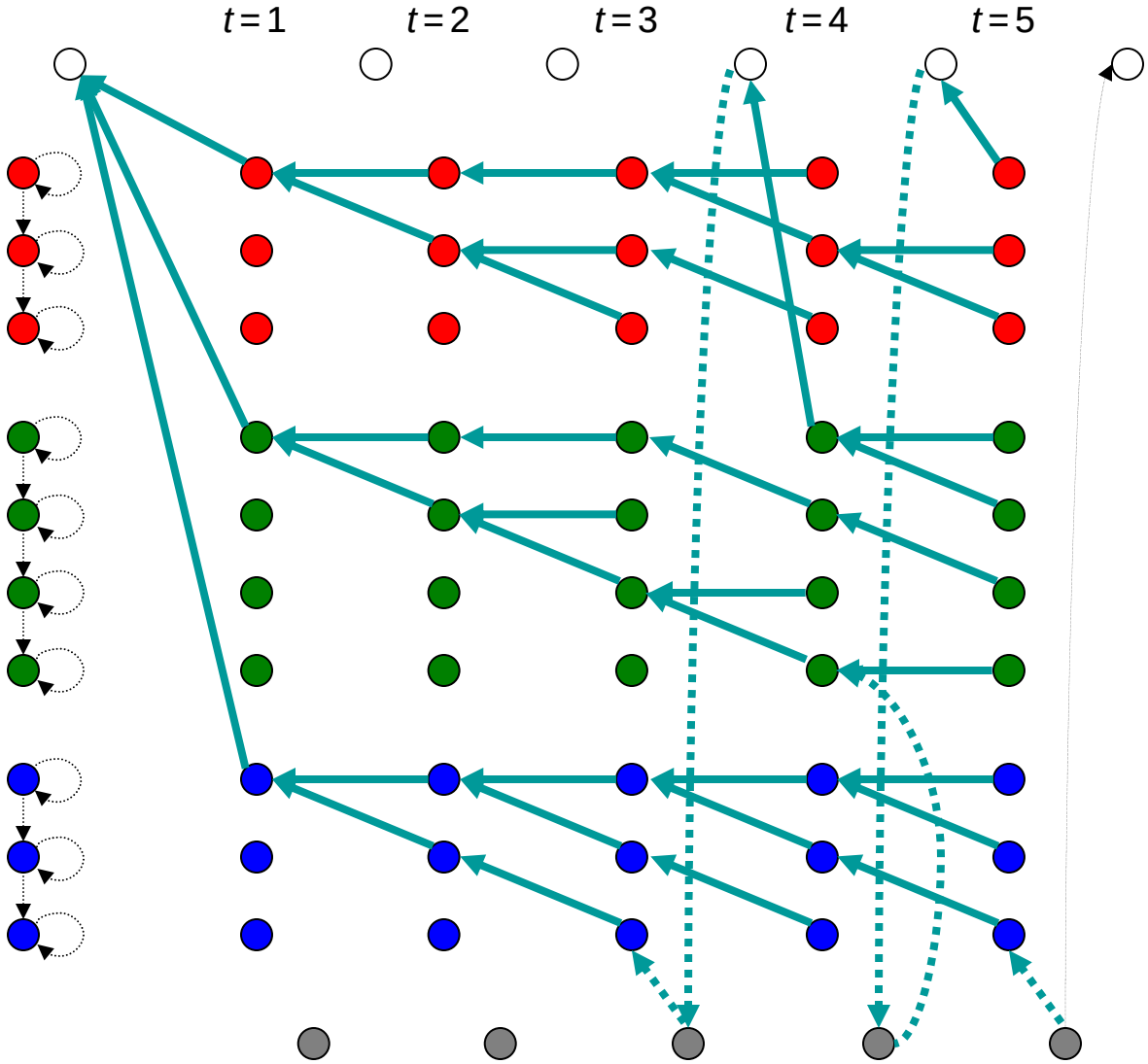


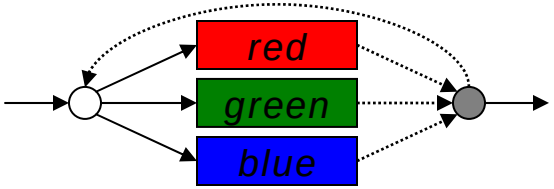
Trellis with Complete Set of Backpointers



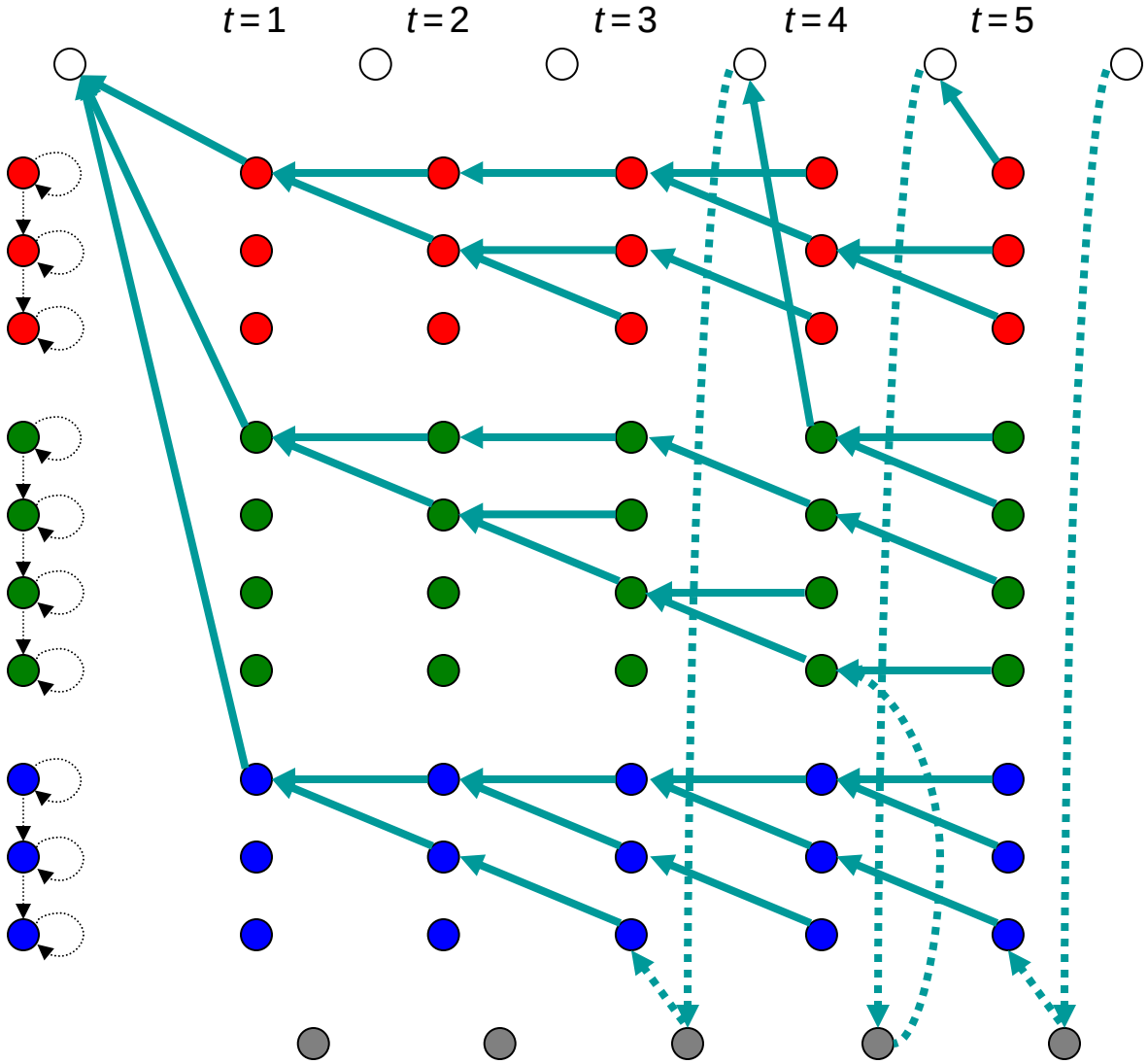


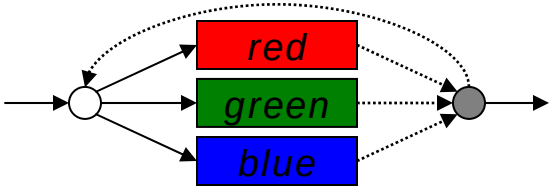
Trellis with Complete Set of Backpointers



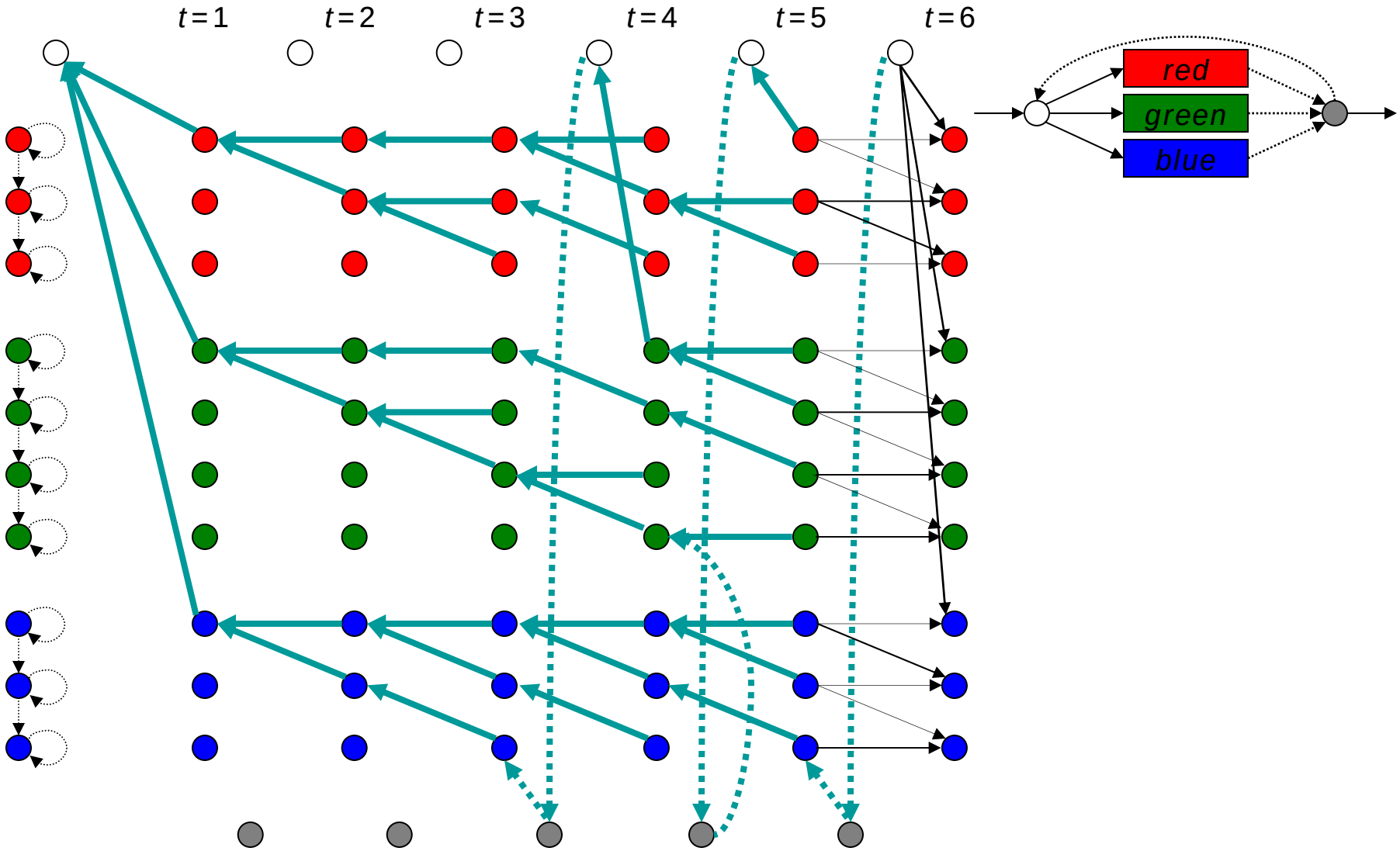


Trellis with Complete Set of Backpointers

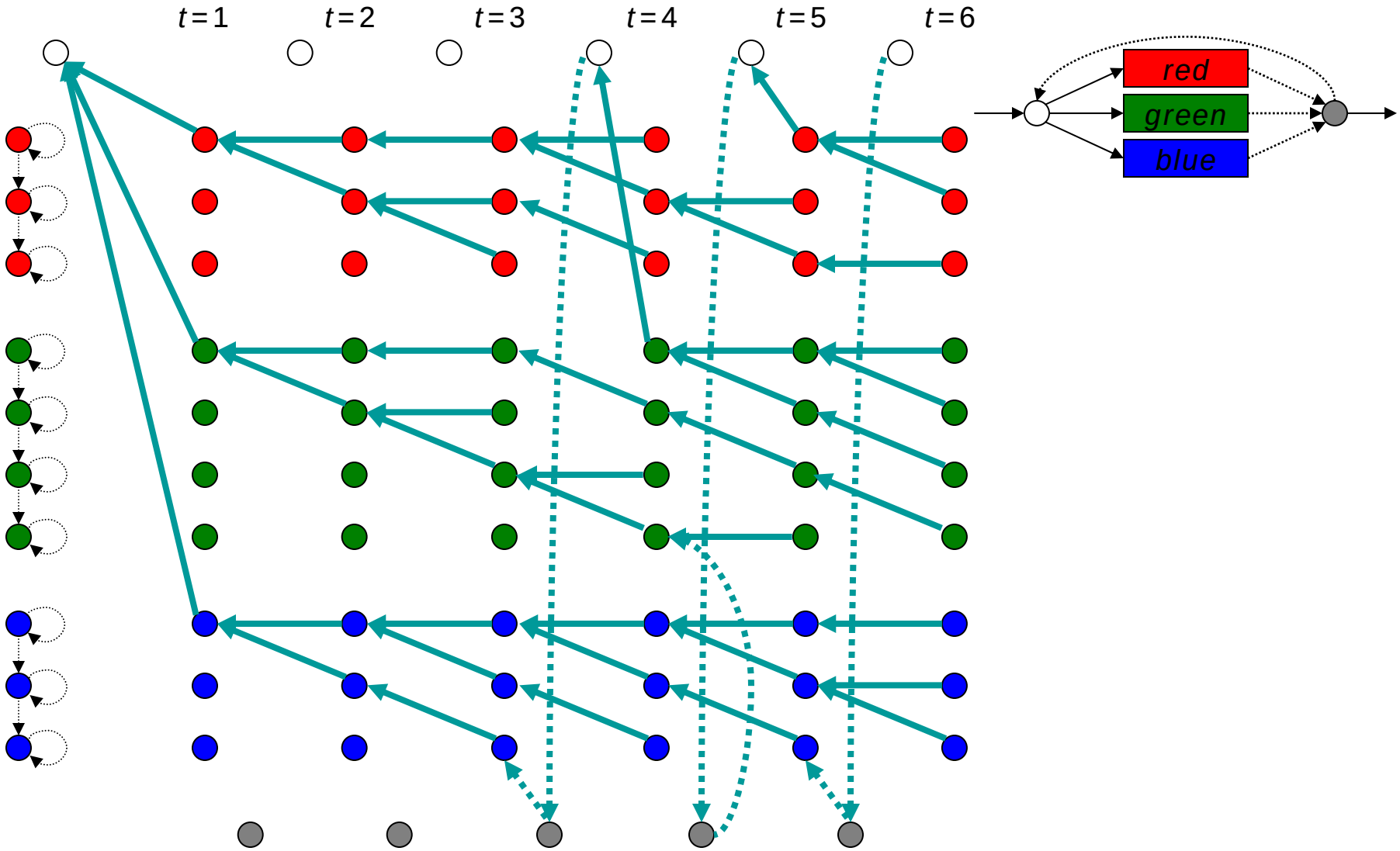




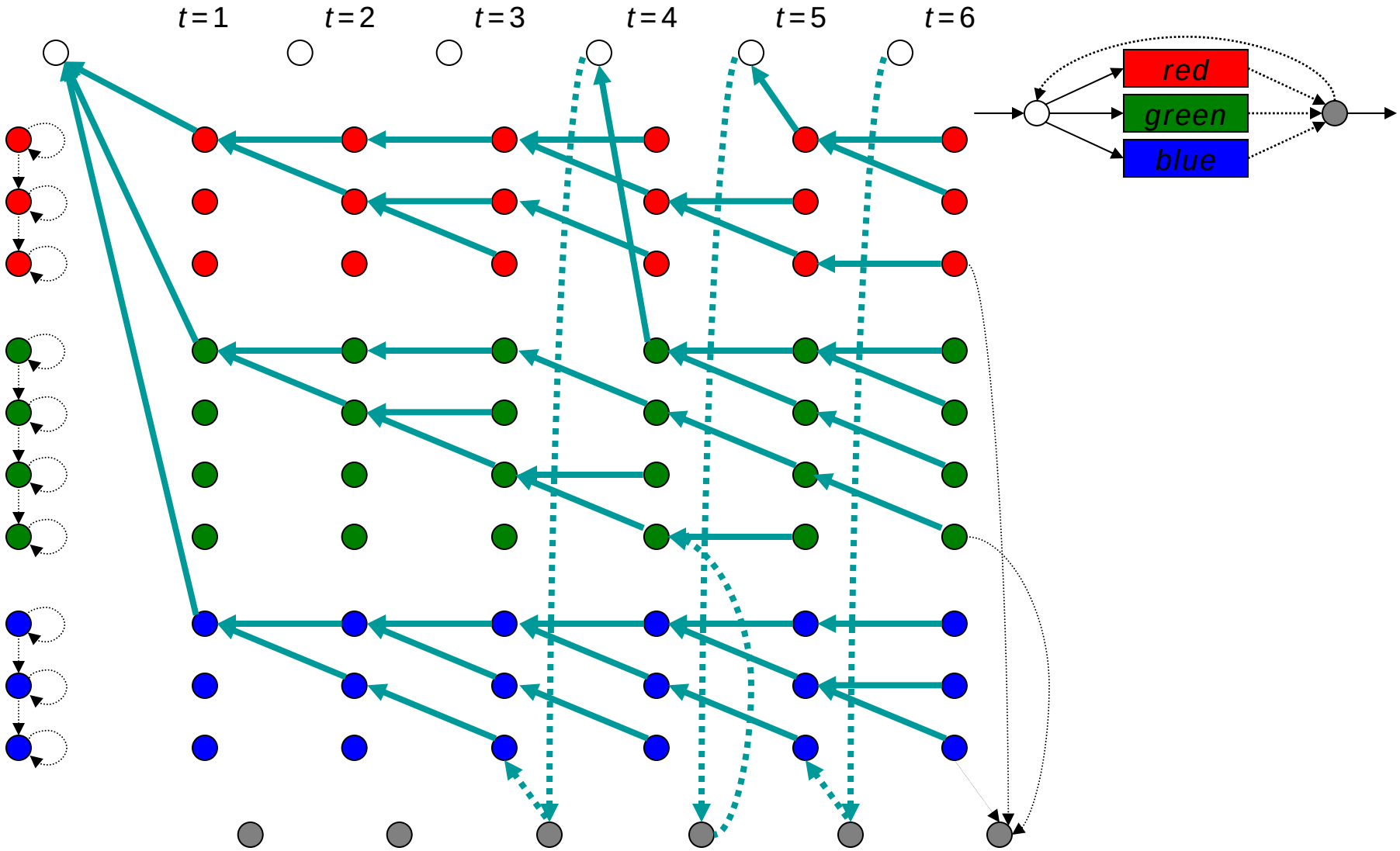
Trellis with Complete Set of Backpointers



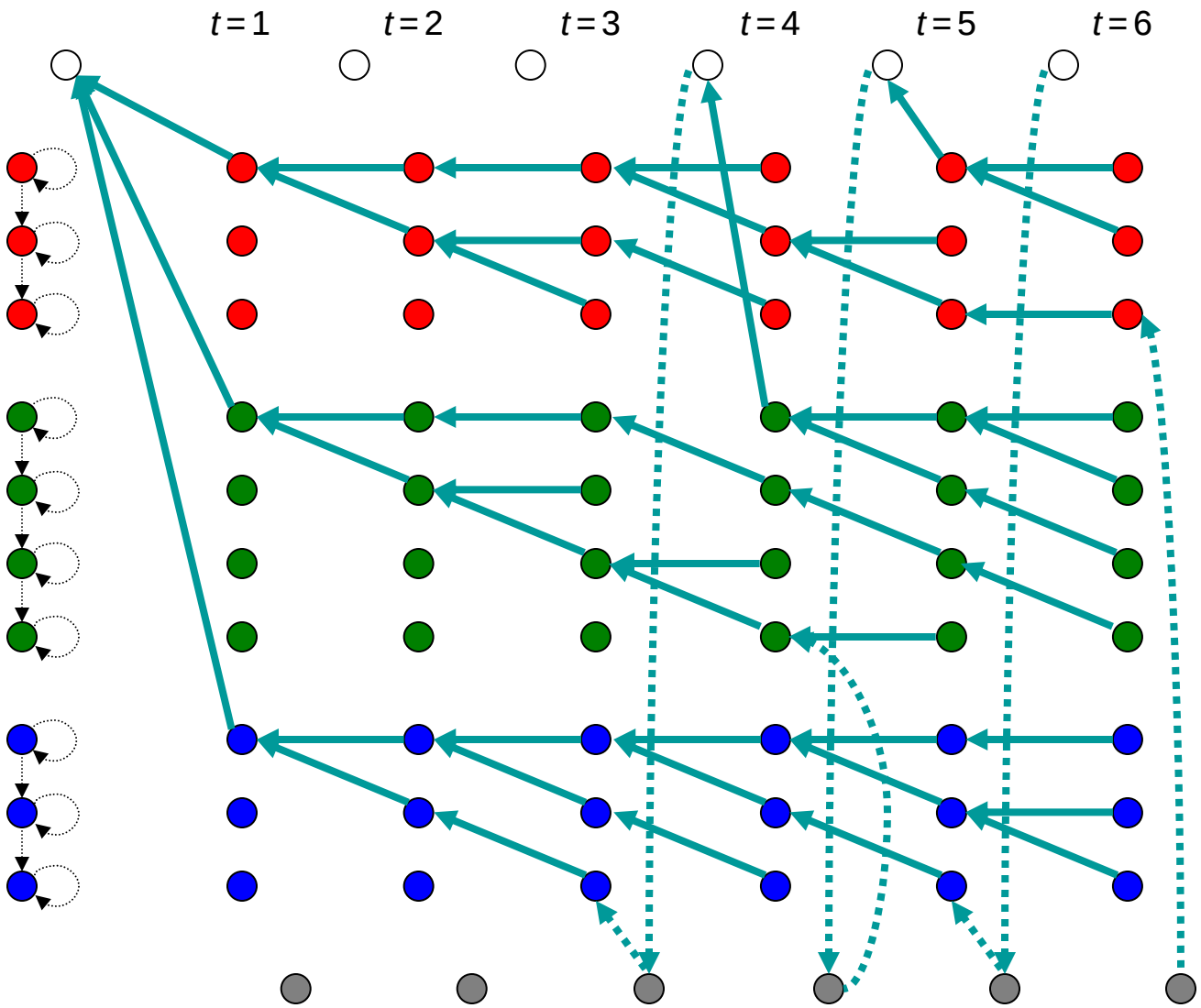
Trellis with Complete Set of Backpointers



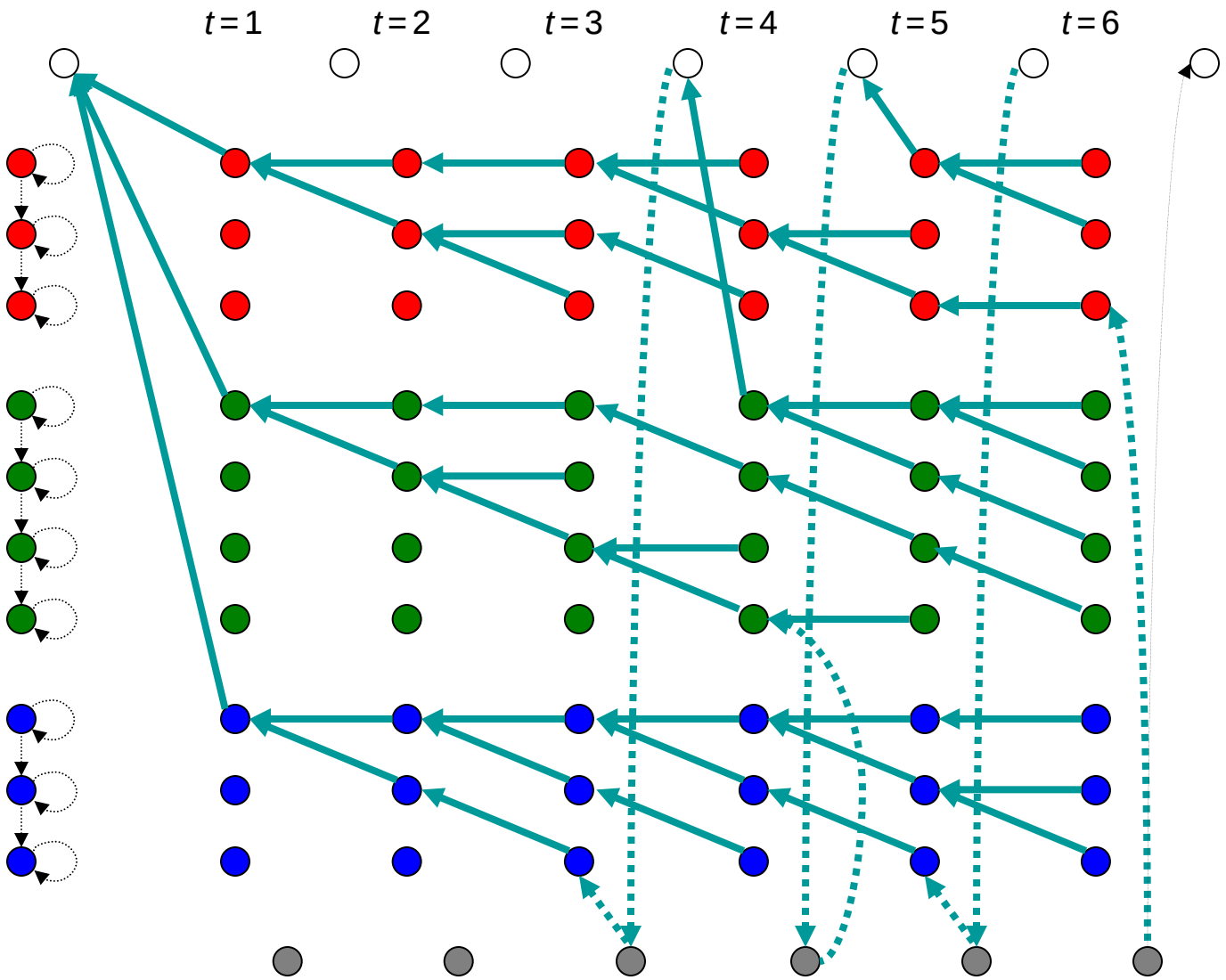
Trellis with Complete Set of Backpointers



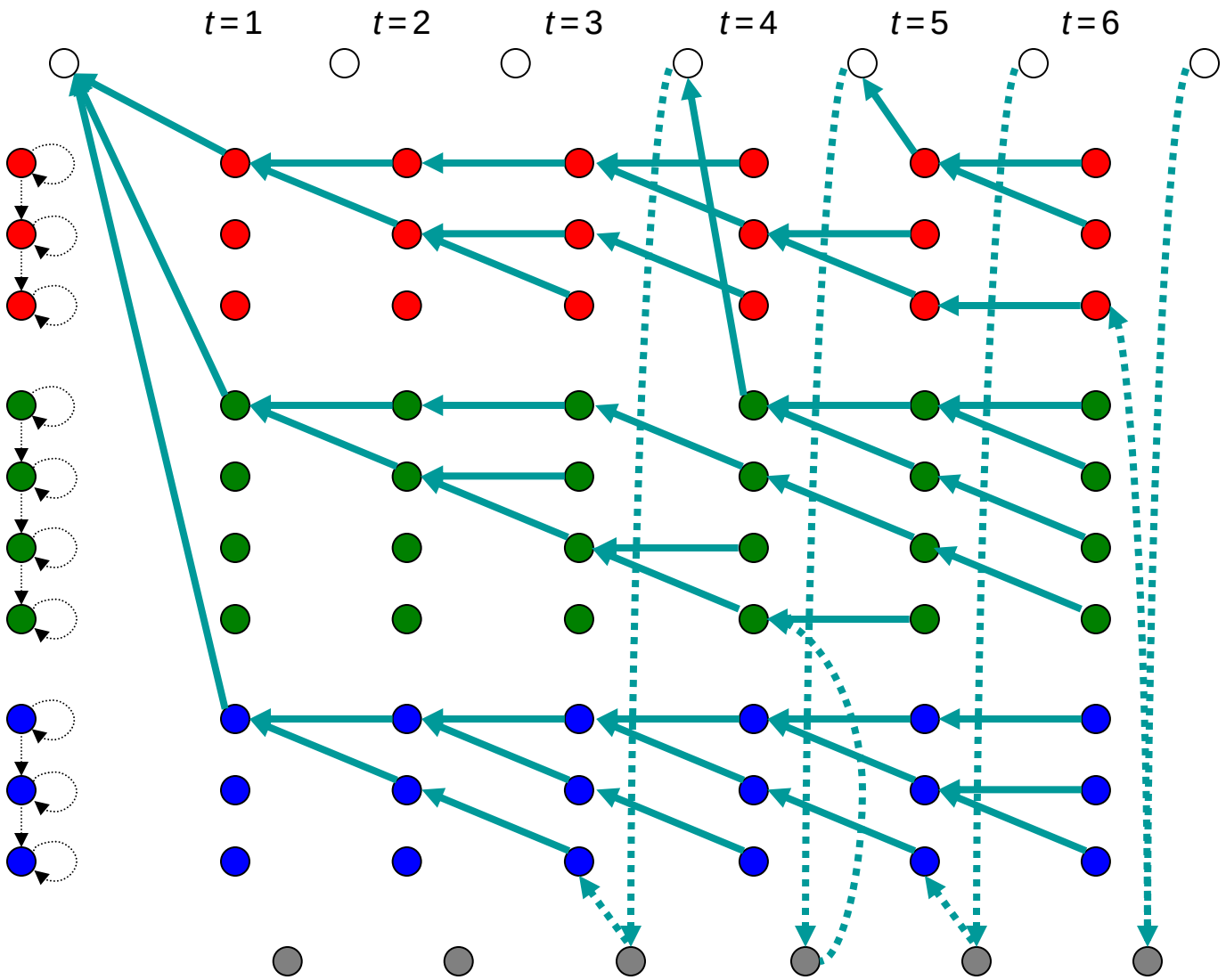
Trellis with Complete Set of Backpointers



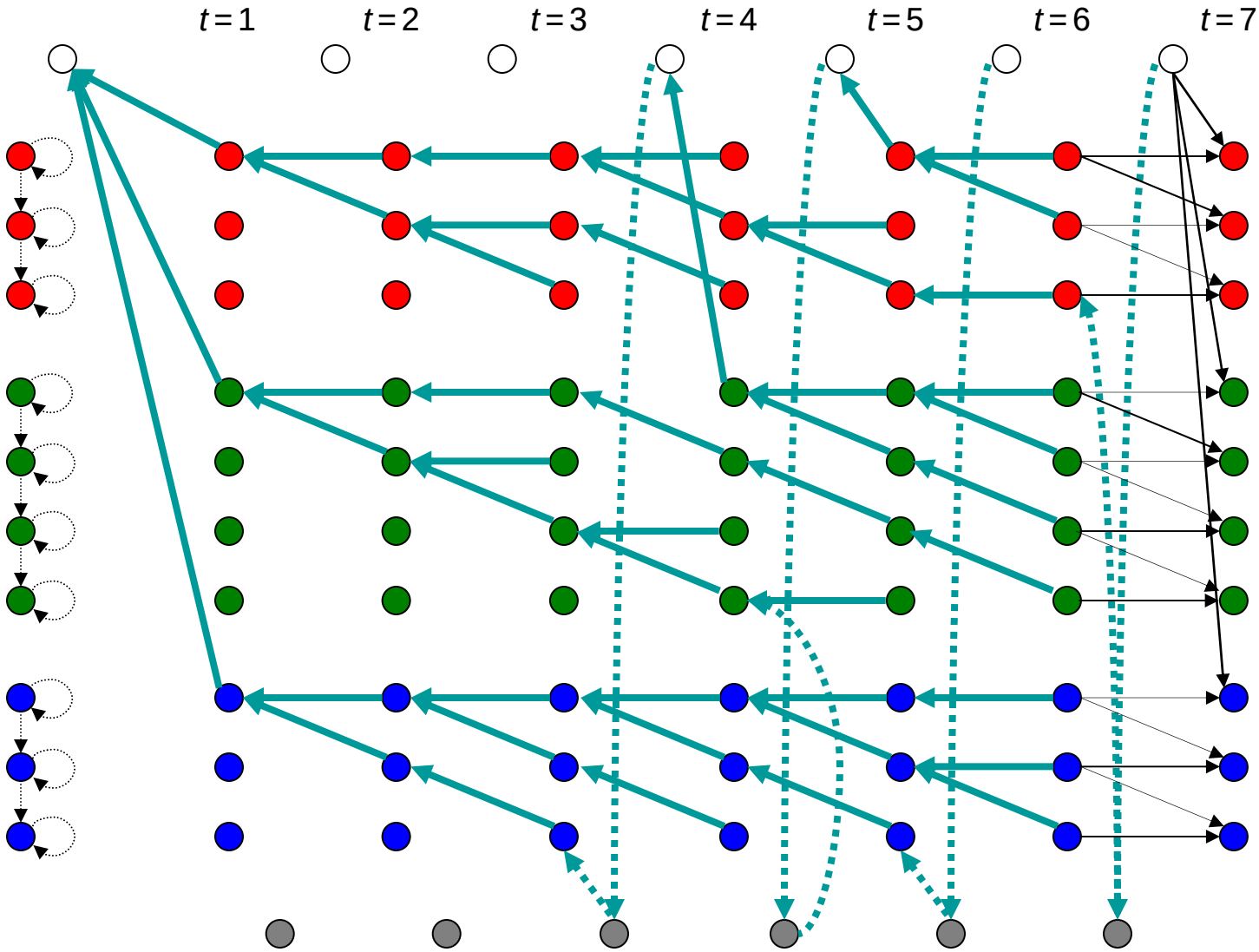
Trellis with Complete Set of Backpointers



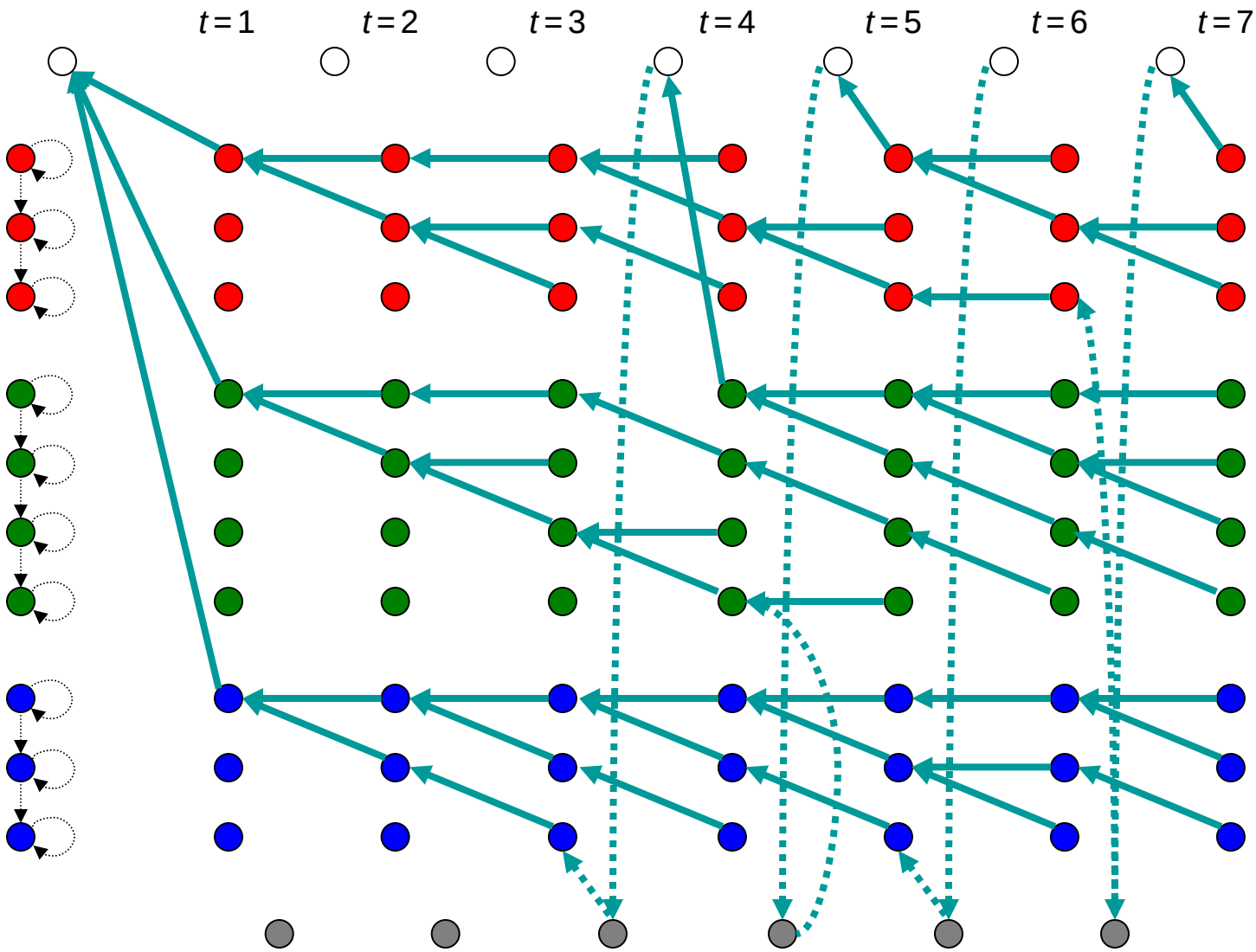
Trellis with Complete Set of Backpointers



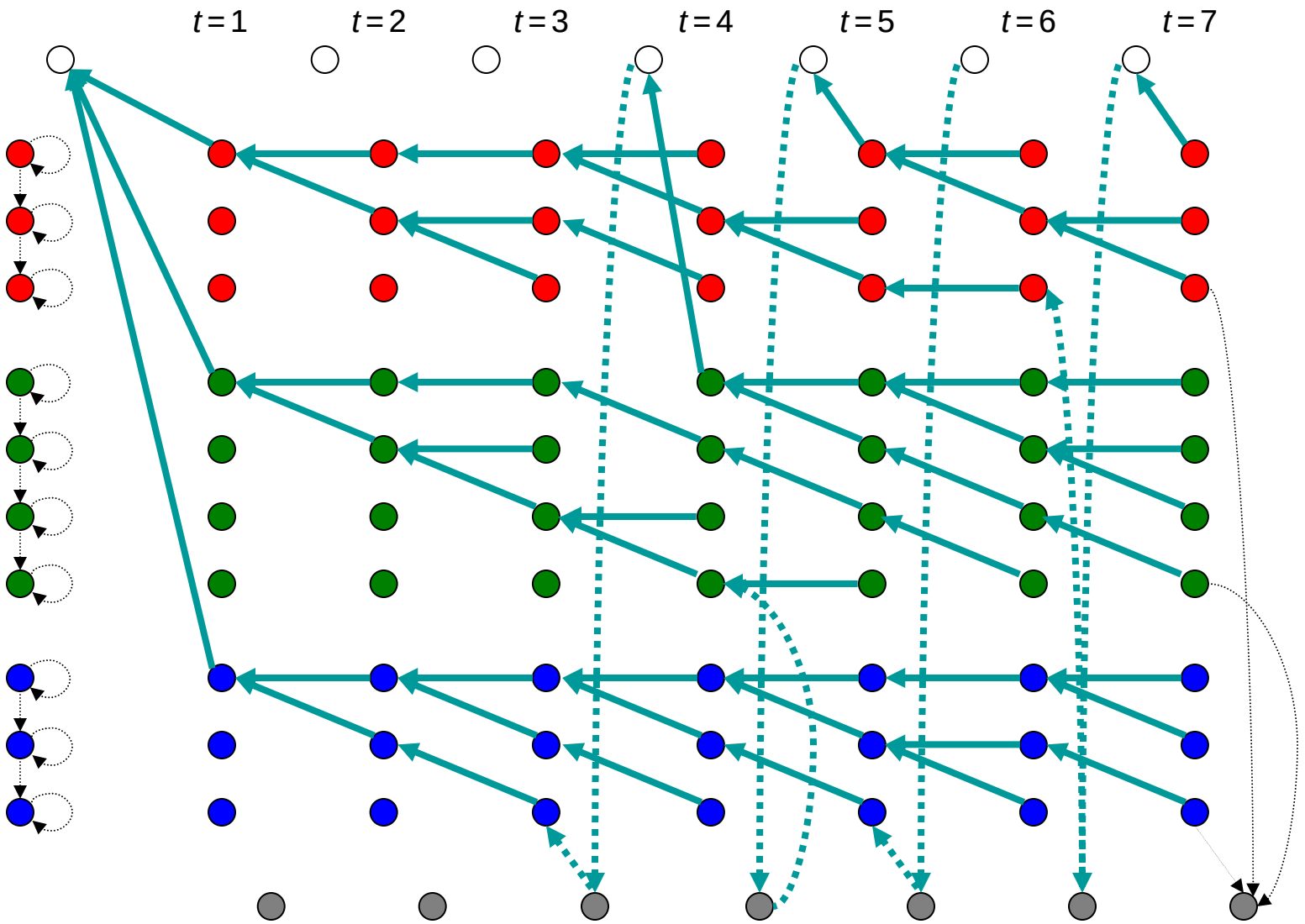
Trellis with Complete Set of Backpointers



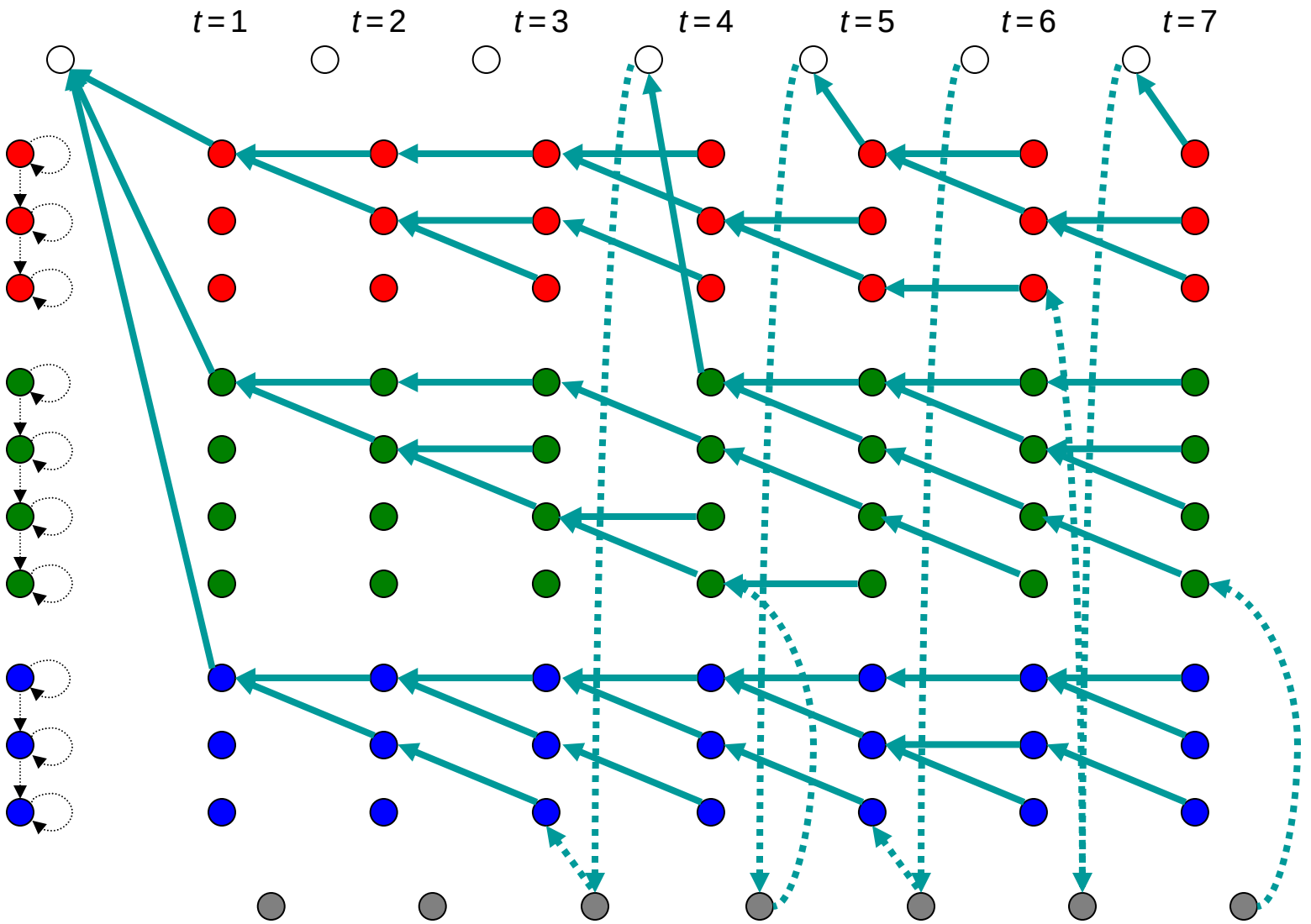
Trellis with Complete Set of Backpointers



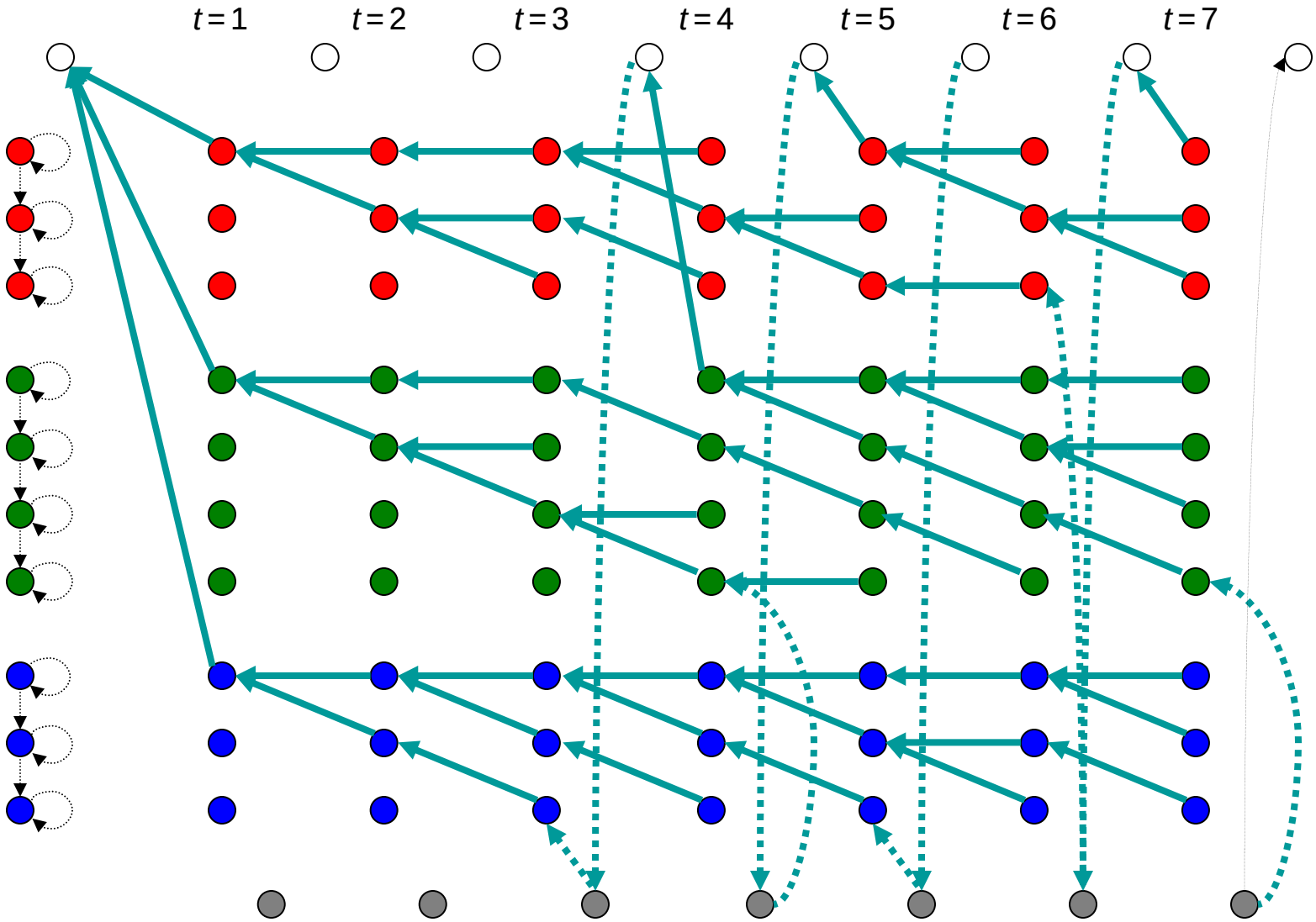
Trellis with Complete Set of Backpointers



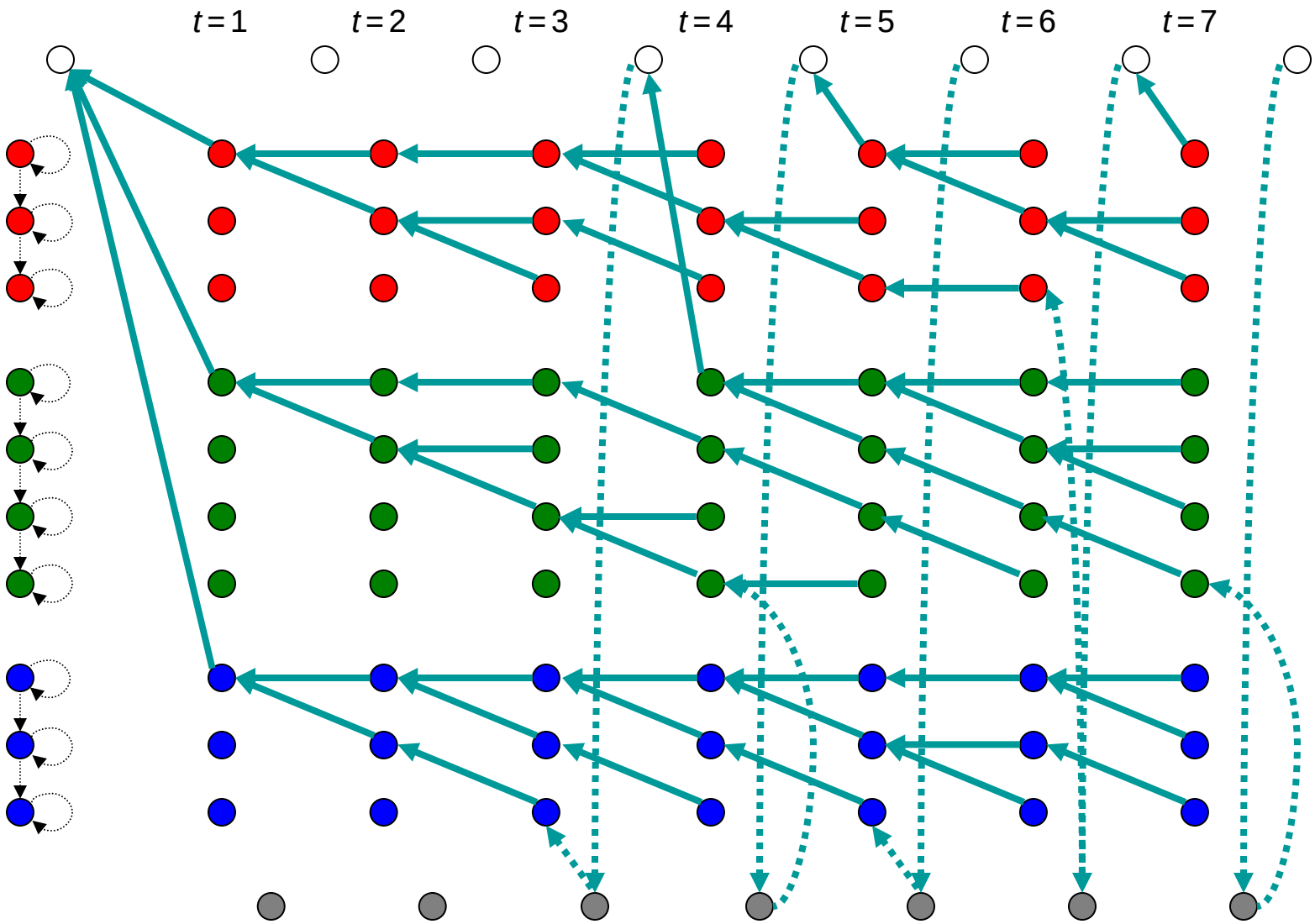
Trellis with Complete Set of Backpointers



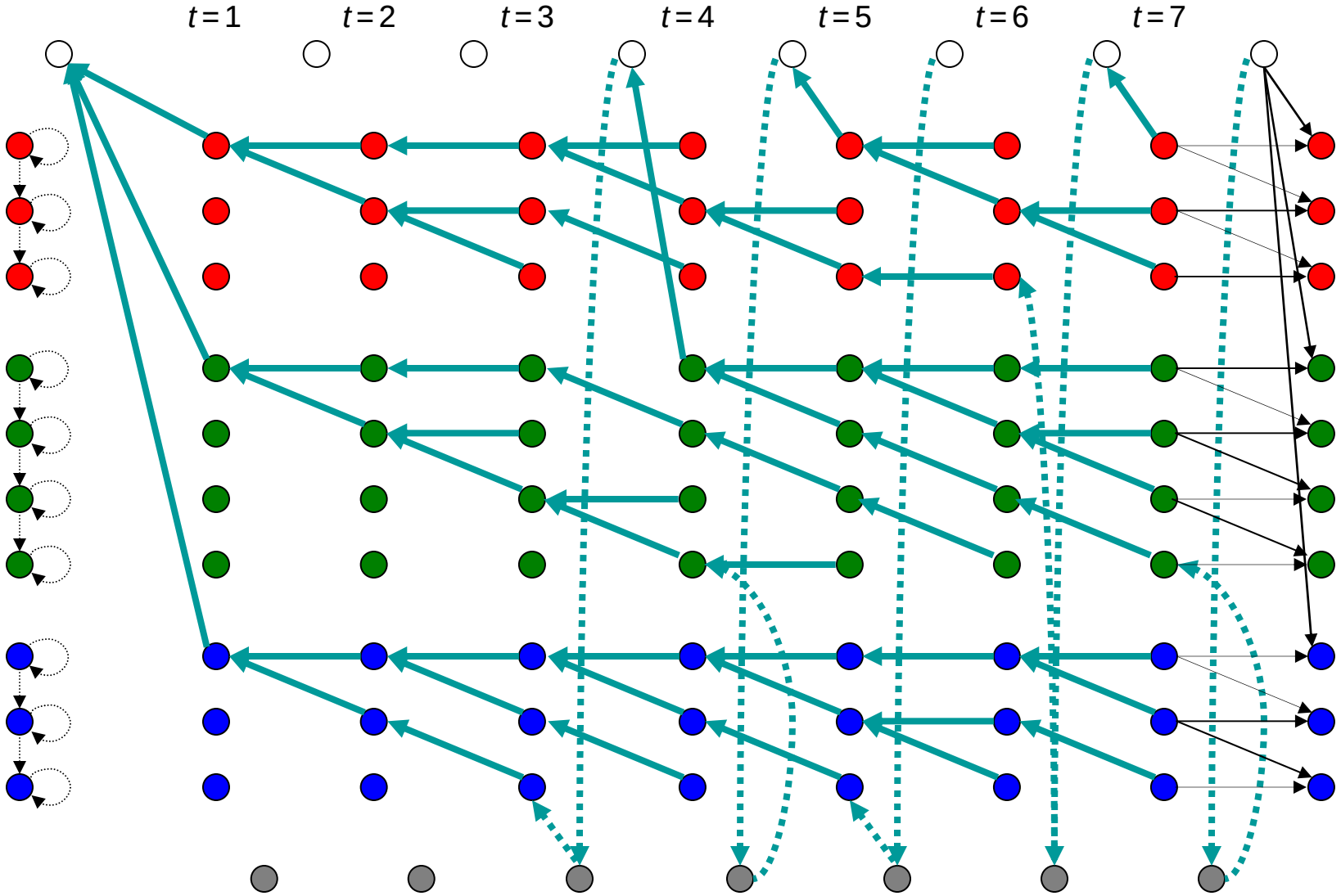
Trellis with Complete Set of Backpointers



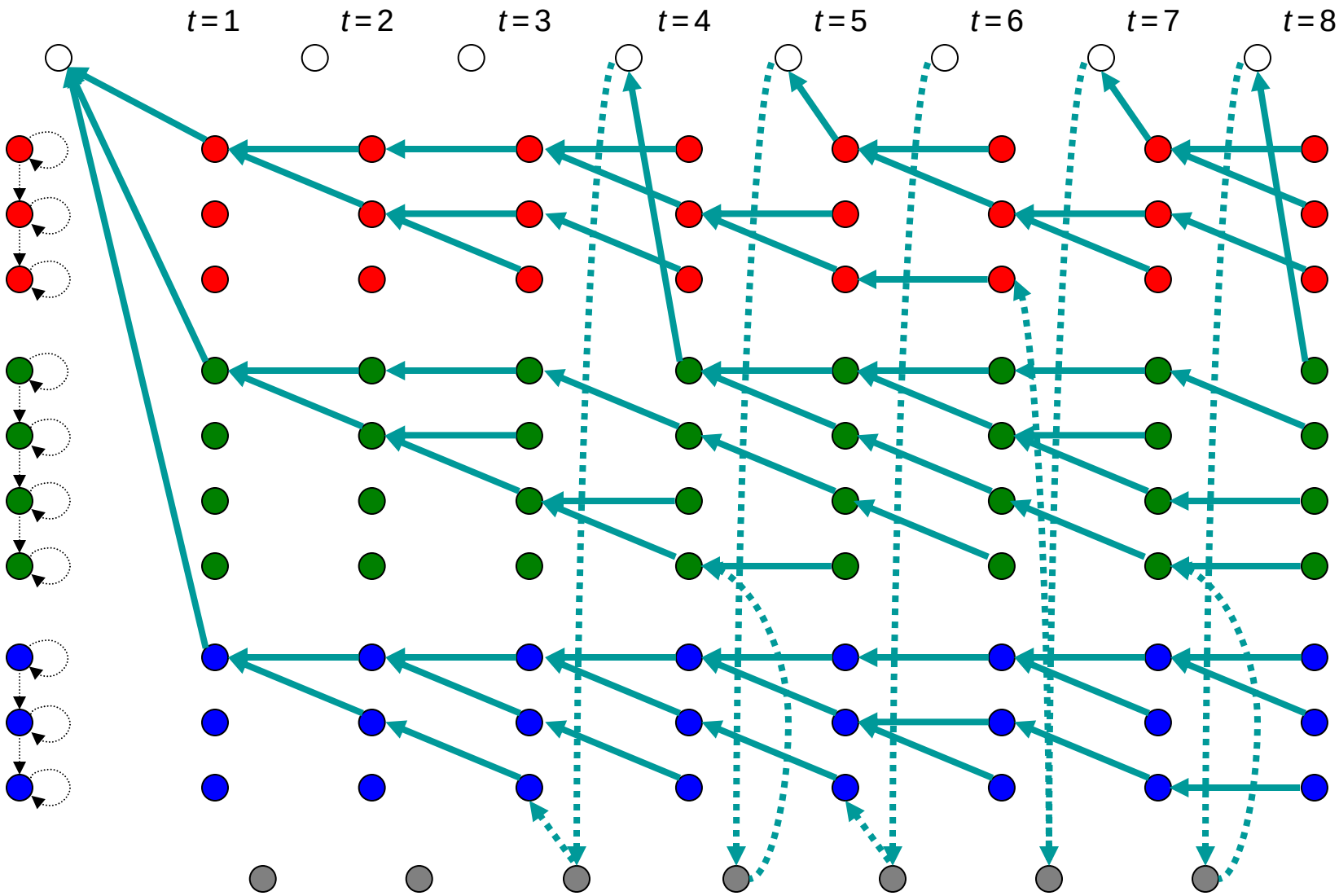
Trellis with Complete Set of Backpointers



Trellis with Complete Set of Backpointers



Trellis with Complete Set of Backpointers



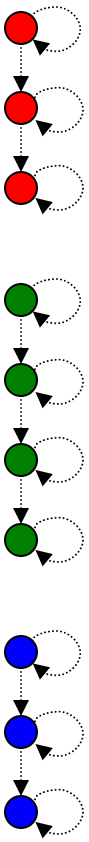
Using a Backpointer Table

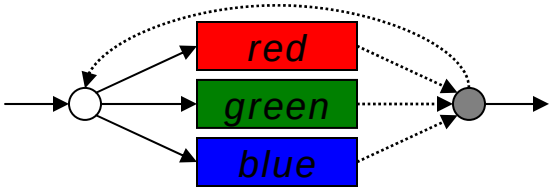
- Only retain back pointers to the entry into words

Trellis with Complete Set of Backpointers

1

1, t=0, scr1,p=0,...

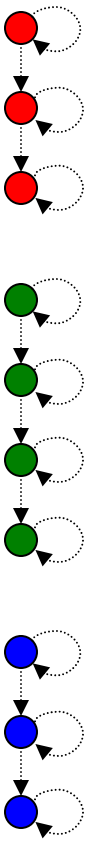


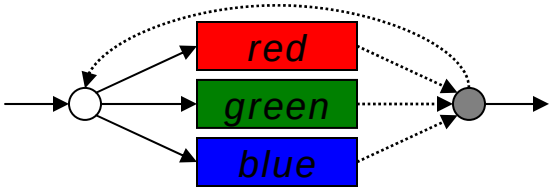


Trellis with Complete Set of Backpointers

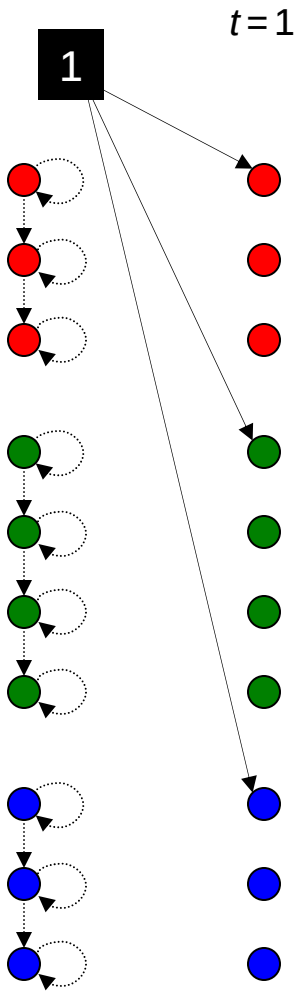
1

1, t=0, scr1,p=0,...

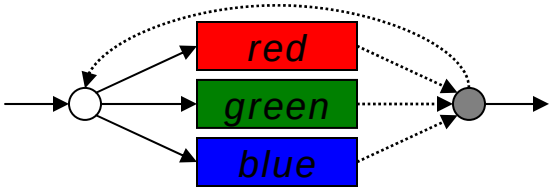




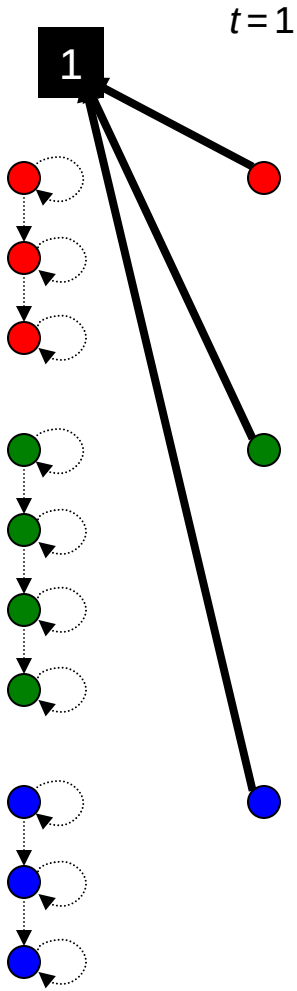
Trellis with Complete Set of Backpointers



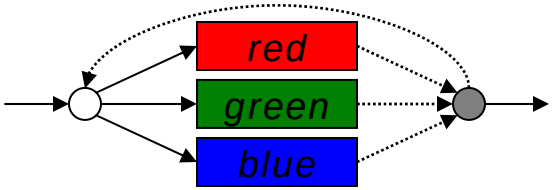
1, t=0, scr1,p=0,...



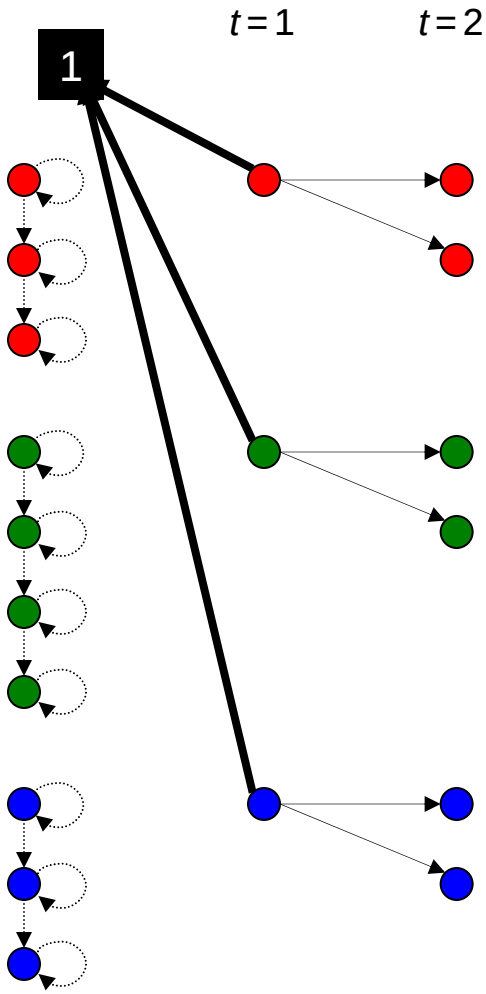
Trellis with Complete Set of Backpointers



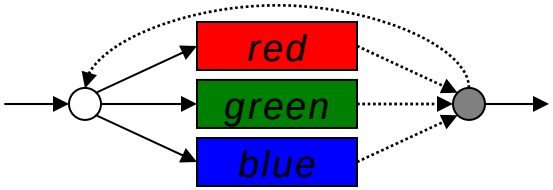
1, t=0, scr1,p=0,...



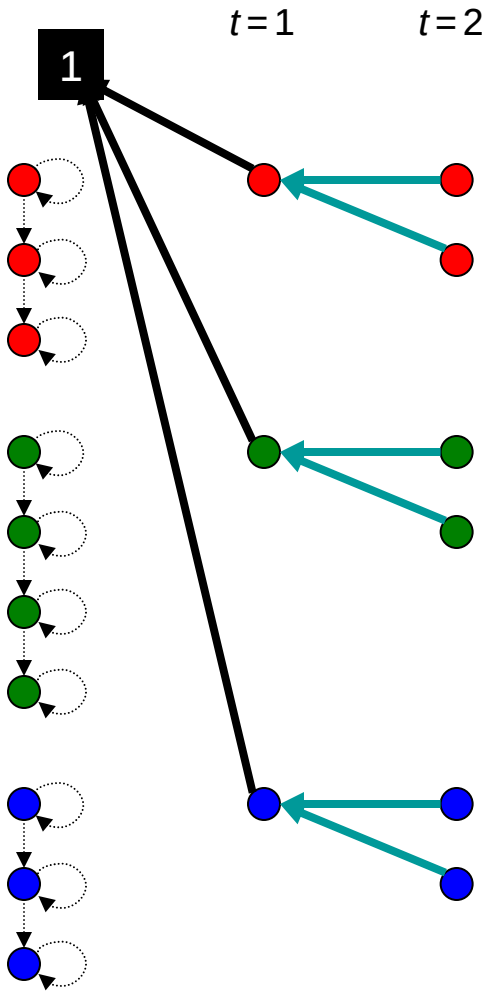
Trellis with Complete Set of Backpointers



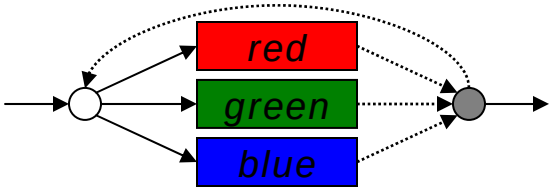
1, t = 0, scr1,p = 0,...



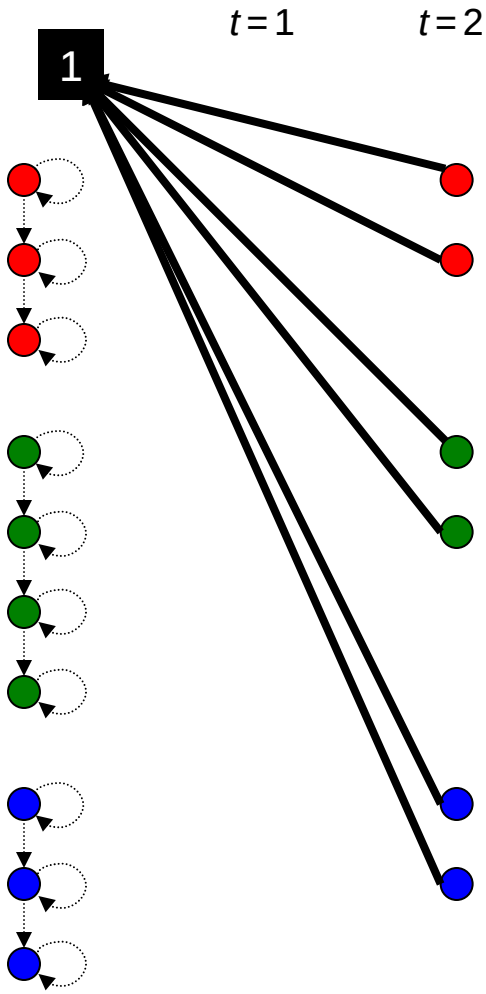
Trellis with Complete Set of Backpointers



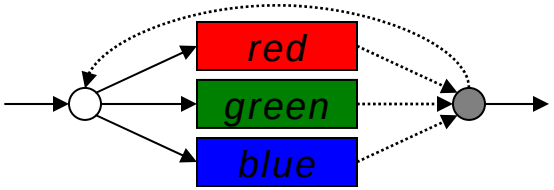
1, t = 0, scr1, p = 0,...



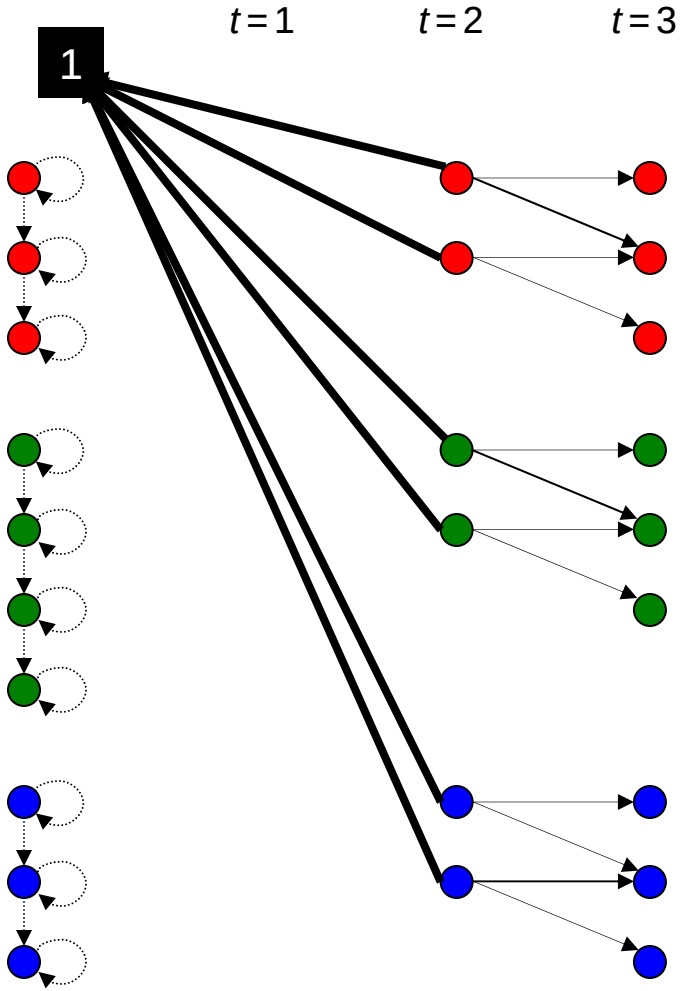
Trellis with Complete Set of Backpointers



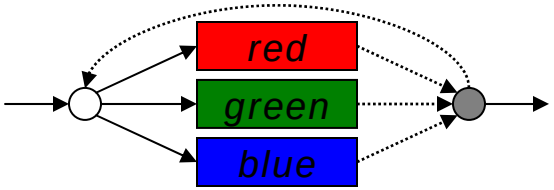
1, t = 0, scr1, p = 0,...



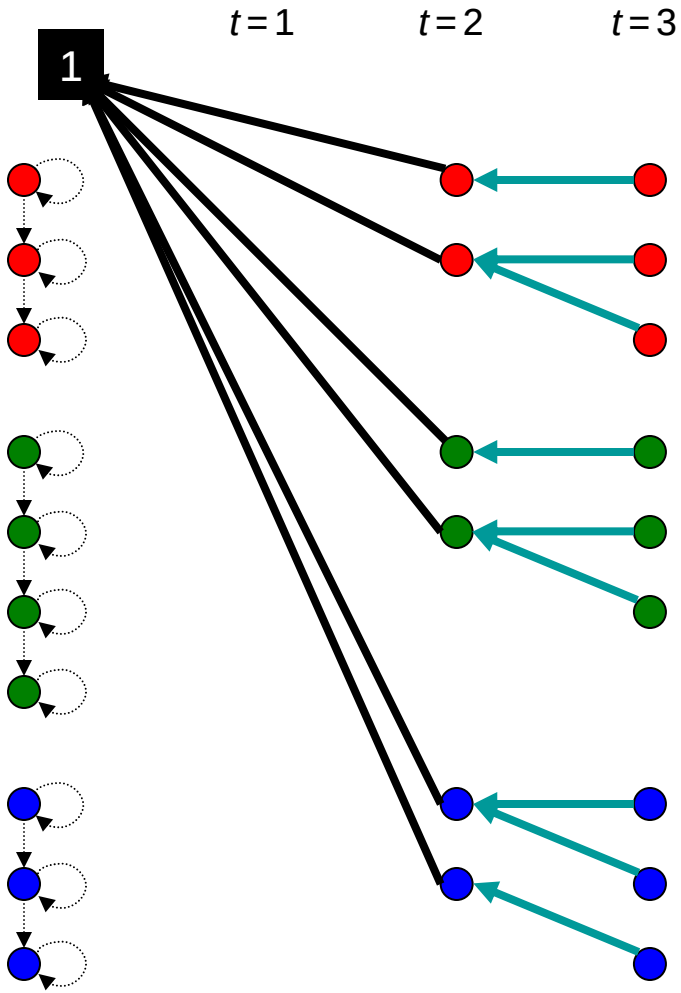
Trellis with Complete Set of Backpointers



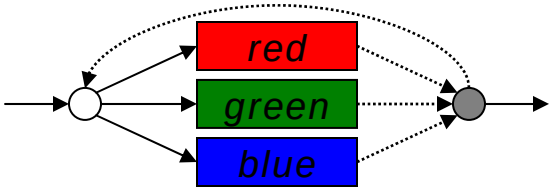
1, t=0, scr1,p=0,...



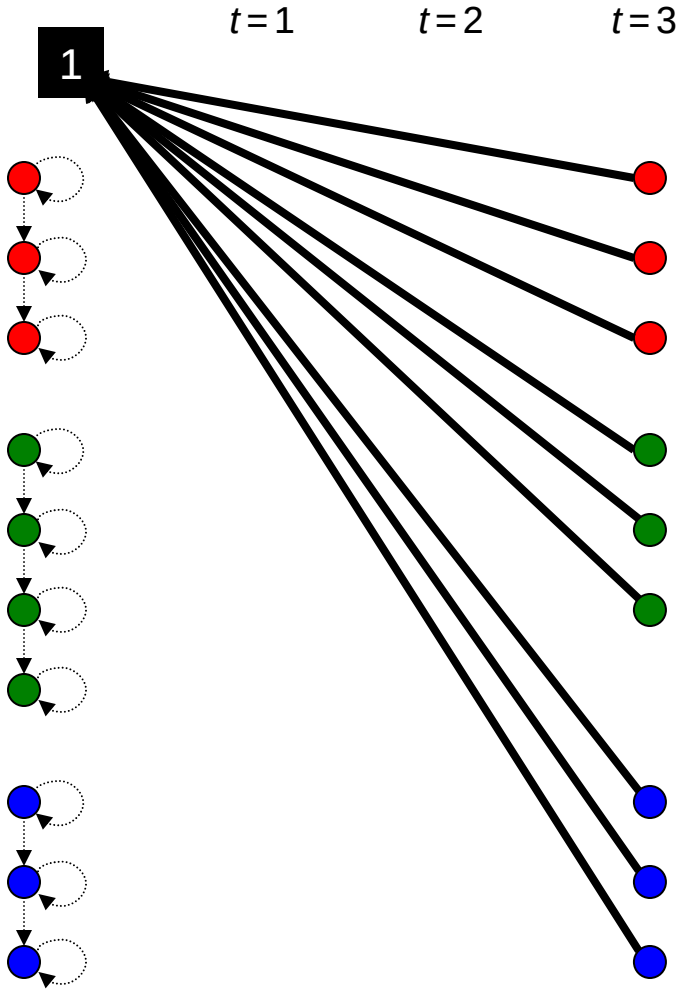
Trellis with Complete Set of Backpointers



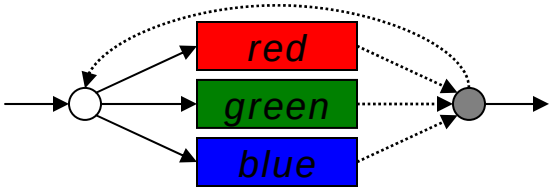
1, t=0, scr1,p=0,...



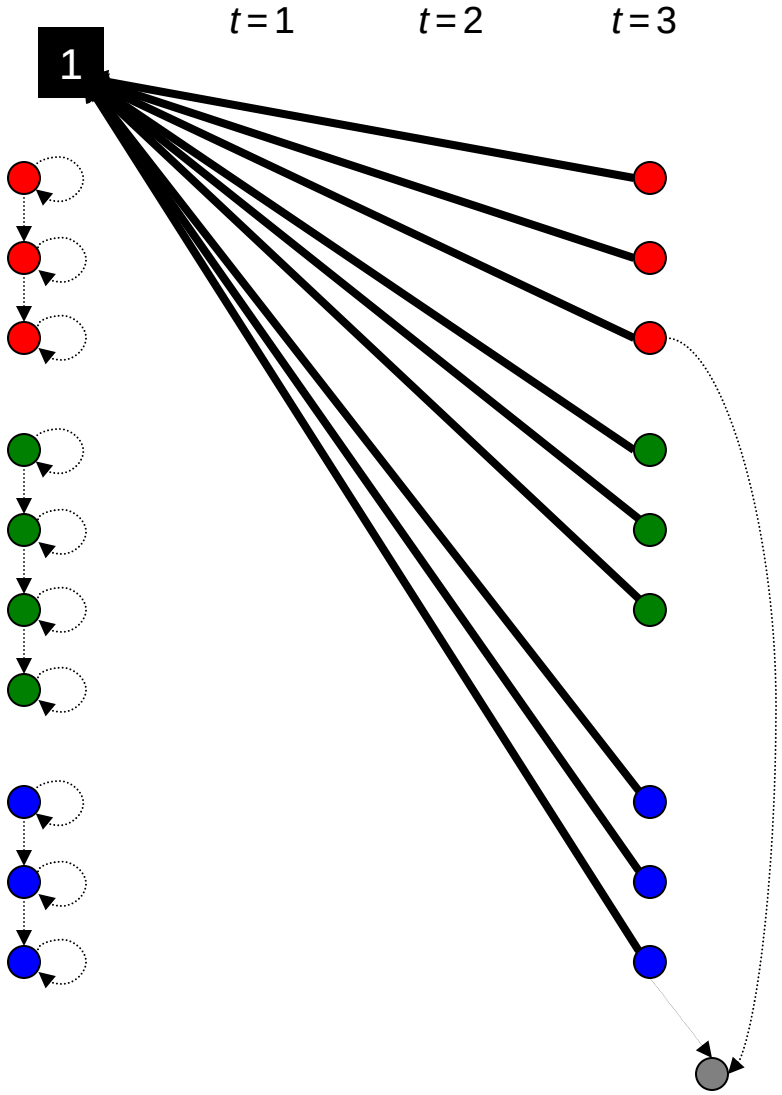
Trellis with Complete Set of Backpointers



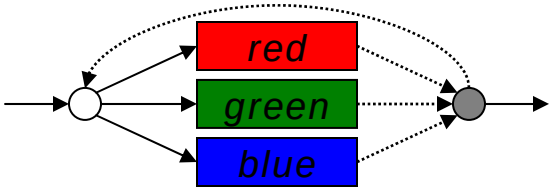
1, t=0, scr1,p=0,...



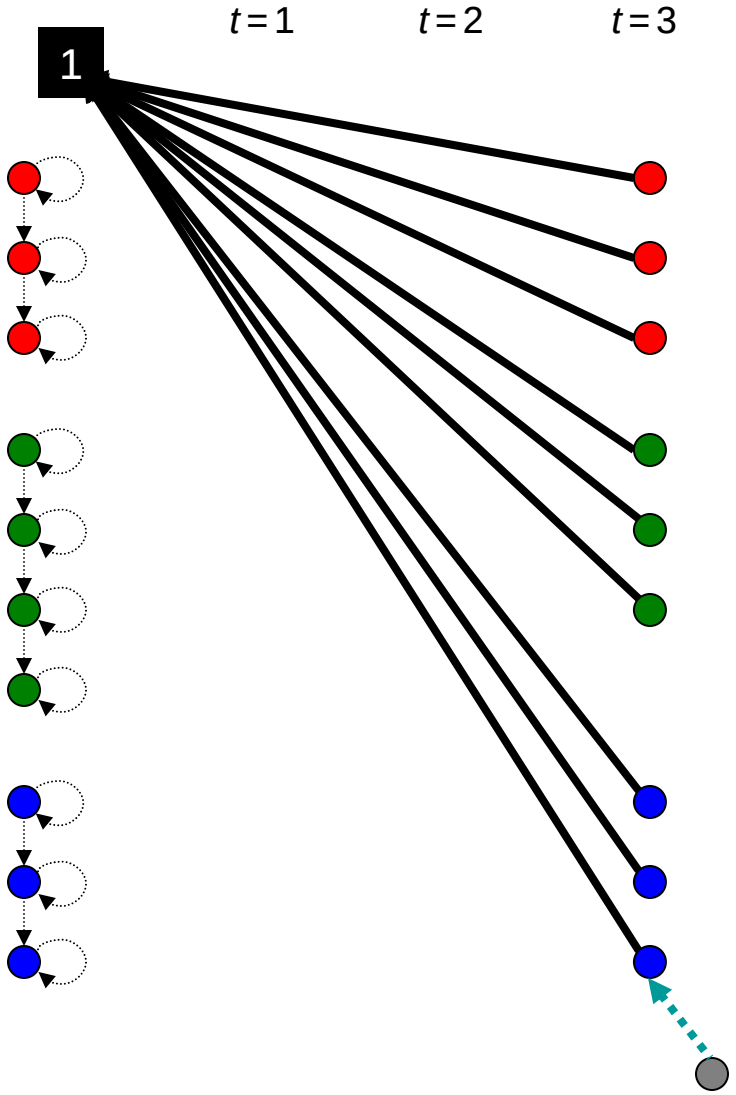
Trellis with Complete Set of Backpointers



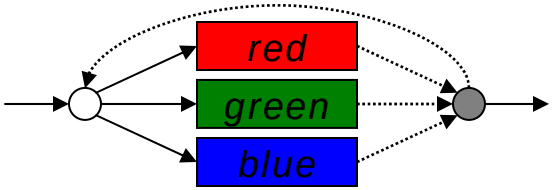
1, t=0, scr1,p=0,...



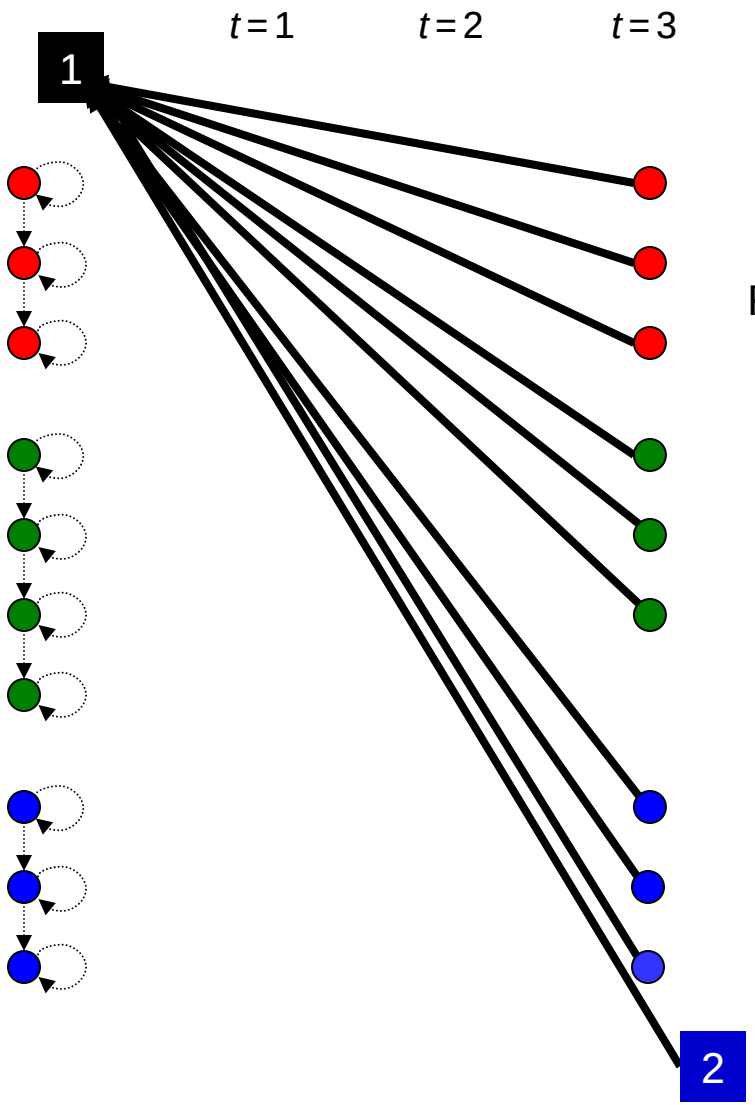
Trellis with Complete Set of Backpointers



1, t=0, scr1,p=0,...



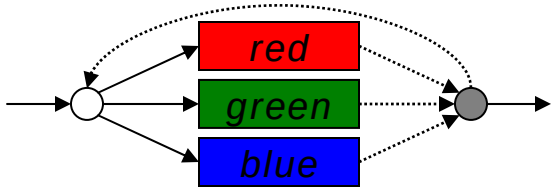
Trellis with Complete Set of Backpointers



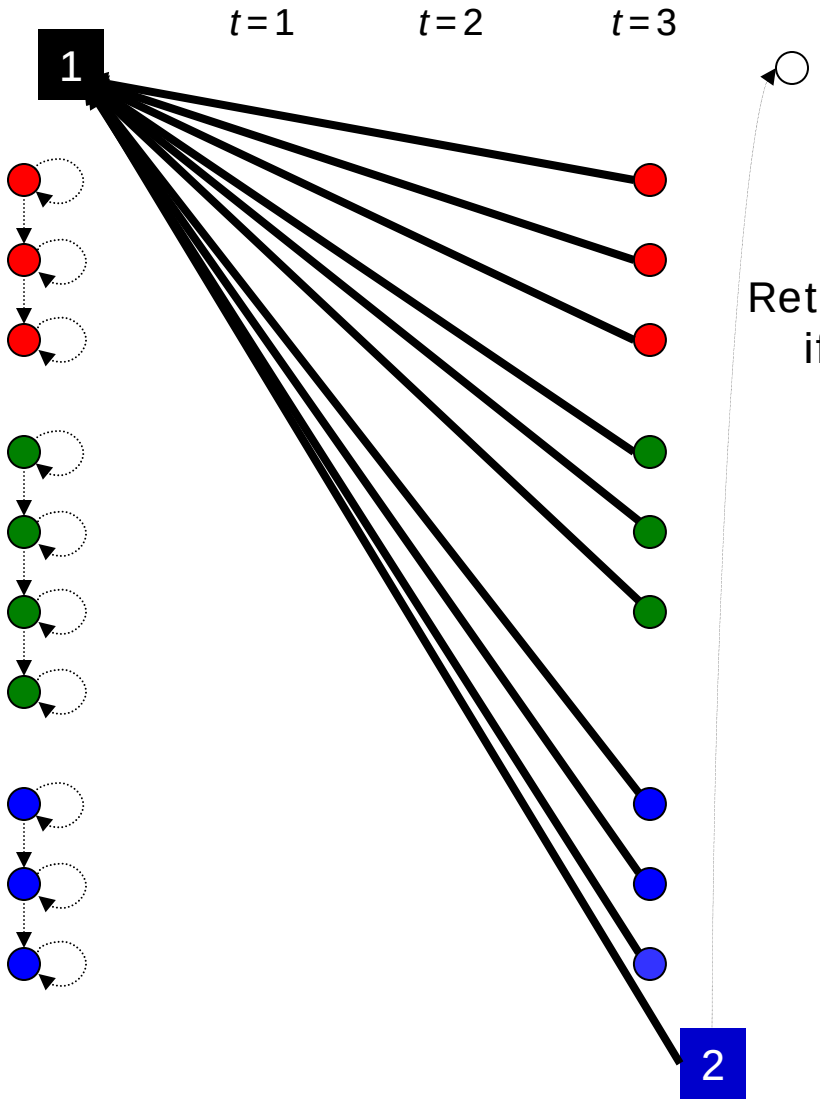
1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

Retain backpointers (and add the to the table) if deleting them will result in loss of word history



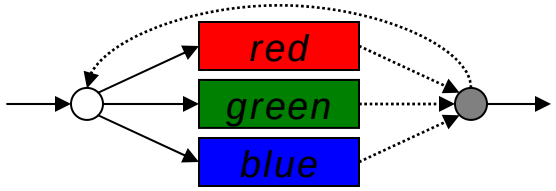
Trellis with Complete Set of Backpointers



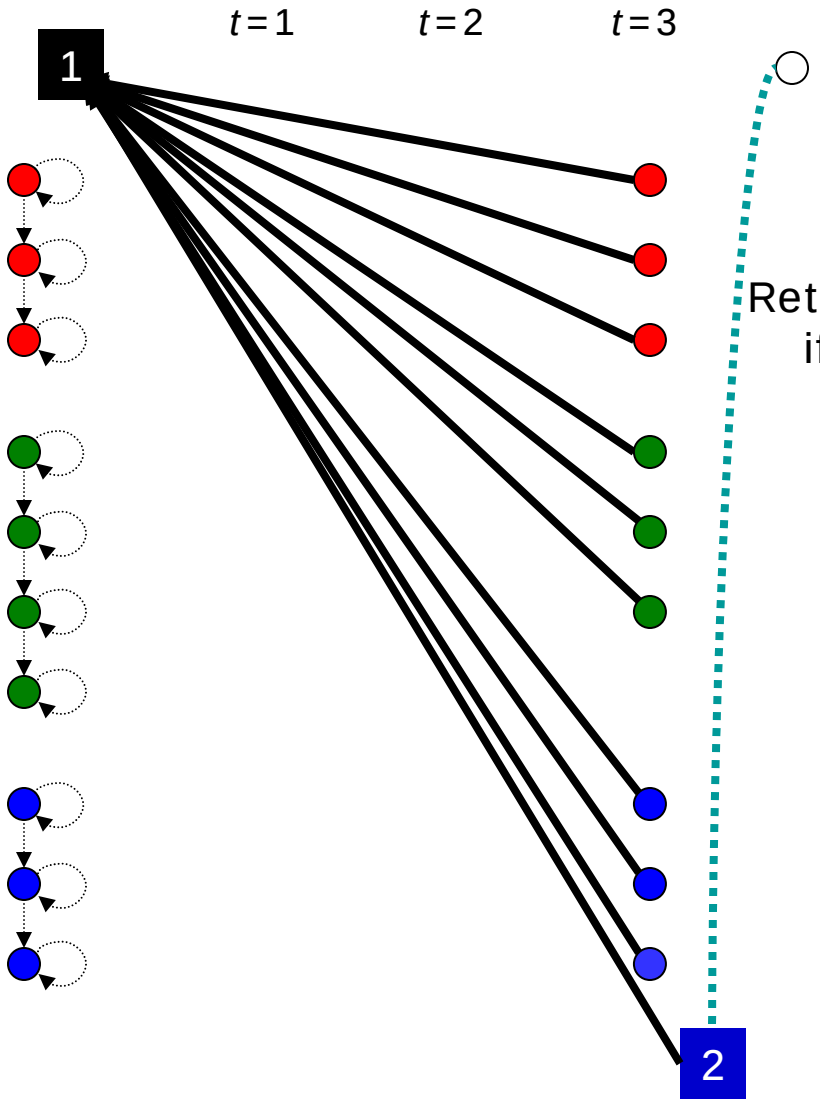
1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

Retain backpointers (and add the to the table) if deleting them will result in loss of word history



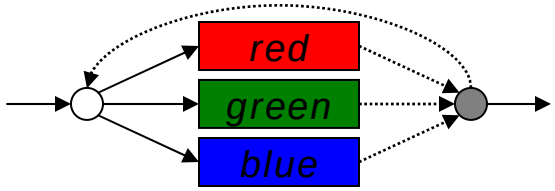
Trellis with Complete Set of Backpointers



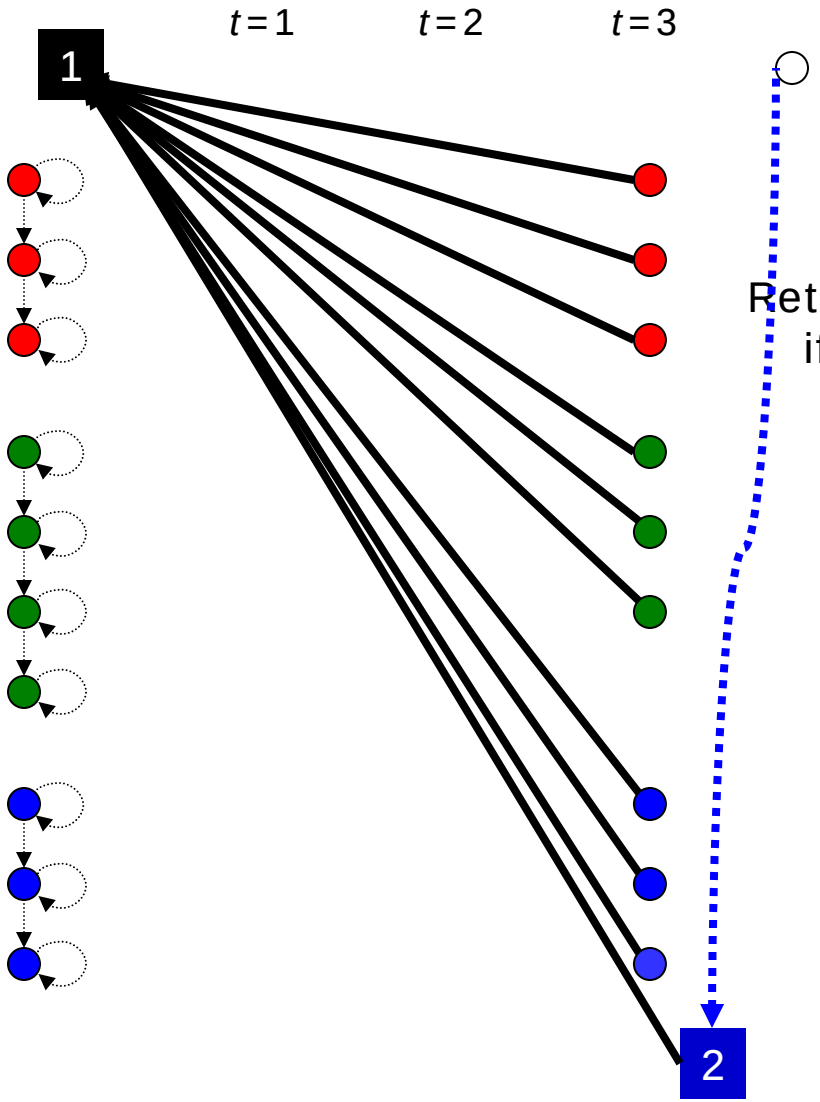
1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

Retain backpointers (and add the to the table) if deleting them will result in loss of word history



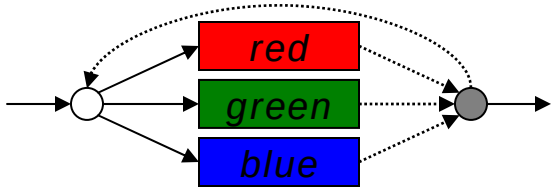
Trellis with Complete Set of Backpointers



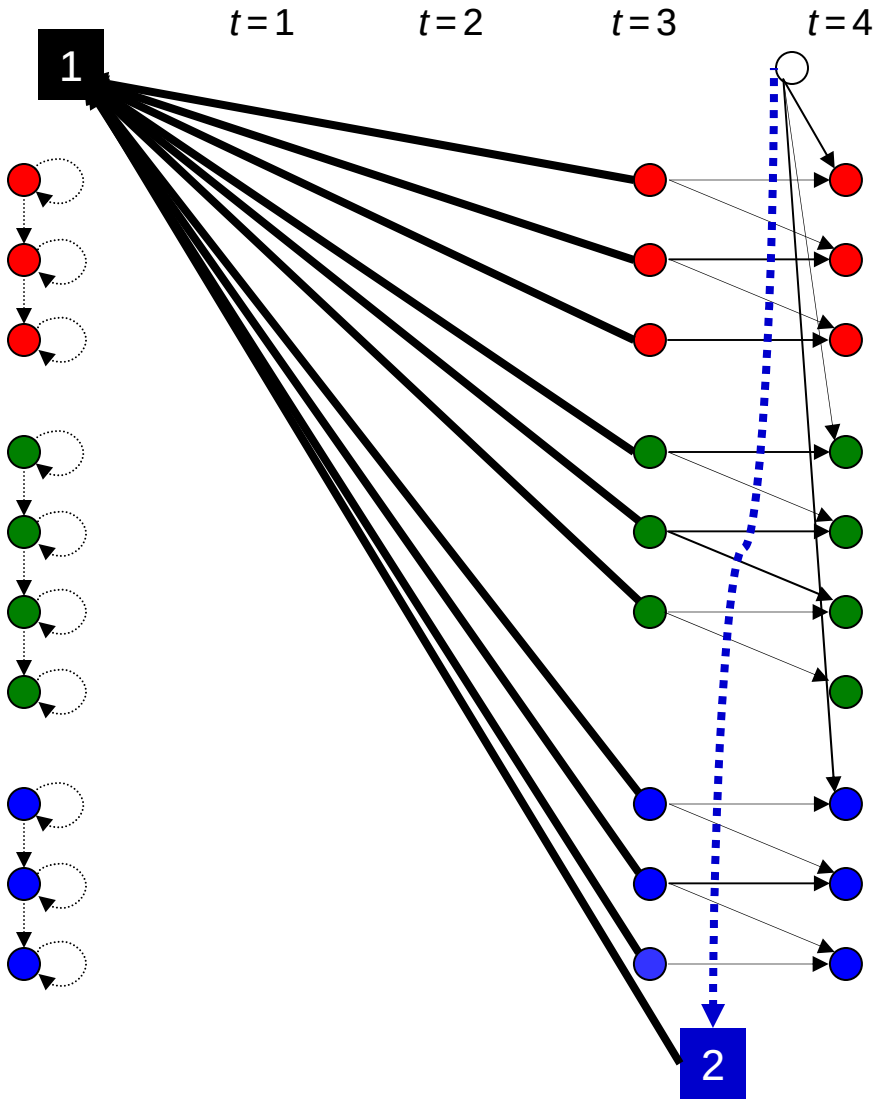
1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

Retain backpointers (and add the to the table) if deleting them will result in loss of word history

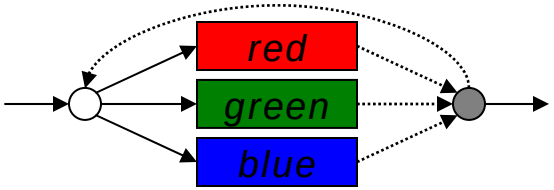


Trellis with Complete Set of Backpointers

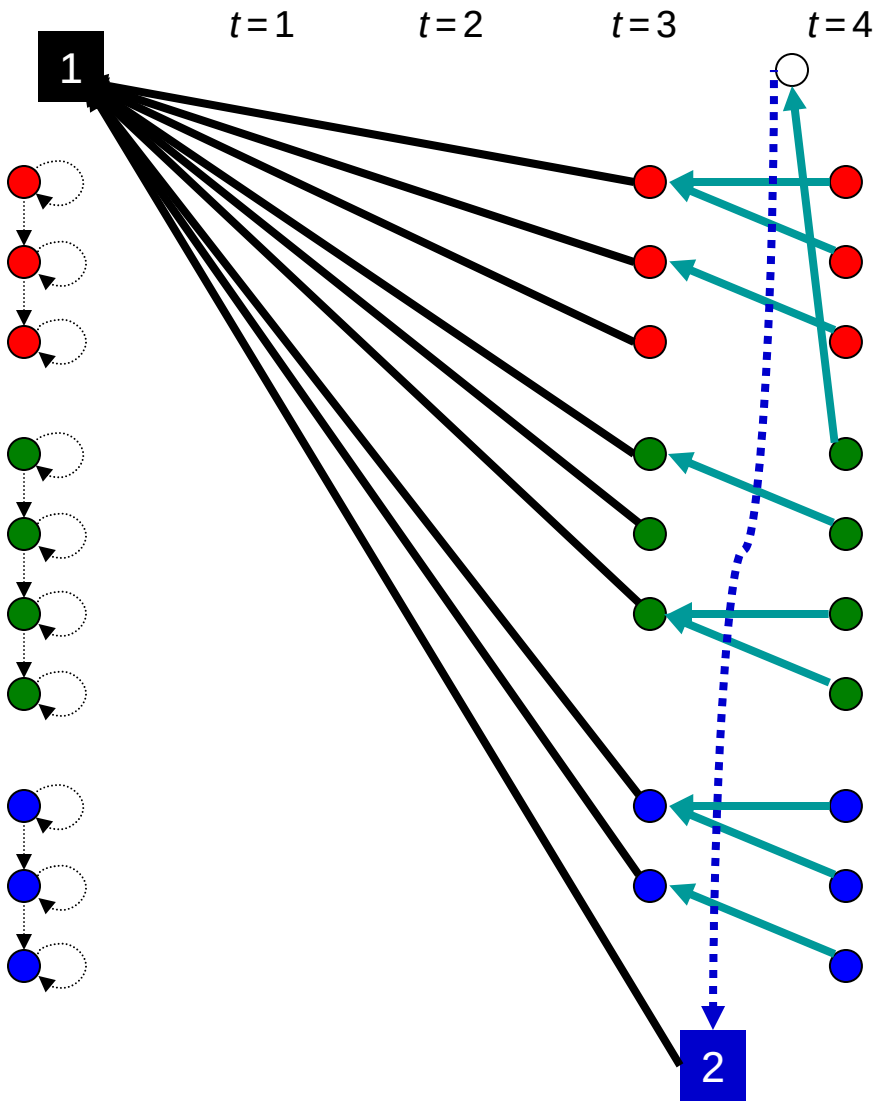


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

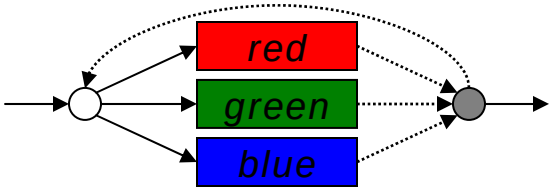


Trellis with Complete Set of Backpointers

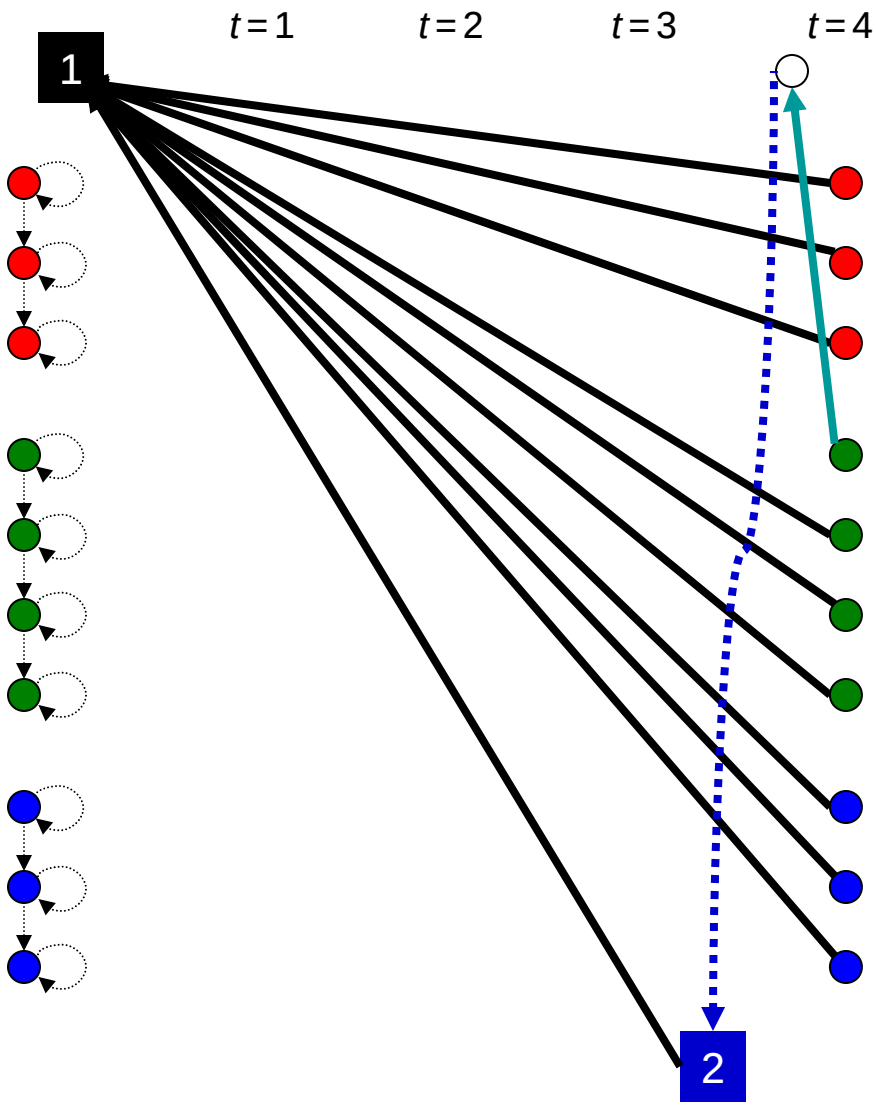


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

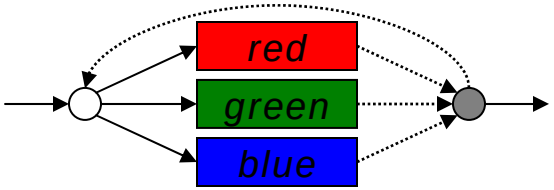


Trellis with Complete Set of Backpointers

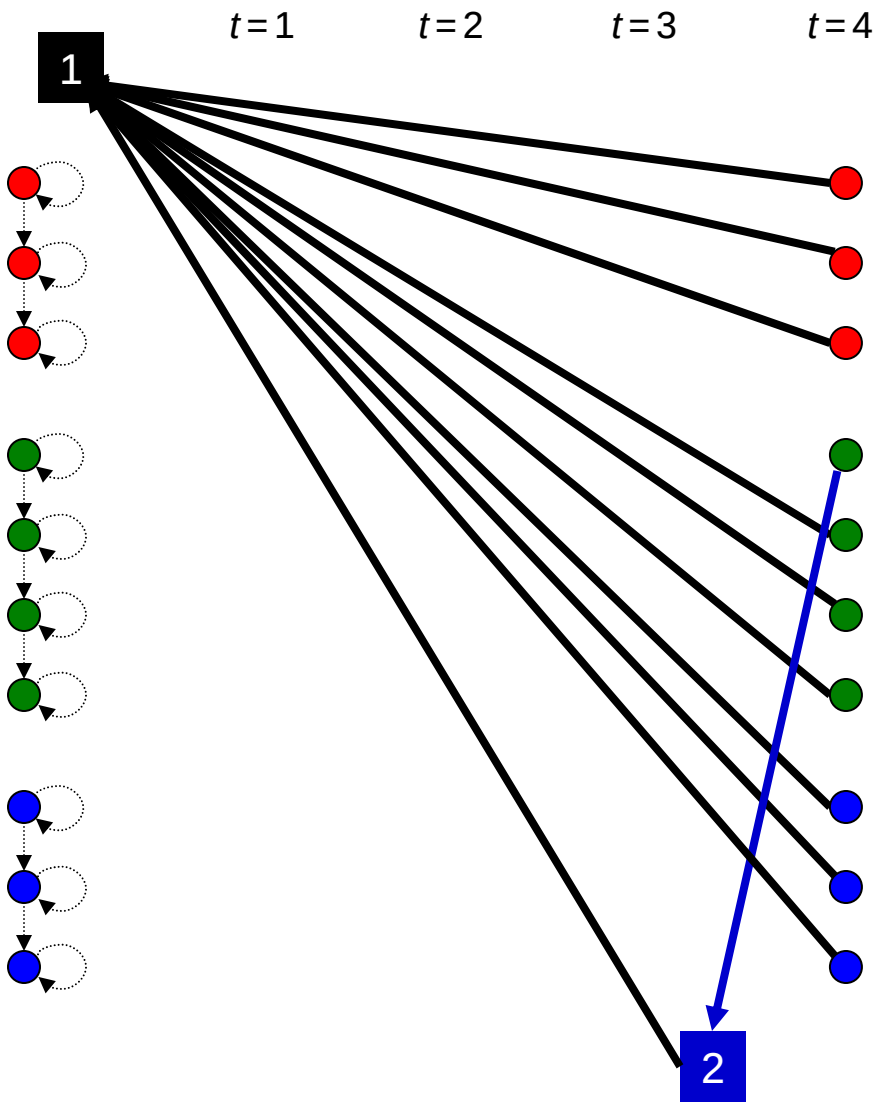


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

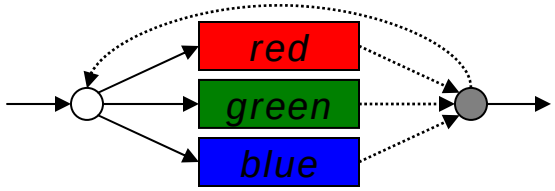


Trellis with Complete Set of Backpointers

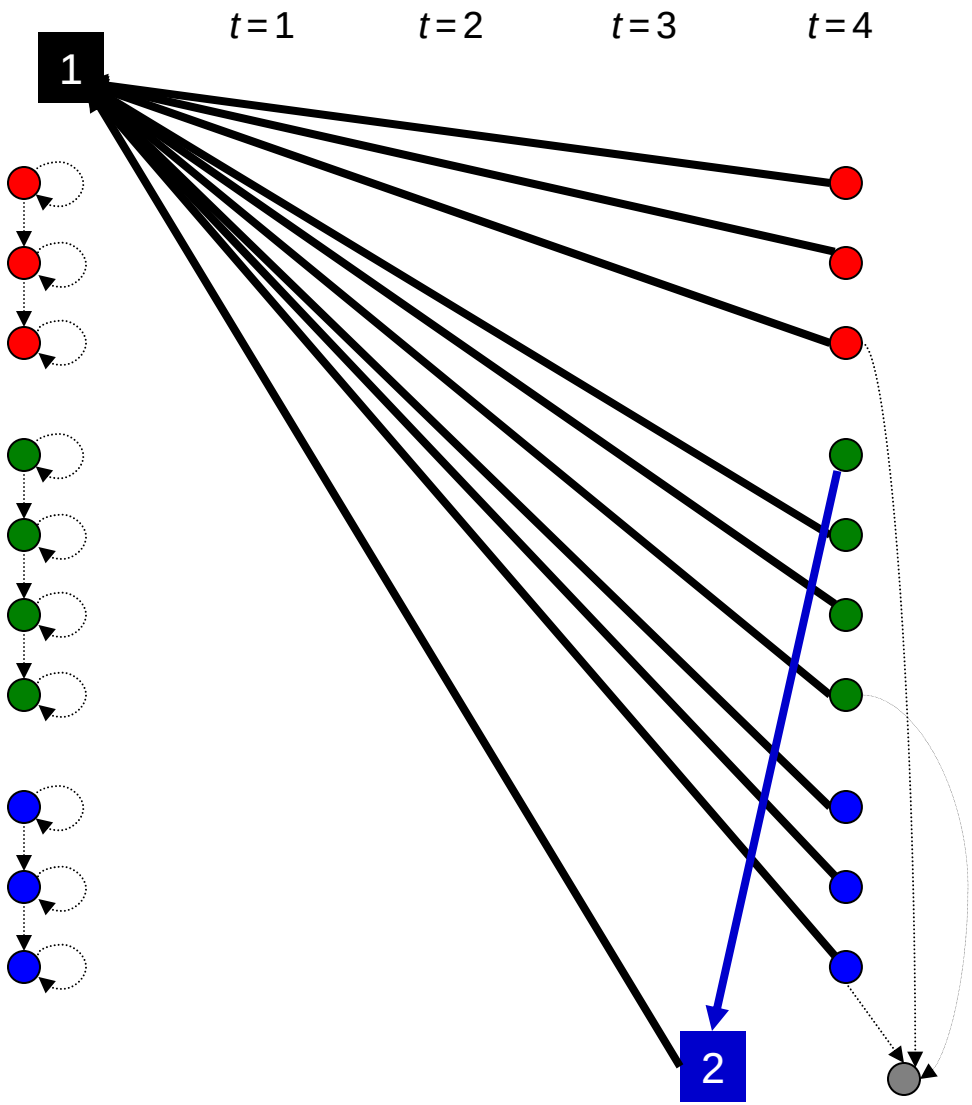


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

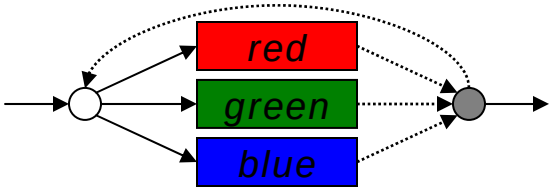


Trellis with Complete Set of Backpointers

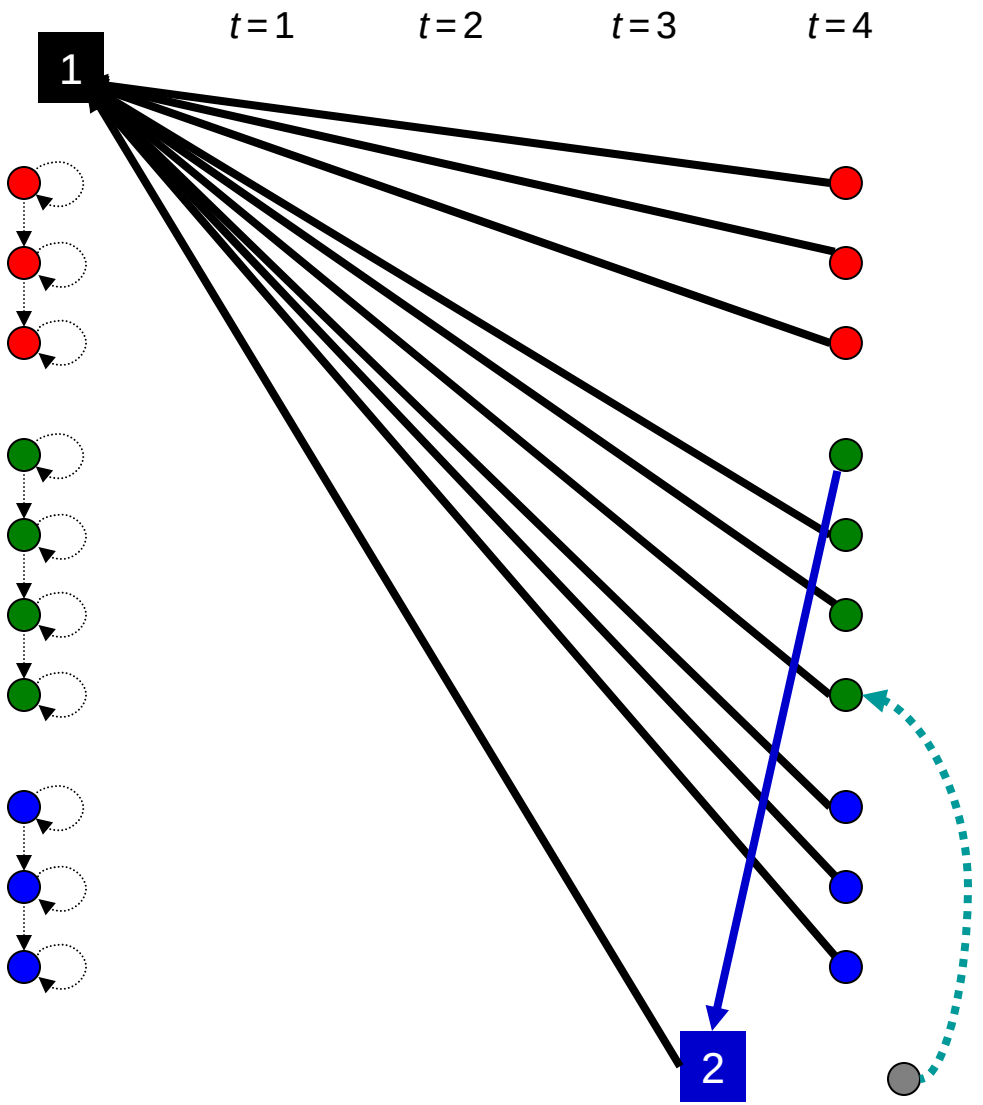


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

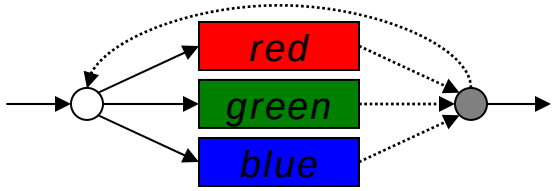


Trellis with Complete Set of Backpointers

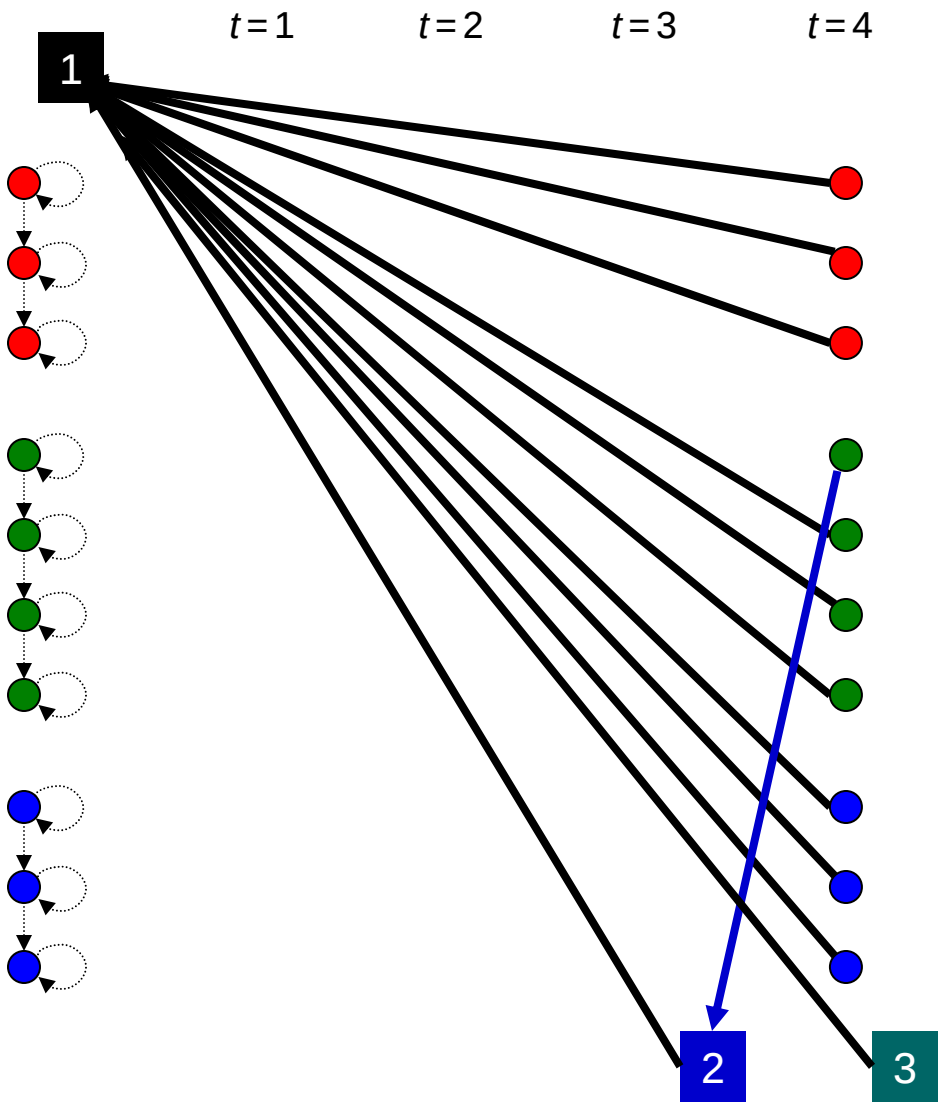


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...



Trellis with Complete Set of Backpointers



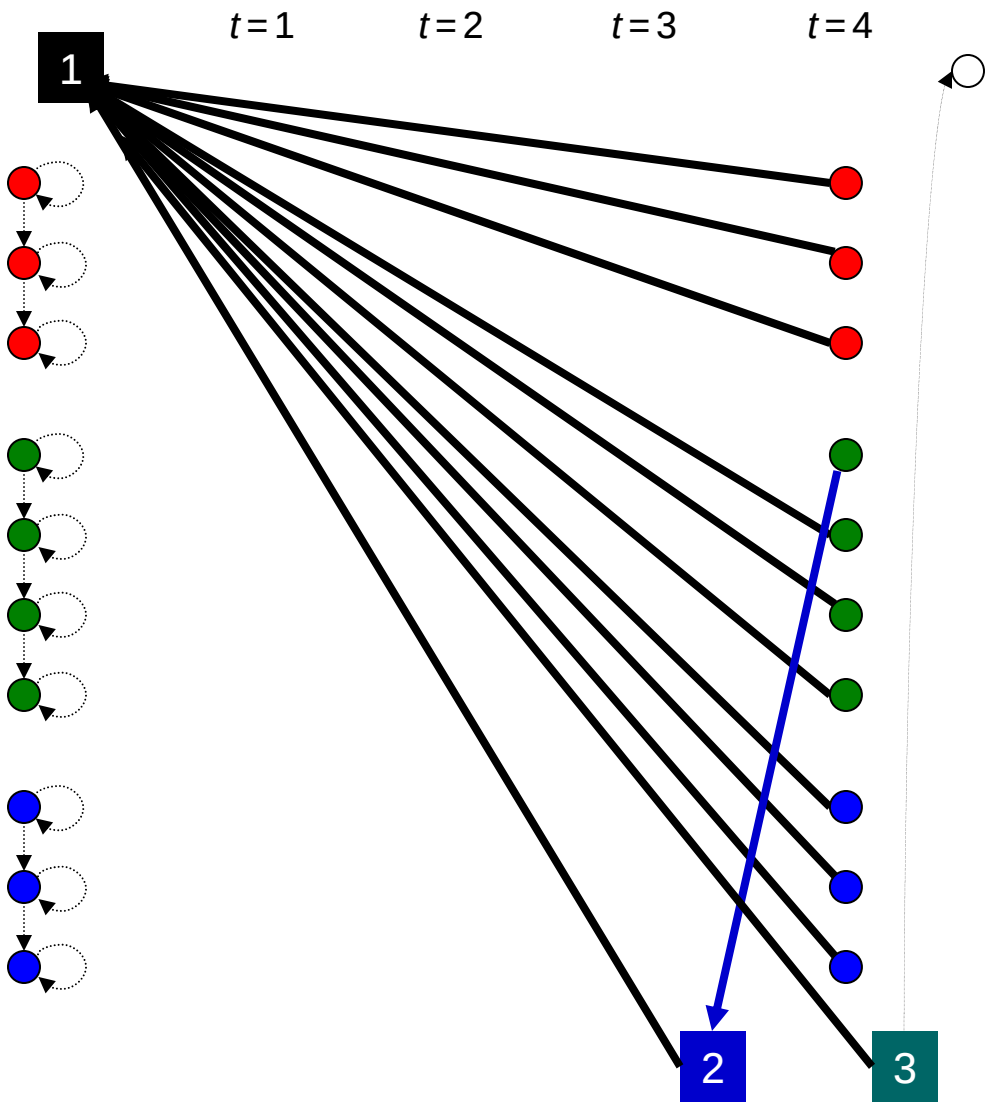
1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, gr

Retain backpointers (and add the to the table)
if deleting them will result in loss of word history

Trellis with Complete Set of Backpointers

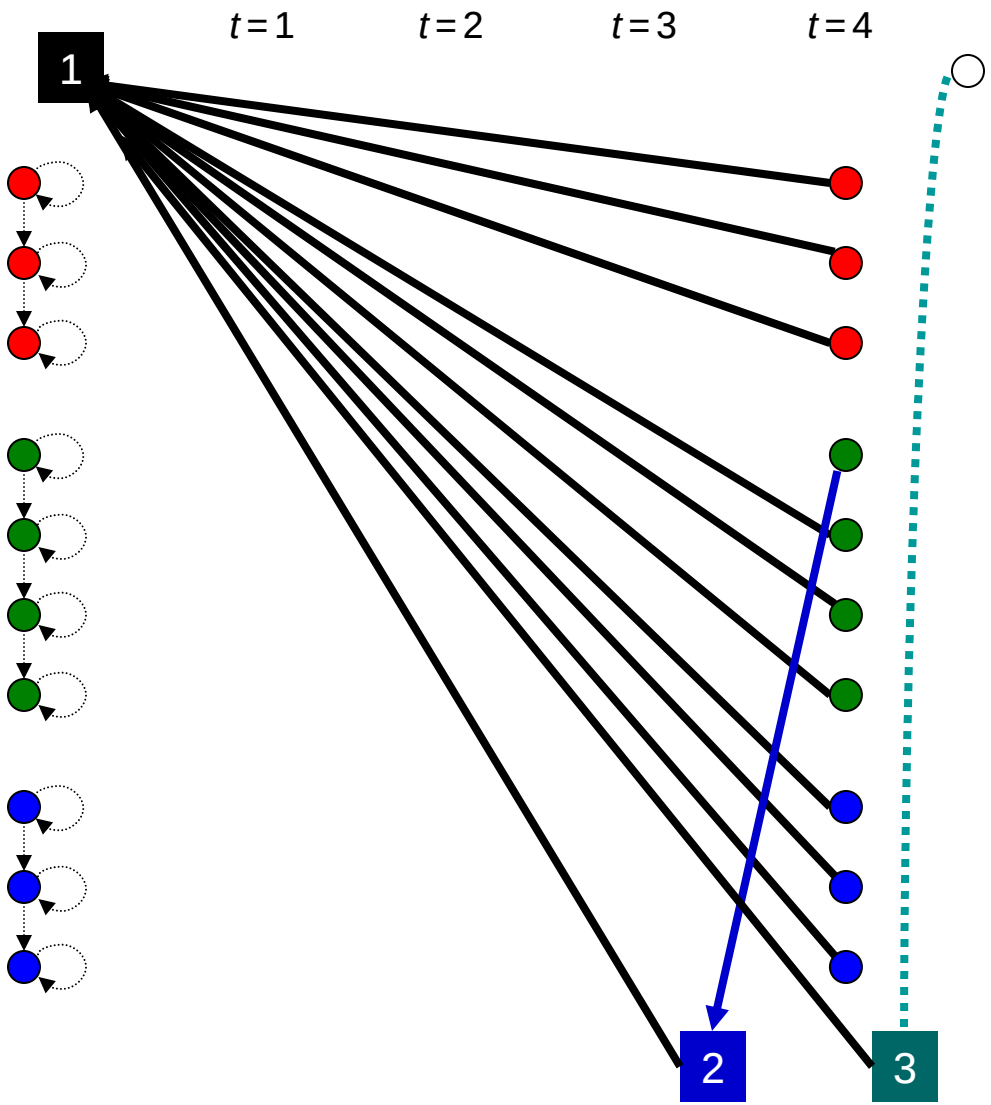


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

Trellis with Complete Set of Backpointers

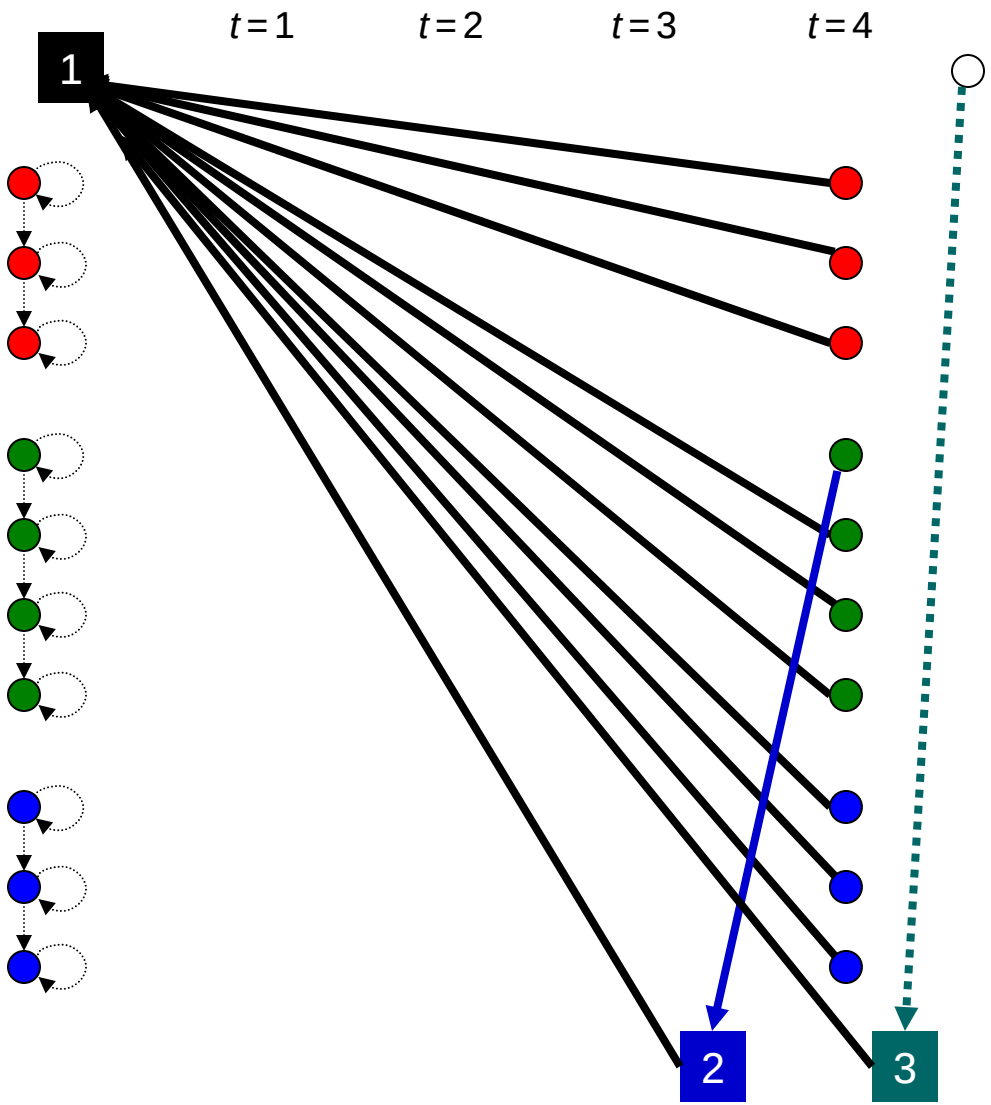


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

Trellis with Complete Set of Backpointers

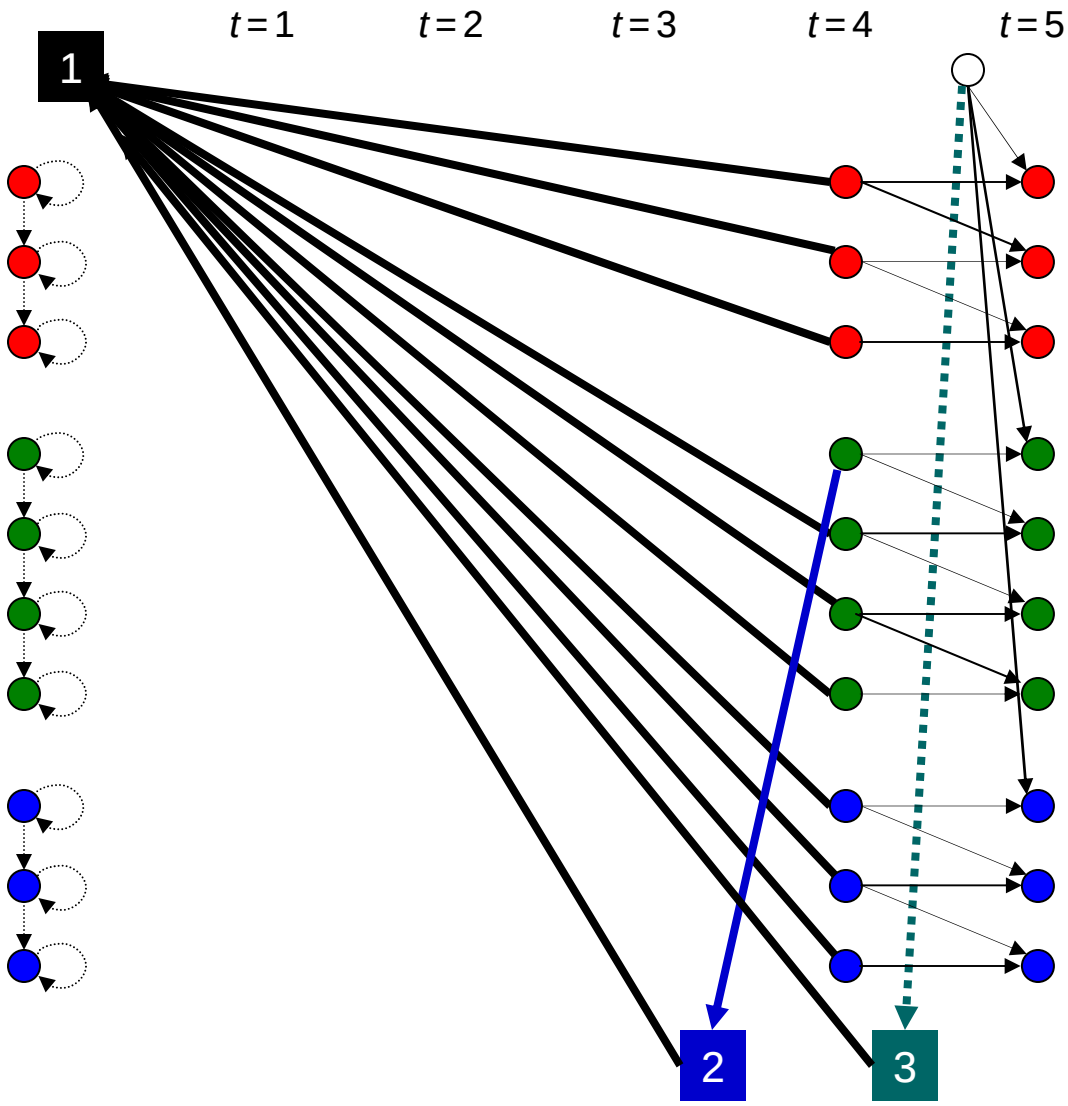


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

Trellis with Complete Set of Backpointers

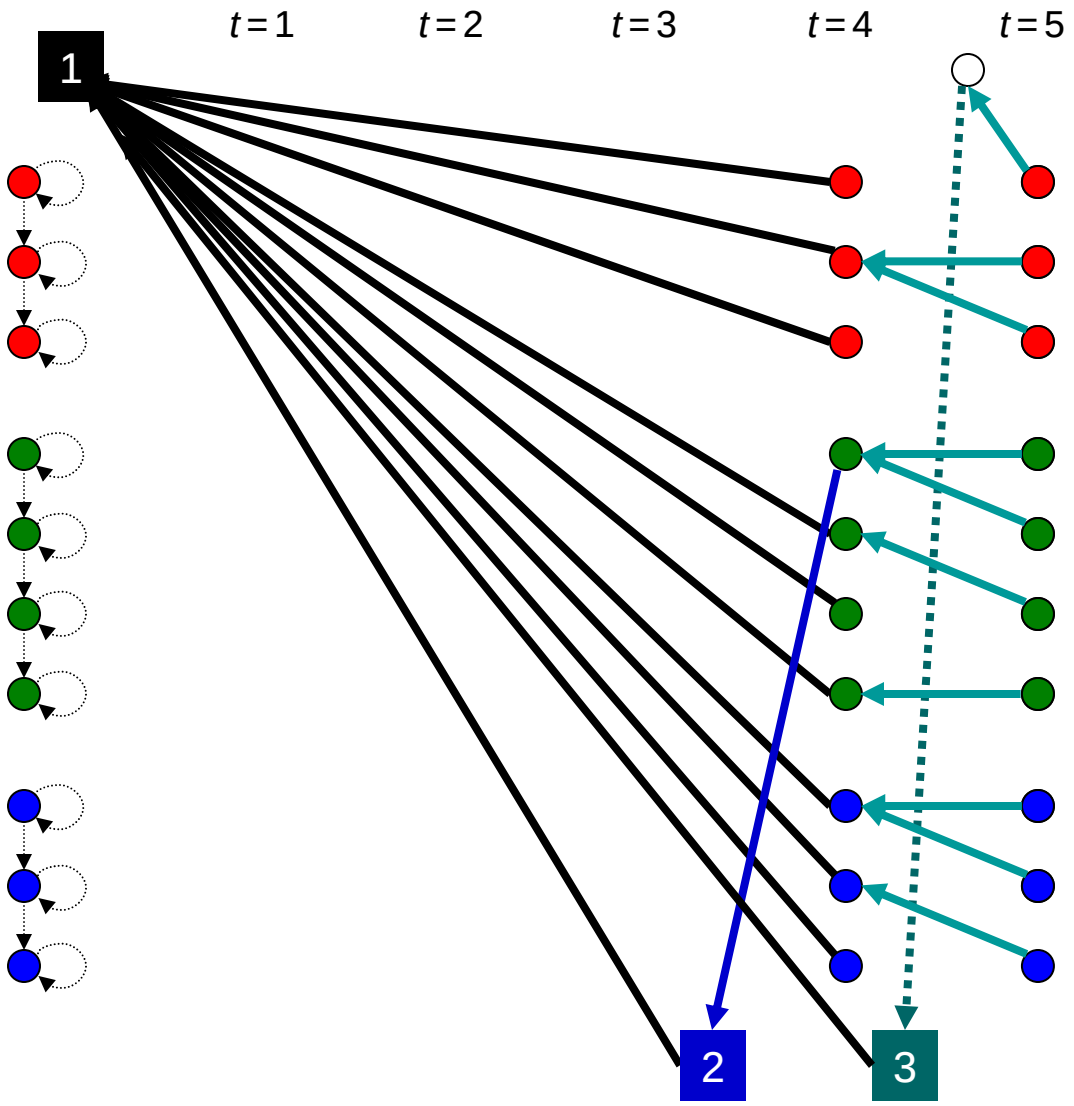


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

Trellis with Complete Set of Backpointers

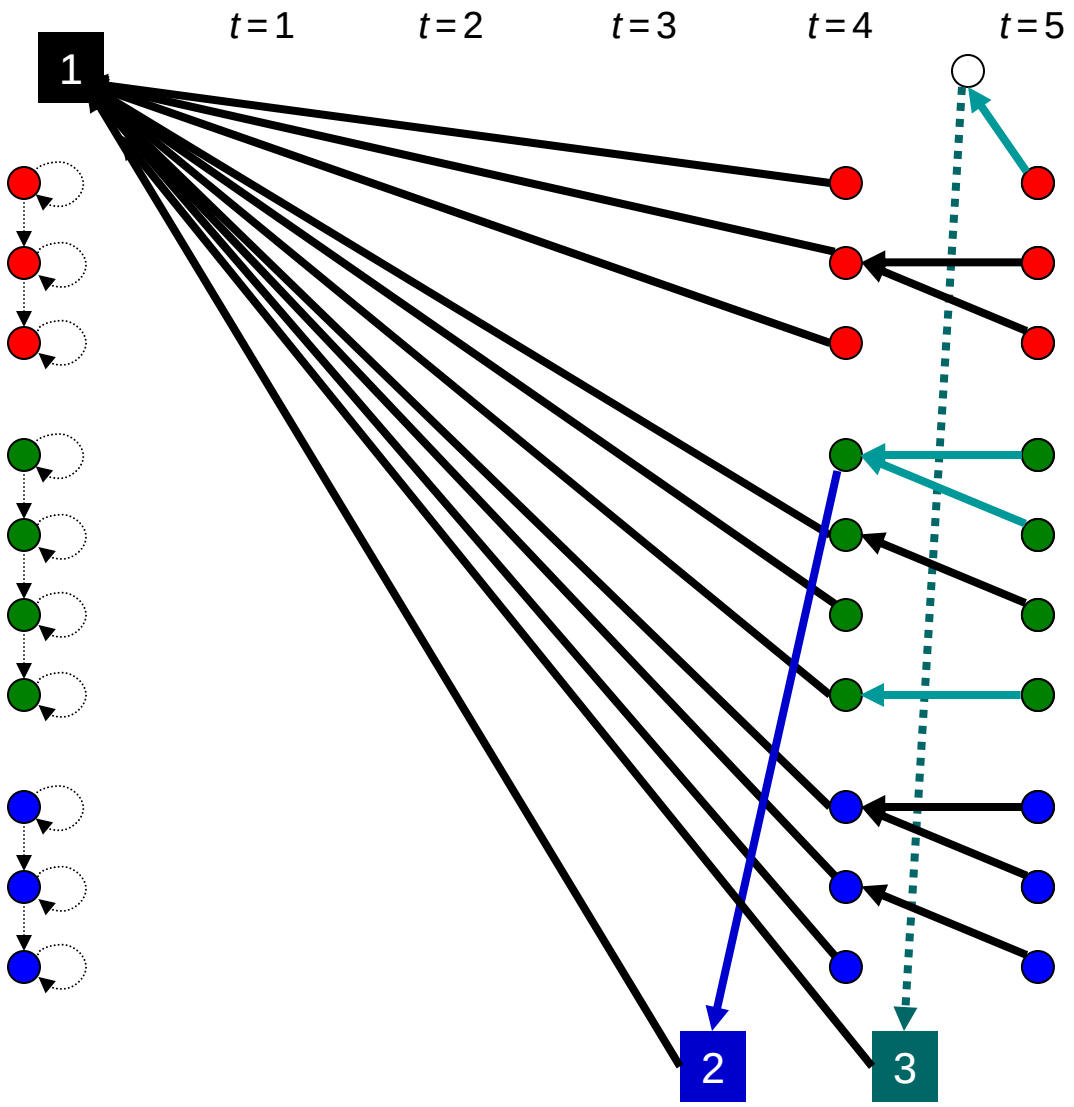


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

Trellis with Complete Set of Backpointers

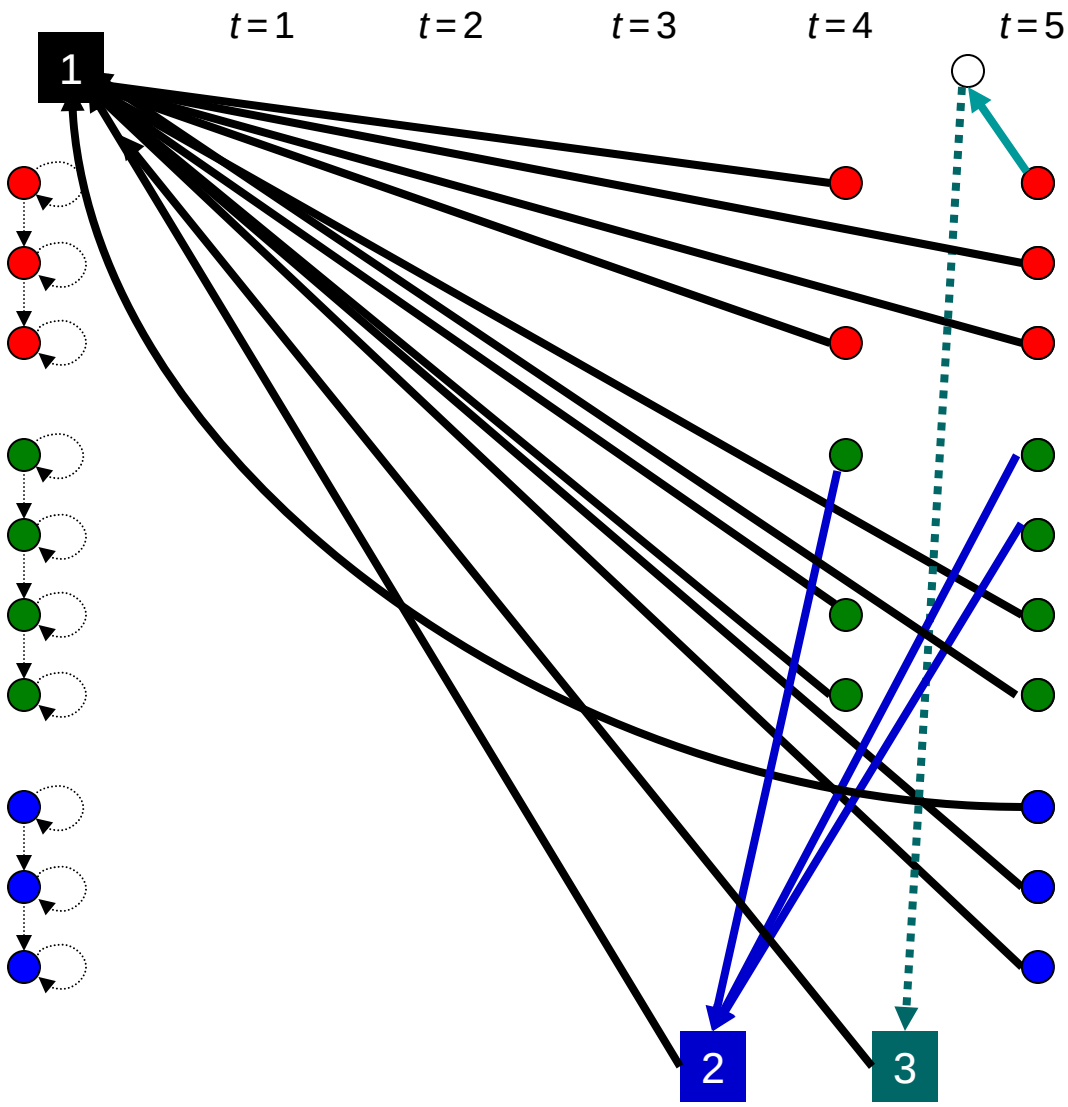


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

Trellis with Complete Set of Backpointers

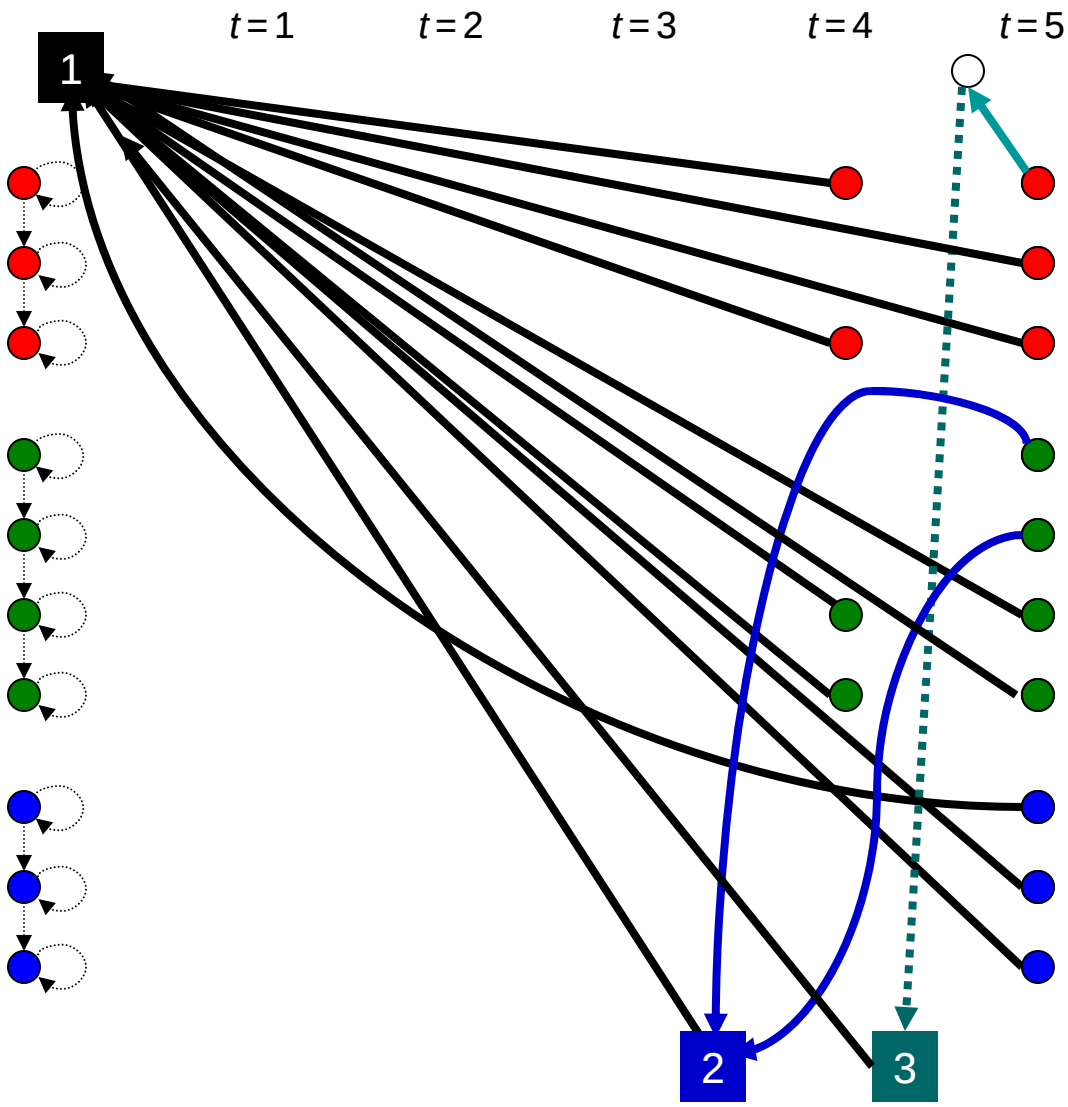


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

Trellis with Complete Set of Backpointers

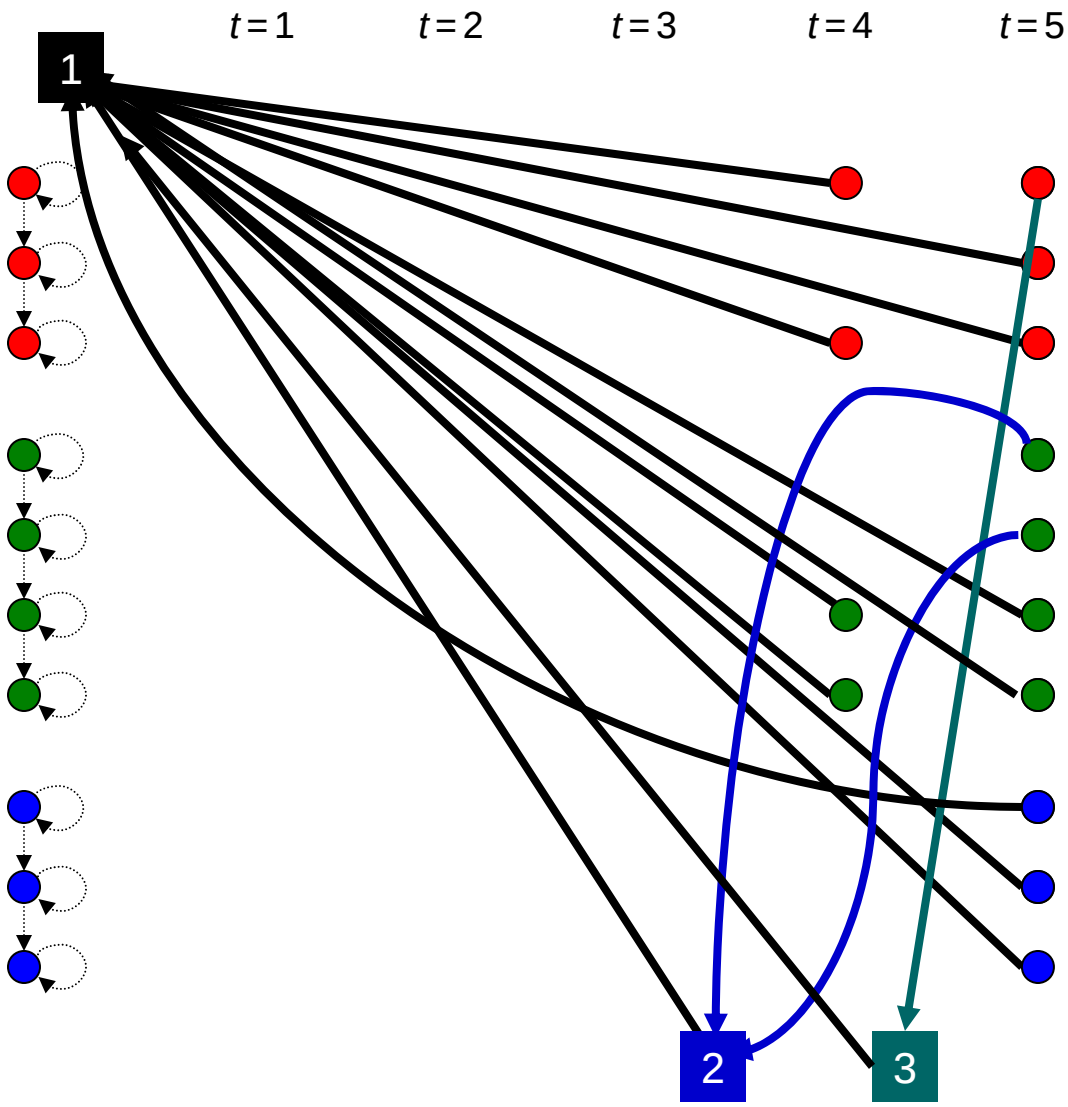


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

Trellis with Complete Set of Backpointers

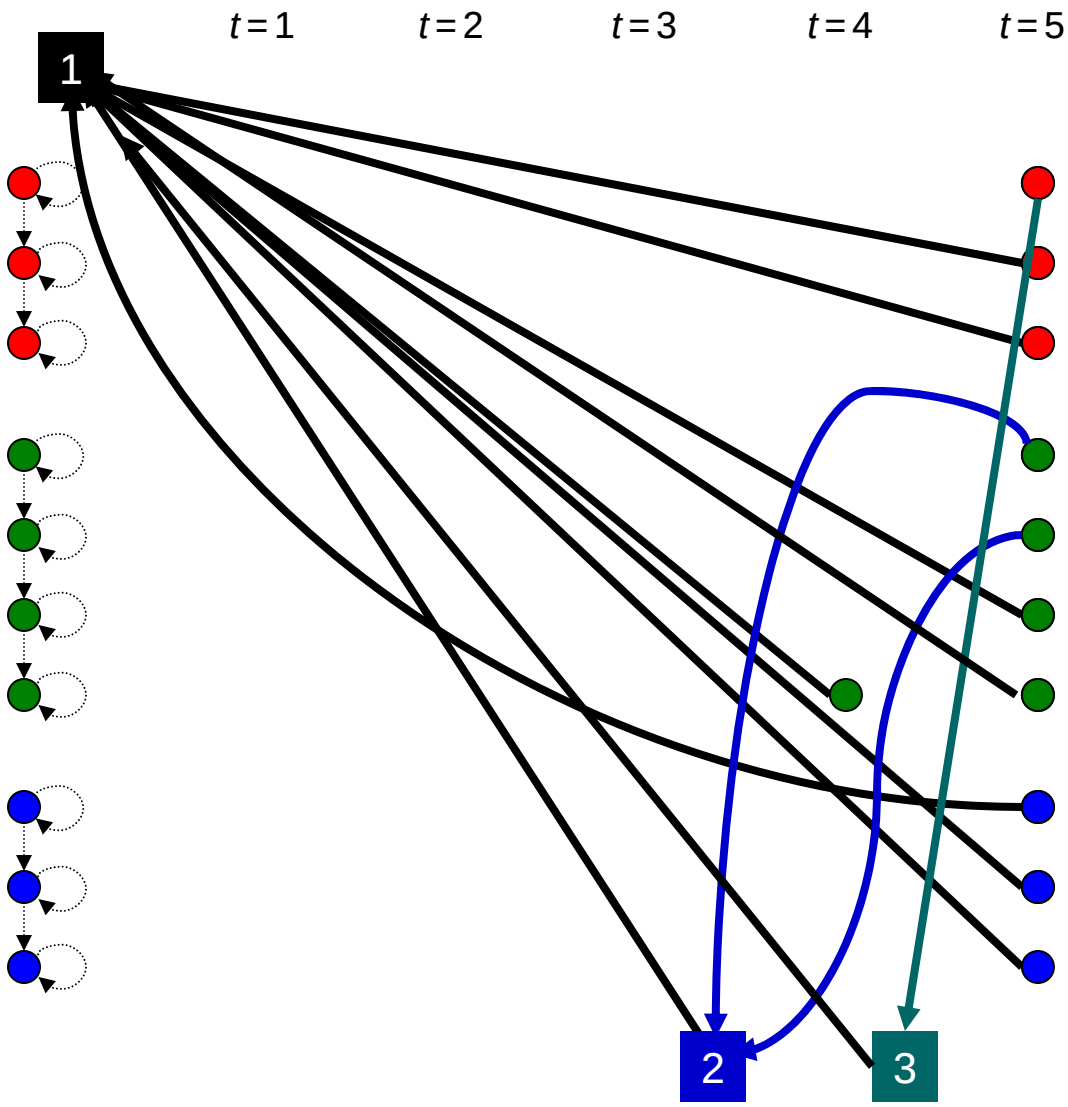


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

Trellis with Complete Set of Backpointers

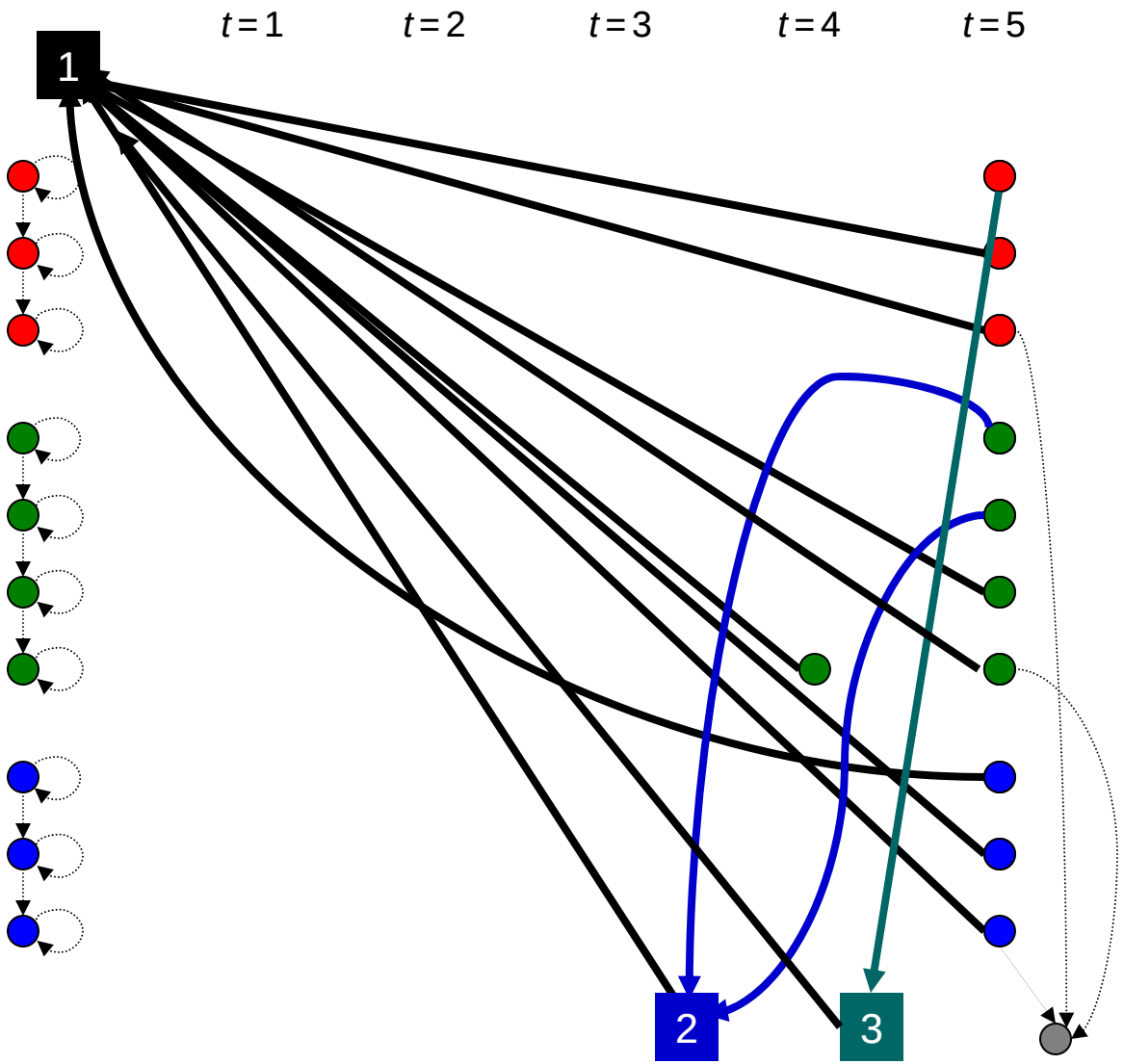


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

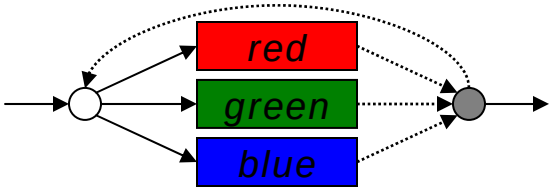
Trellis with Complete Set of Backpointers



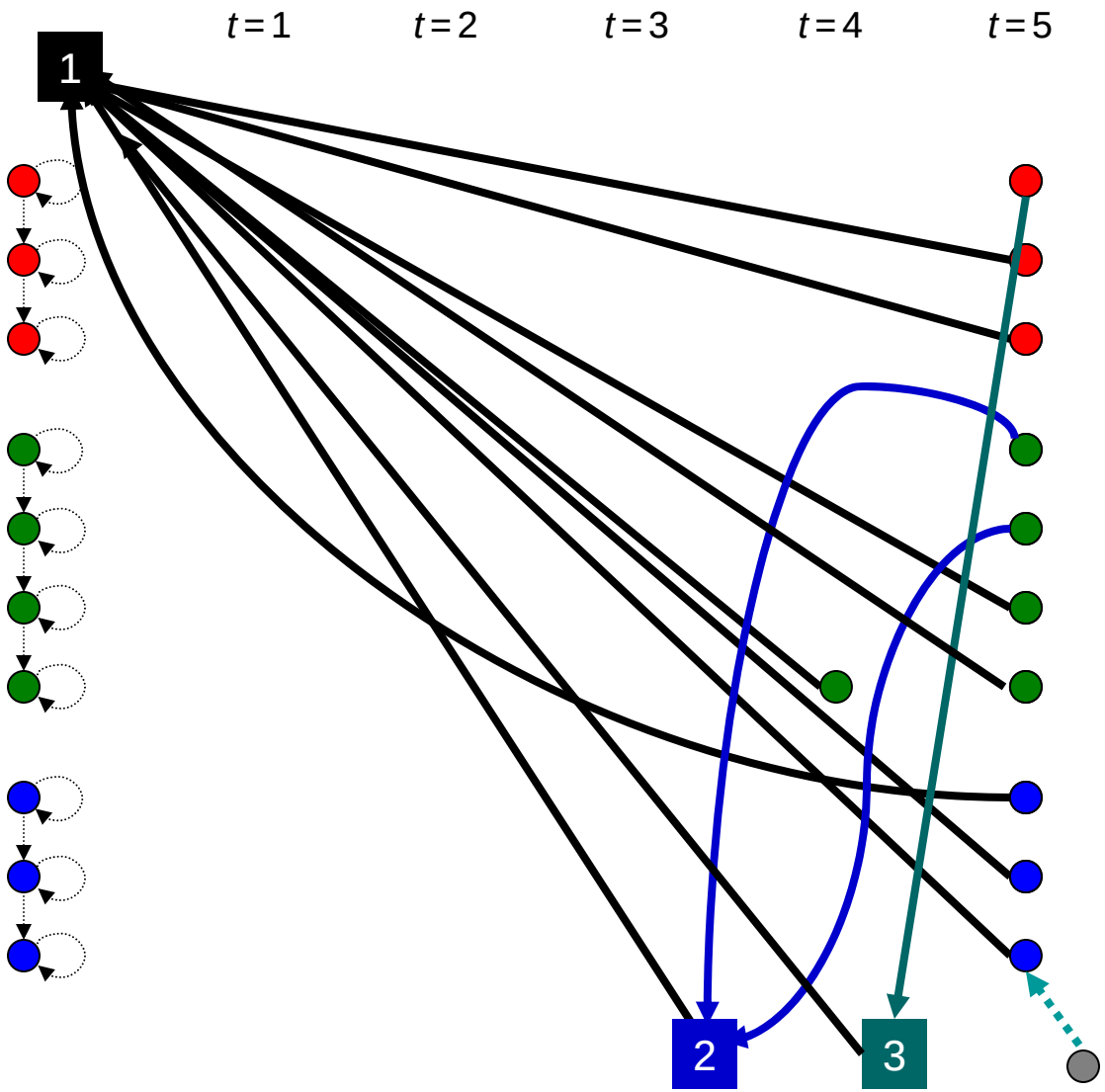
1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br



Trellis with Complete Set of Backpointers

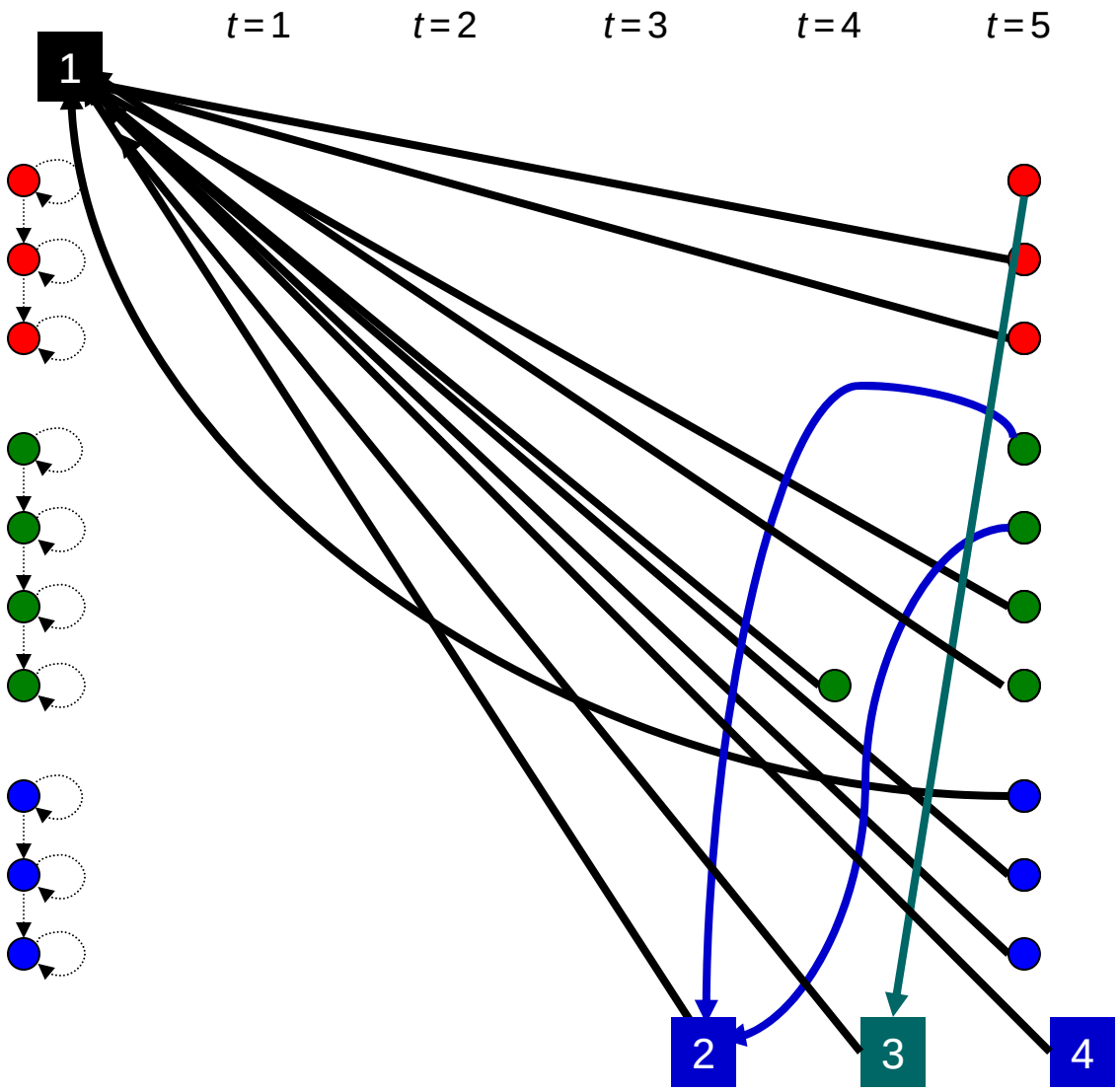


1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, br

Trellis with Complete Set of Backpointers



1, t=0, scr1,p=0,<s>...

2, t=3, scr2,p=1,bl...

3, t=4, scr3,p=1, gr

4, t=5, scr4,p=1,bl

Retain backpointers (and add the to the table) if deleting them will result in loss of word history

Using Backpointers

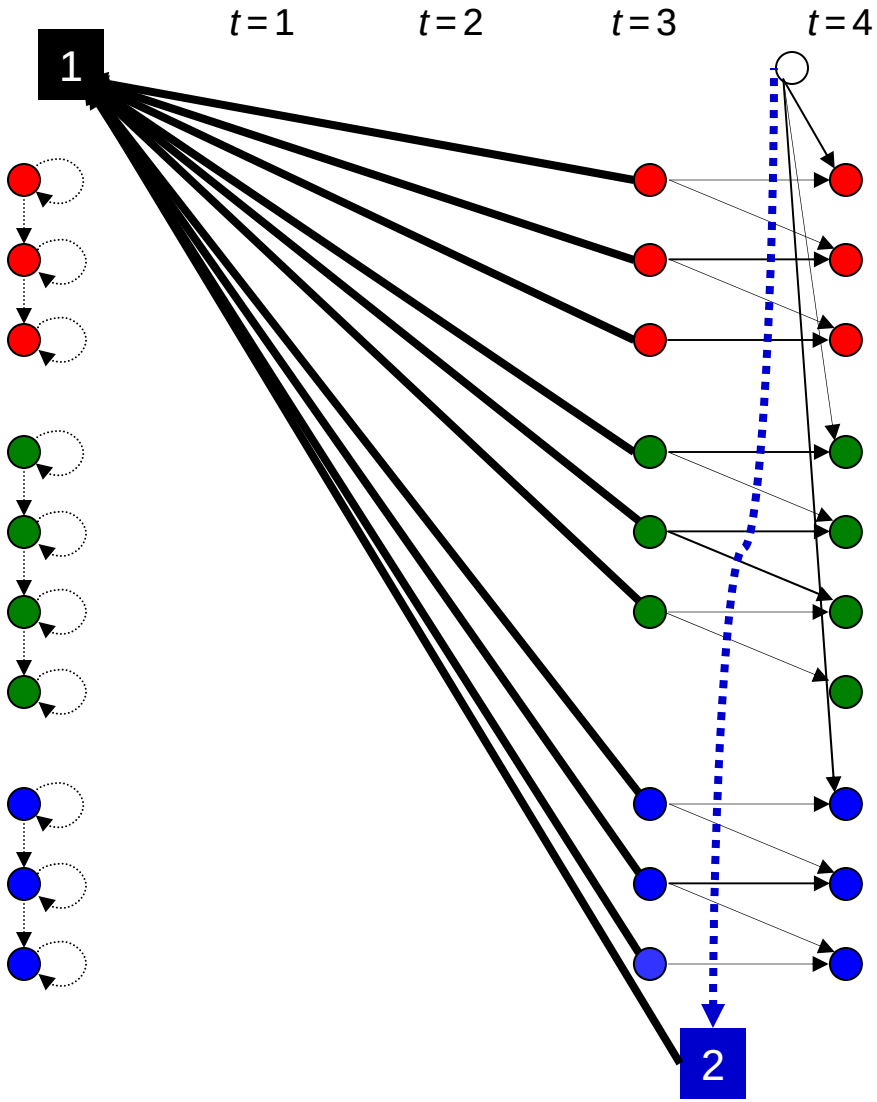
- The Backpointer table in the previous figure only retains sufficient history to obtain the best hypothesis
- Sometimes we would like to retain additional information in the backpointer table that tells us what other words were considered (not pruned out) during recognition
  - e.g. when we want to create lattices for finding N-best hypotheses or to compute confidence
- In this case the backpointer table is expanded to include *all* trellis nodes at the final states of words
  - Additionally, all trellis nodes corresponding to non-emitting nodes may also be stored

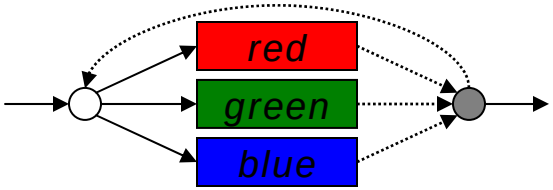
Word-entry Pruning

- Word-entry points behave differently from the rest of the trellis
- Typically the identity of a word is most clear in the initial frames
- Language probabilities also get introduced at this point

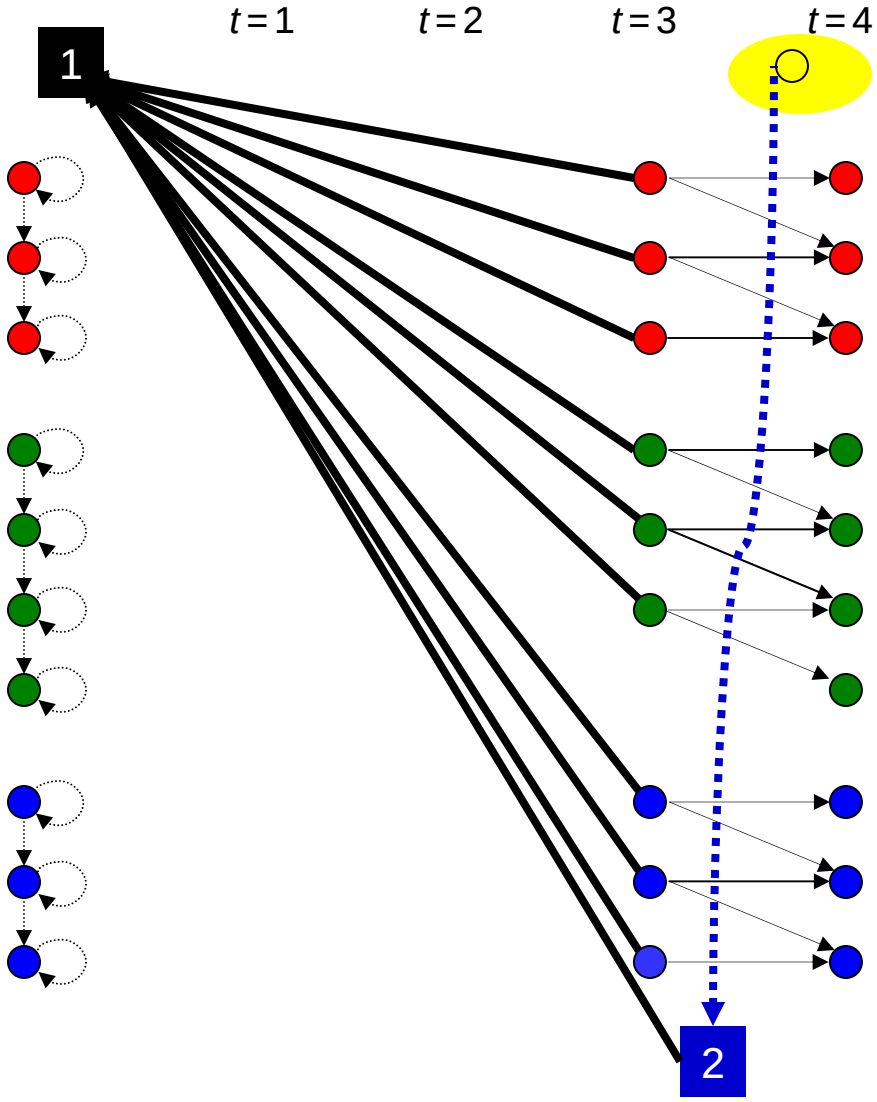
- Additional computational benefits may be drawn from this observation by pruning word entries differently from other points in the lattice
- Two forms of pruning
  - Absolute: Only allow the best N word entries at any time to survive
  - Relative: Only allow word entries within a small beam of the best word entry score to survive

Word Entries





Word Entries

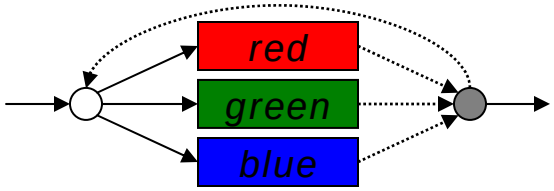


Entry into generating states heralds the possible beginning of a new word

Pick the "generating" state
(In this example there is only one: This is a very simple graph)

Retain all generating states with a score within a small beam of this

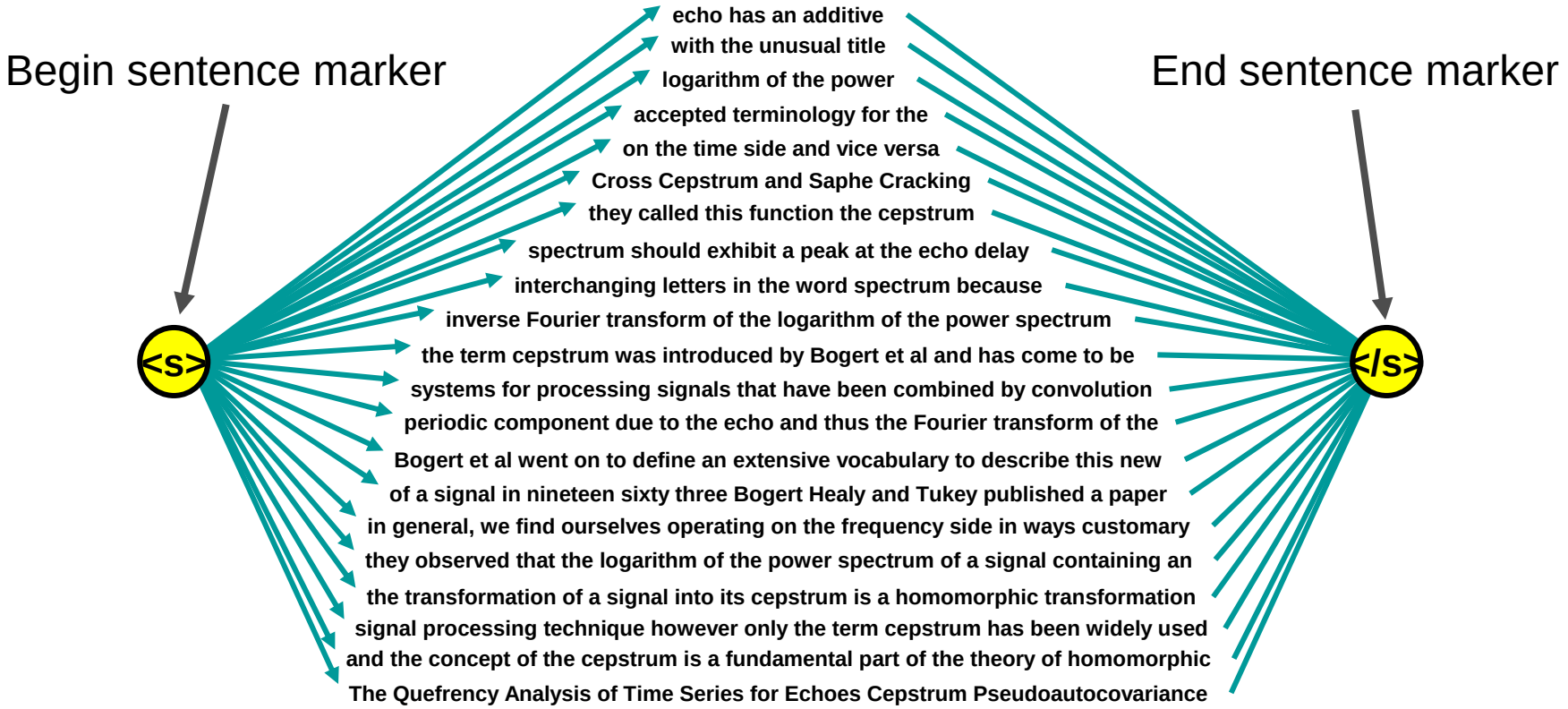
Alternately retain the best N generating states



QUESTIONS?

- Next up: Natural language
- Any questions on topics so far?

Bayesian Natural Language Recognition



- In a natural language task, we try to recognize from among all possible word sequences
  - A very large number
  - Implicitly, we have an infinitely large graph, where each path is a single word sequence

Compacting Natural Language: Factored Representations

- A factored representation of the a priori probability of a word sequence

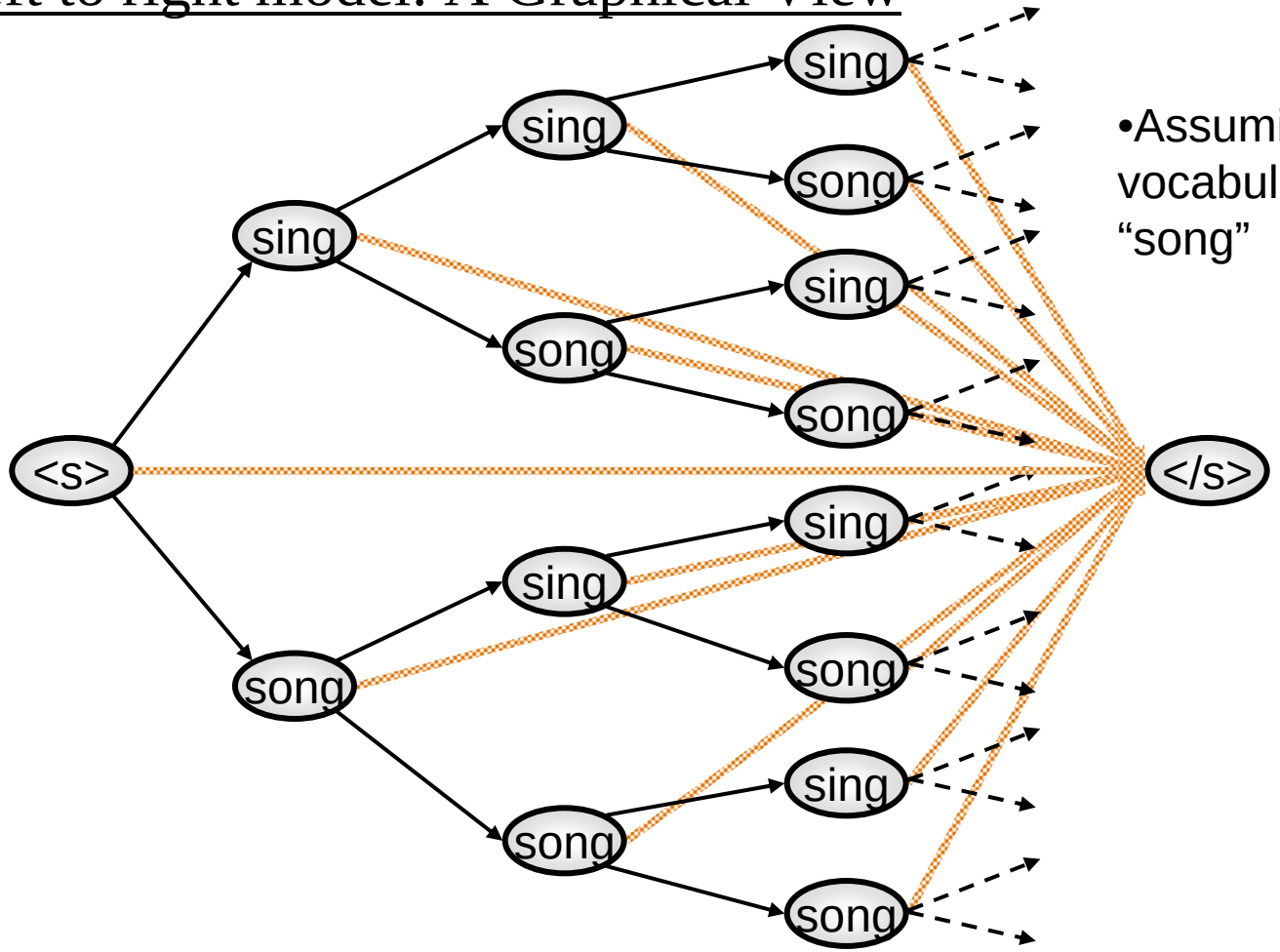
P(<s> word1 word2 word3 word4…</s>) =
P(<s>) P(word1 | <s>) P(word2 | <s> word1) P(word3 | <s> word1 word2)…

- P(word1 word2 word3 word4 …) is incrementally obtained :

word1
word1 **word2**
word1 word2 **word3**
word1 word2 word3 **word4**
…..

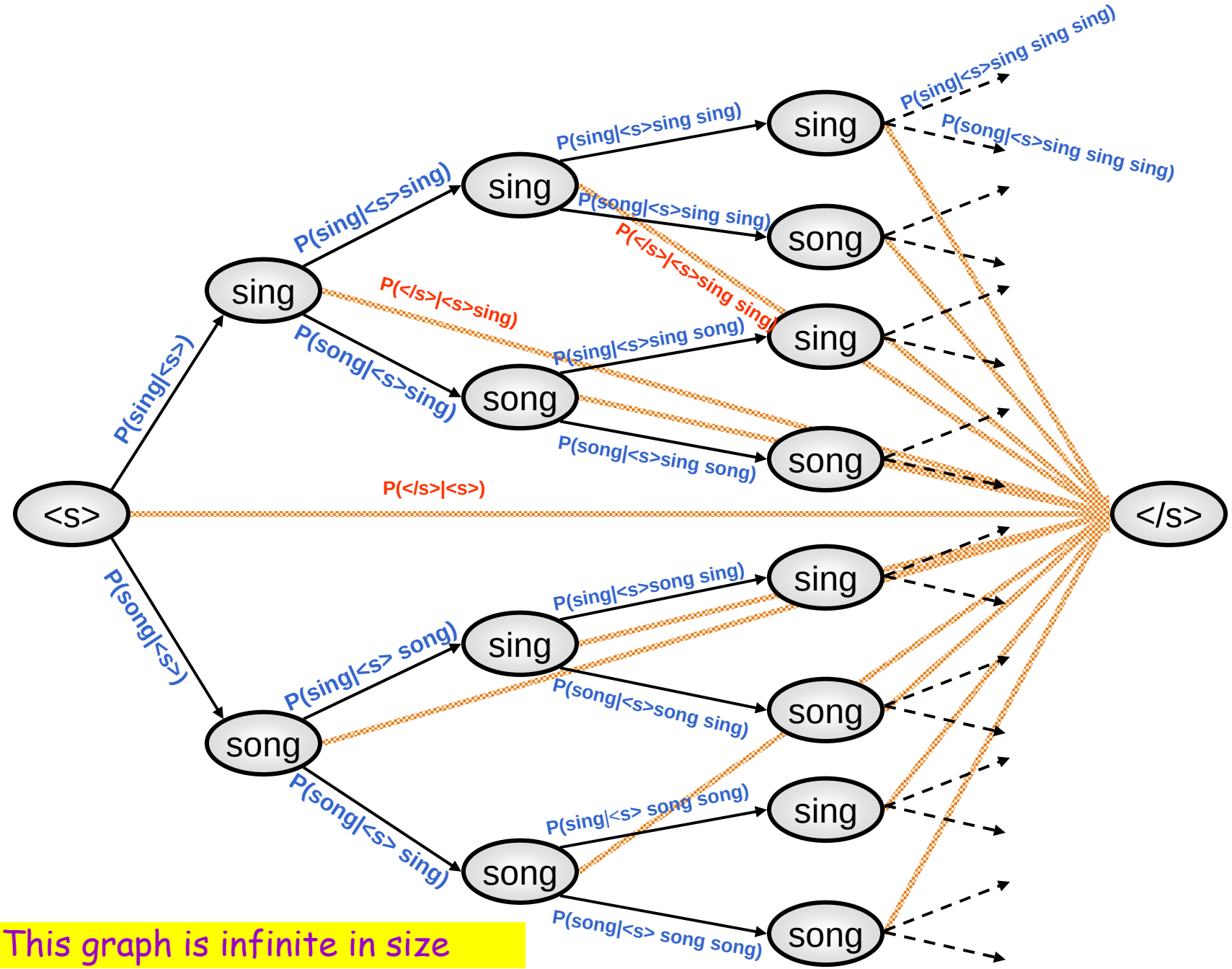
- This factored representation allows graph compaction
  - But the graphs will still be impossibly large

The left to right model: A Graphical View



•Assuming a two-word vocabulary: “sing” and “song”

- *A priori* probabilities for word sequences are spread through the graph
  - They are applied on every edge
- This is a much more compact representation of the language than the full graph shown earlier
- But is still ***infinitely*** large in size



This graph is infinite in size

Left-to-right language probabilities and the N-gram model

- The N-gram assumption

$$P(w_K \mid w_1, w_2, w_3, \ldots w_{K-1}) = P(w_K \mid w_{K-(N-1)}, w_{K-(N-2)}, \ldots, w_{K-1})$$

- The probability of a word is assumed to be dependent only on the past N-1 words
  - For a 4-gram model, the probability that a person will follow "two times two is" with "four" is assumed to be identical to the probability that they will follow "seven times two is" with "four".

- This is not such a poor assumption
  - Surprisingly, the words we speak (or write) at any time are largely (but not entirely) dependent on the previous 3-4 words.
- Speech recognition systems typically use 3-gram or TRIGRAM models

N-gram LMs and compact graphs

- The graphical representation of language can be reduced by making assumptions about langauge generation
  - The N-gram assumption!
- By restricting the order of an N-gram LM, the inifinitely sized tree-shaped graph representing the language can be collapsed into finite-sized graphs.
- In our example possible word sequences are
  - Sing
  - Sing sing
  - Sing song sing
  - Sing sing song
  - Song
  - Song sing sing sing sing sing song
  - ….
  - …
- There are infinite possible sequences
- Lets see how we can collapse this::

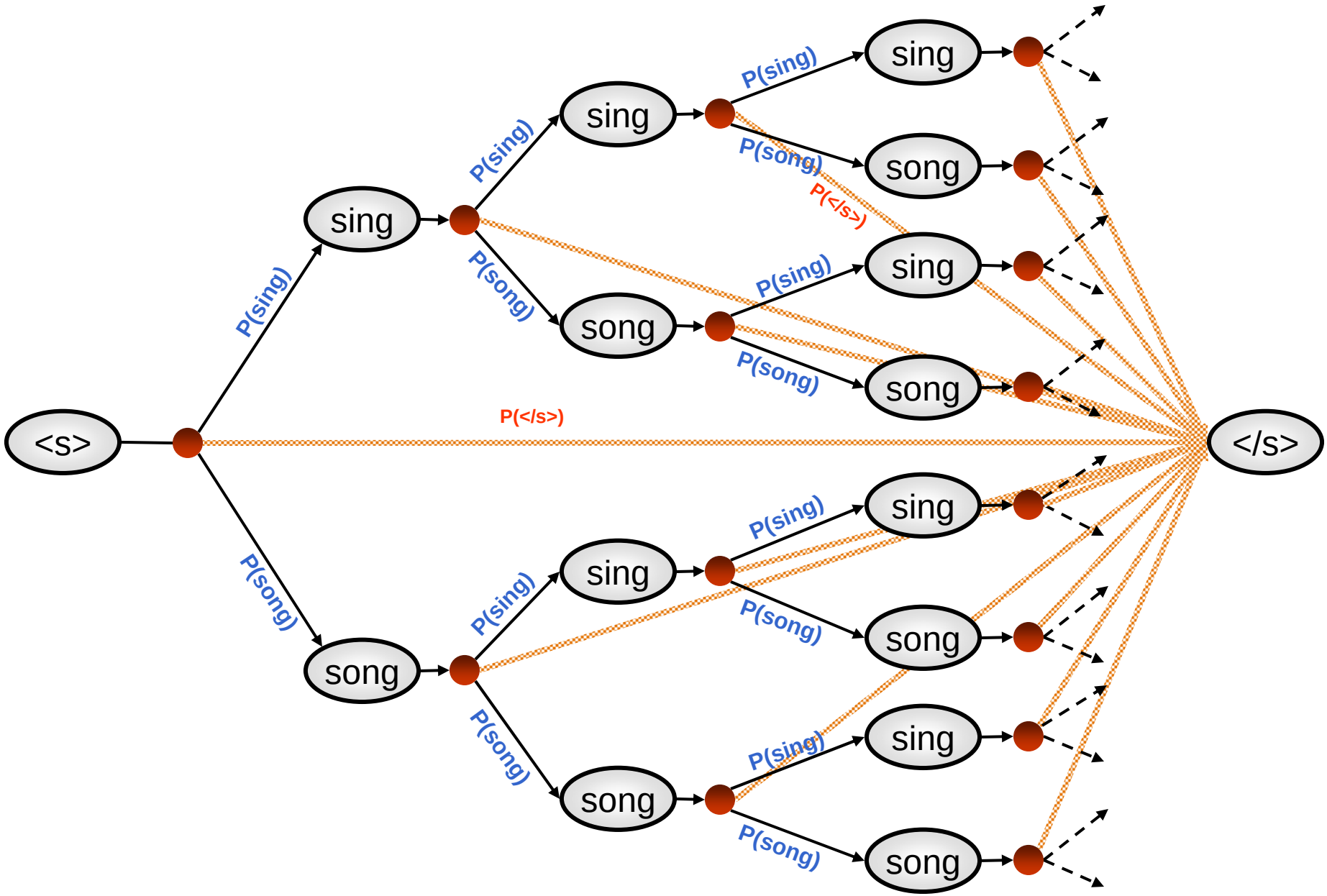
Unigram Representation

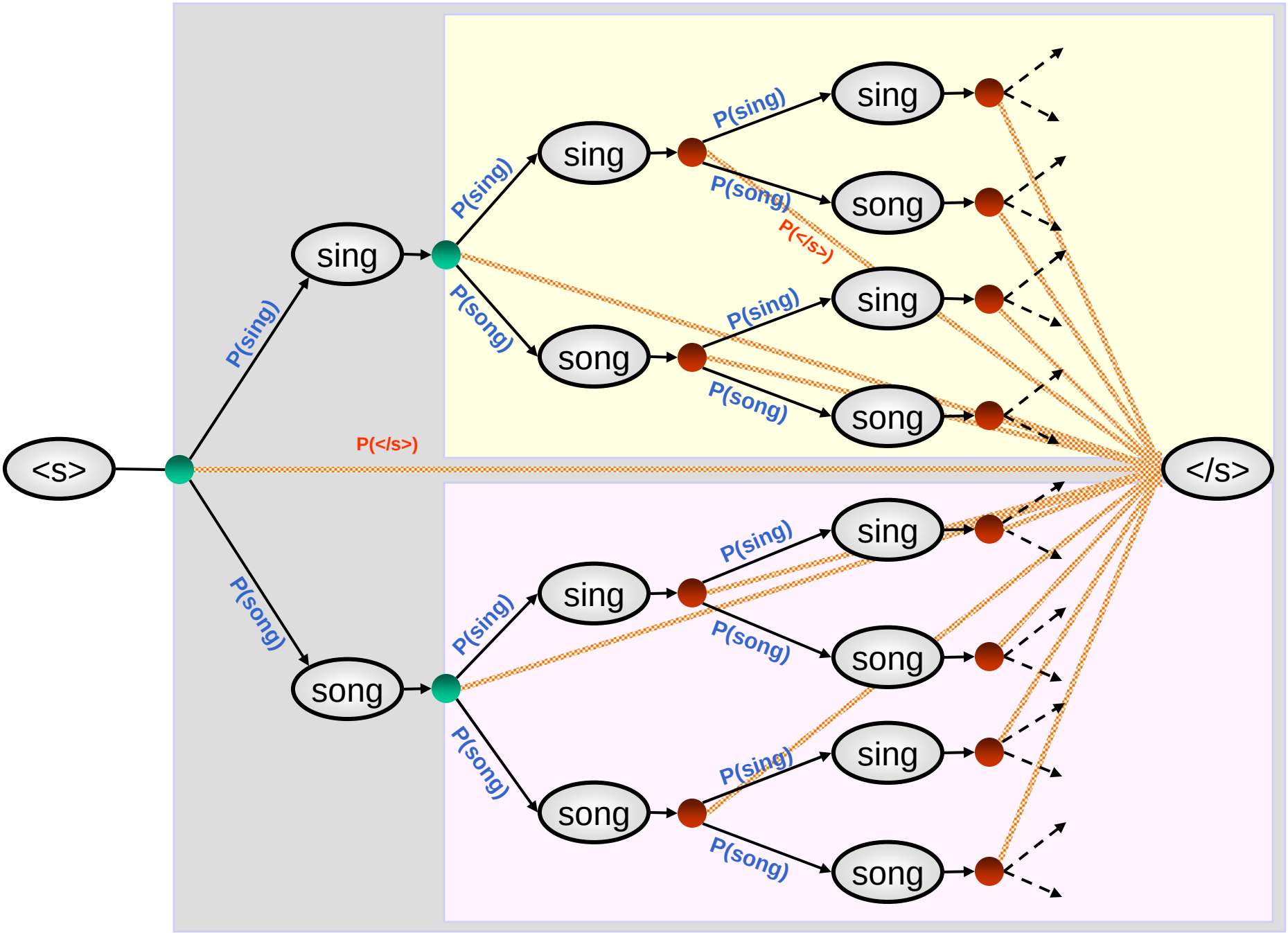
- A bag of words model:
  - Any word is possible after any word.
  - The probability of a word is independent of the words preceding or succeeding it.
- *P*(<s> word1 word2 word3..</s>) = P(word1)P(word2)P(word3)…P(</s>)
  - The "P(</s>)" is critical – it is the probability that the sequence terminates
  - If we didn't have it, every sequence must be infinitely long according to this model
  - The total probability of all possible word sequences must sum to 1.0
  - Only if P(</s>) is explicitly defined will the total probability = 1.0

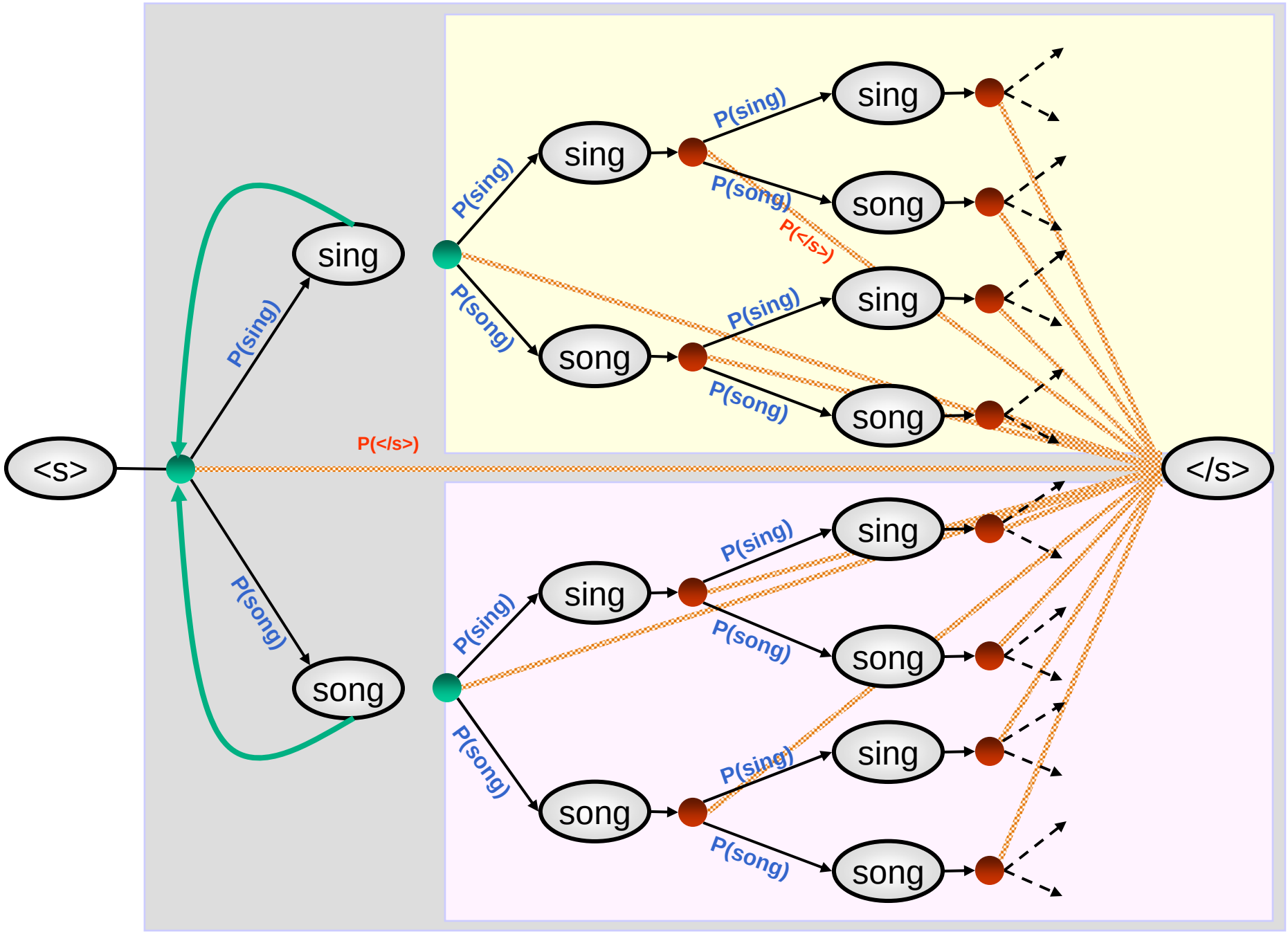
$$P(Star) = P(Star \mid Dog) = P(Star \mid Rock) = P(Star \mid When\ you\ wish\ upon\ a)$$

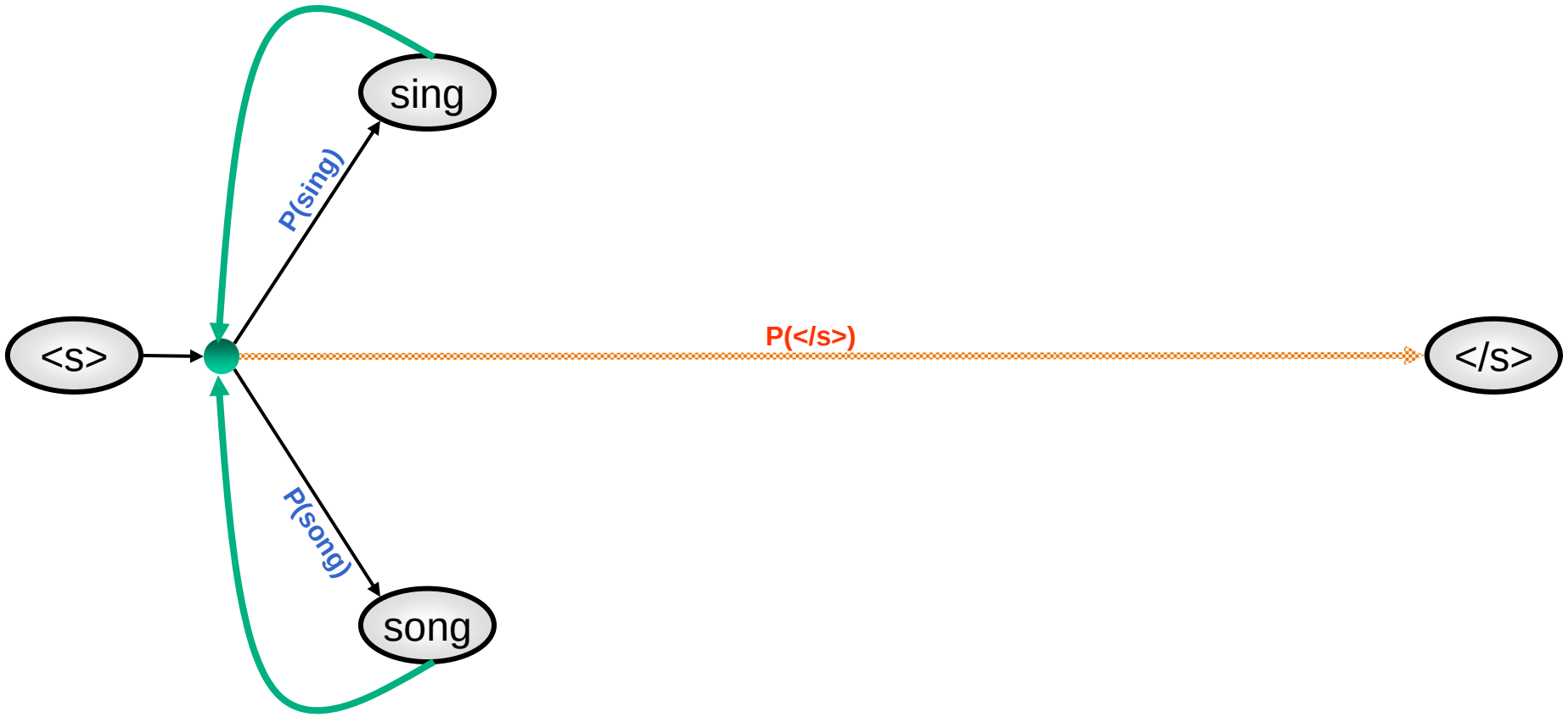
$$P(When\ you\ wish\ upon\ a\ star) =$$
$$P(When)\ P(you)\ P(wish)\ P(upon)\ P(a)\ P(star)\ P(</s>)$$

P(sing song sing song.. </s>) = P(sing)P(song)P(sing)P(song)…P(</s>)



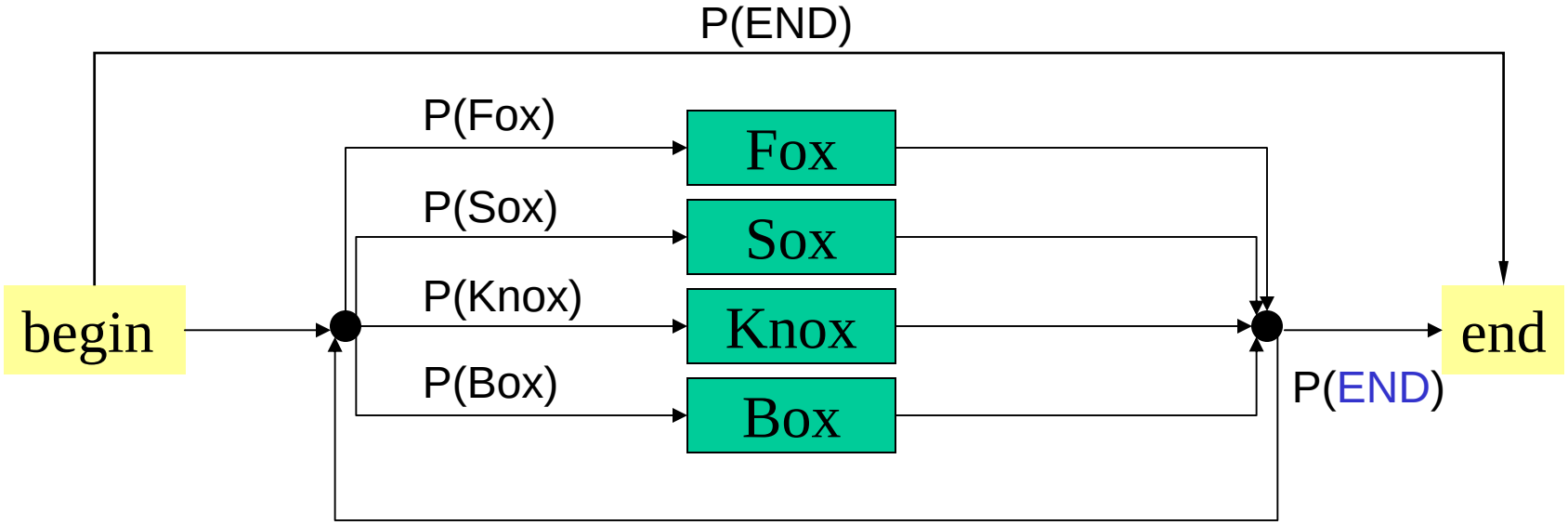

<s>
sing
song
P(sing)
P(song)
P(</s>)
sing
song
sing
song
sing
song
sing
song
sing
song
sing
song
sing
song
</s>


<s>
sing
song
P(sing)
P(song)
P(</s>)
sing
song
sing
song
sing
song
P(sing)
P(song)
P(</s>)
</s>


<s>
sing
song
P(sing)
P(song)
P(</s>)
</s>

Language HMMs for Natural language: example of a graph that uses unigram representations

- Vocabulary: "fox", "sox", "knox", "box"
- Arbitrary length sequences of arbitrary arrangements of these fours words are possible.
- Actual probabilities:
  - P(fox), P(sox), P(knox), P(box), P(END)
  - P(fox) + P(sox) + P(knox) + P(box) + P(END) = 1.0



The black dots are non-emitting states that are not associated with observations

Language HMMs for Natural language: bigram representations

- A unigram model is only useful when no statistical dependency between adjacent words can be assumed
  - Or, alternately, when the data used to learn these dependencies are too small to learn them reliably
    - Learning word dependencies: Later in the program

- In natural language, the probability of a word occurring depends on past words.
- Bigram language model: the probability of a word depends on the previous word
- P(Star | A Rock) = P(Star | The Rock) = P(Star | Rock)
- P(Star | Rock) is not required to be equal to P(Star | Dog)
  - In fact the two terms are assumed to be unrelated.

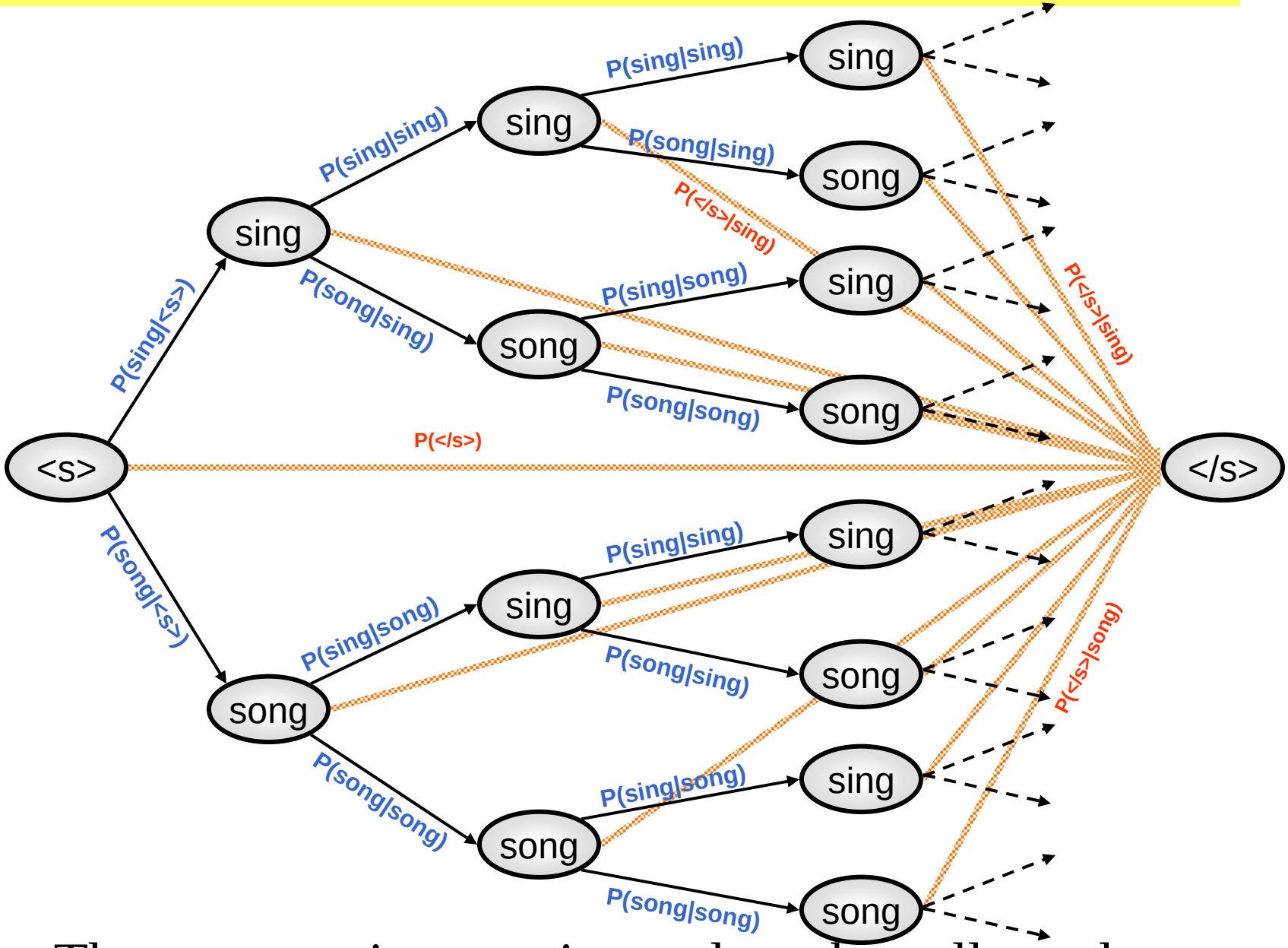
Language HMMs for Natural language: bigram representations

- Simple bigram example:
  - Vocabulary: <s>, “sing”, “song”, </s>
  - P(sing | </s>) = a, P(song | </s>) = b, P(</s> | <s>) = c
    - a+b+c = 1.0
  - P(sing| sing) = d, P(song | sing) = e, P(</s> | sing) = f
    - d+e+f = 1.0
  - P(sing | song) = g, P(song | song) = h, P(</s> | song) = i
    - g+h+i = 1.0

- <s> is a special symbol, indicating the beginning of the utterance
  - P(word | <s>) is the probability that the utterance begins with word
  - Prob(“sing song sing song”) =
    P(sing | <s>) P(song | sing) P(sing | song) P (song | sing) P(</s> | song)

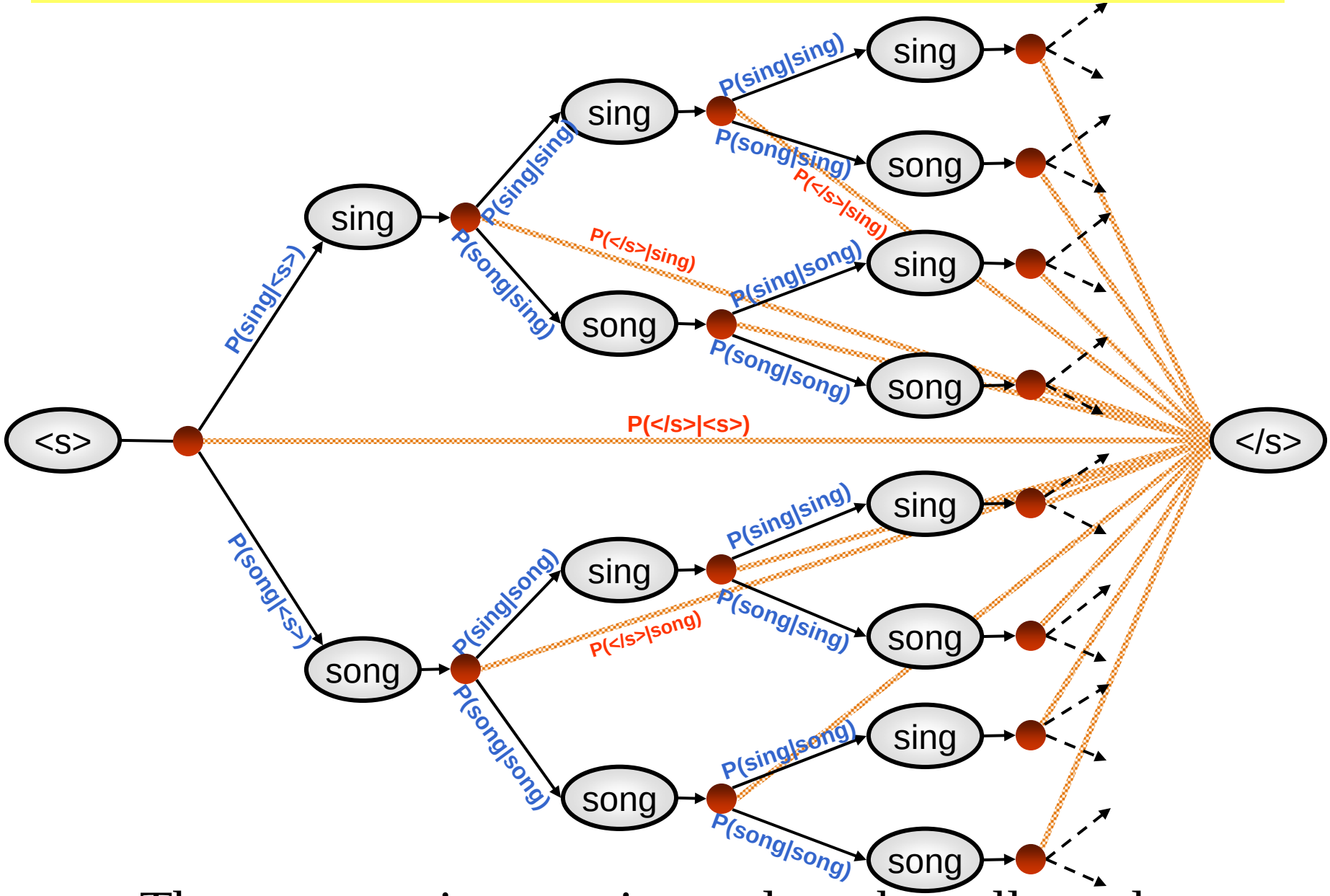
- Can be shown that the total probability of all word sequences of all lengths is 1.0
  - Again, the definition of <s> and </s> symbols, and all bigrams involving the two, is crucial

The two-word example as a full tree with a bigram LM

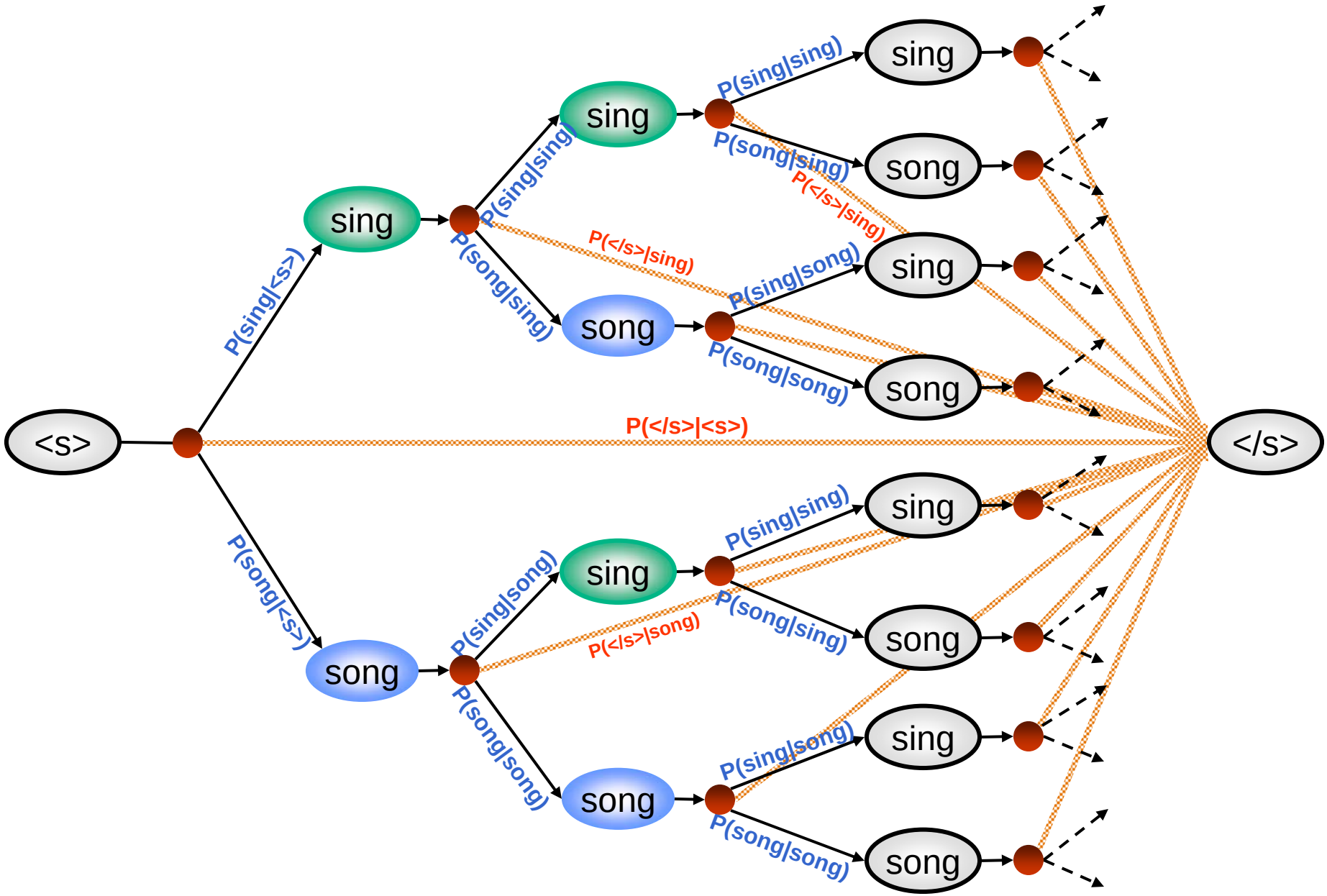


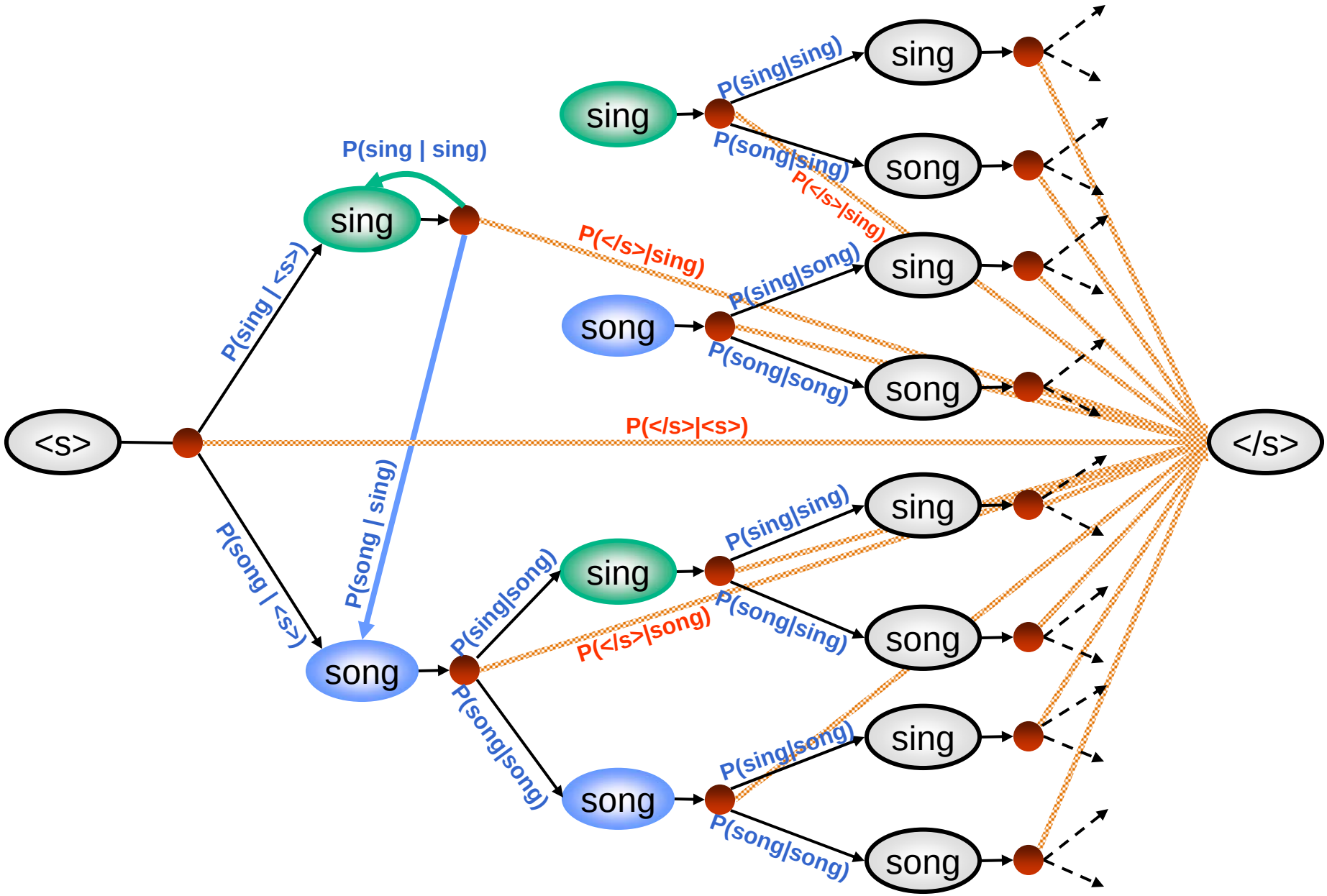
- The structure is recursive and can be collapsed

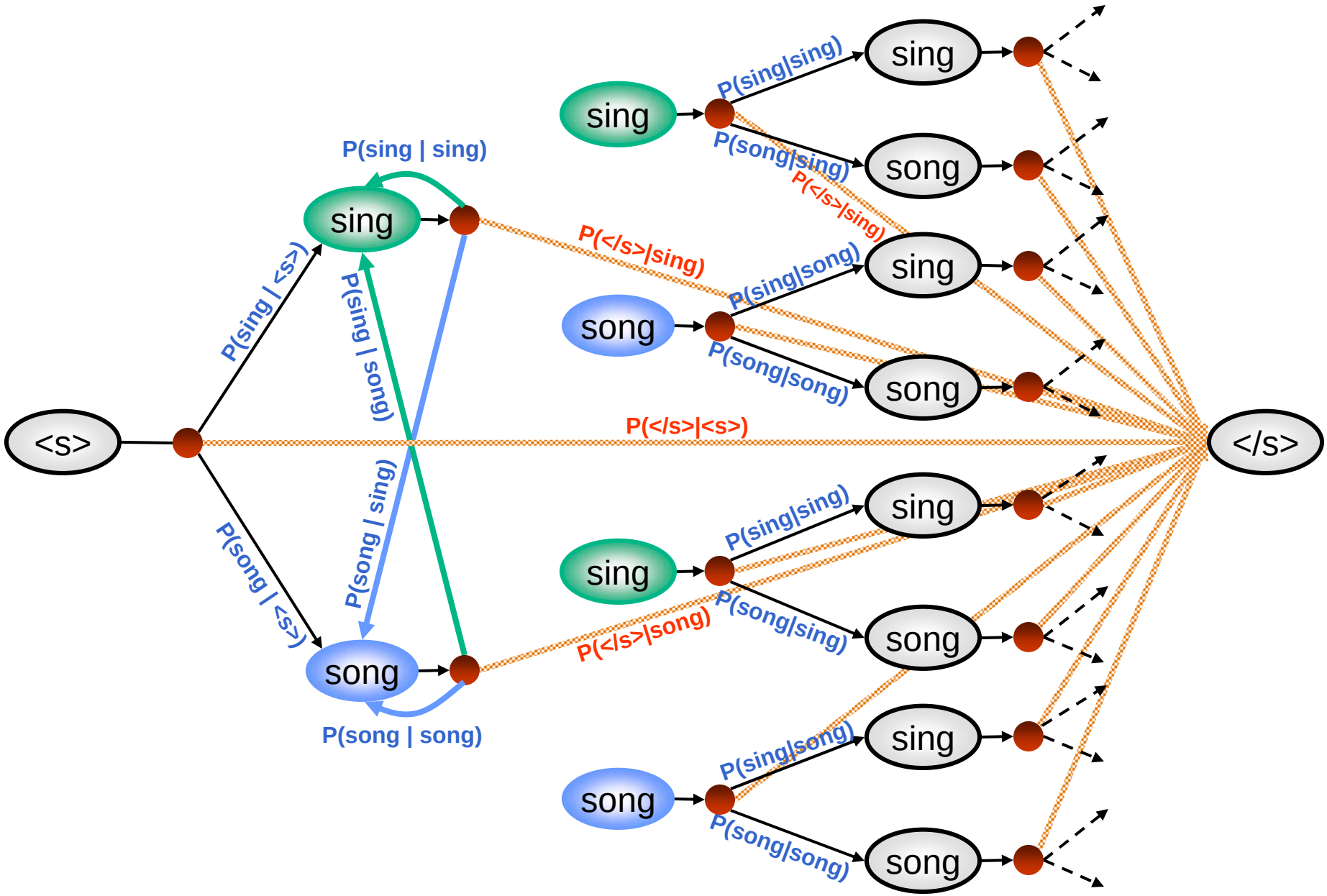
The two-word example as a full tree with a bigram LM

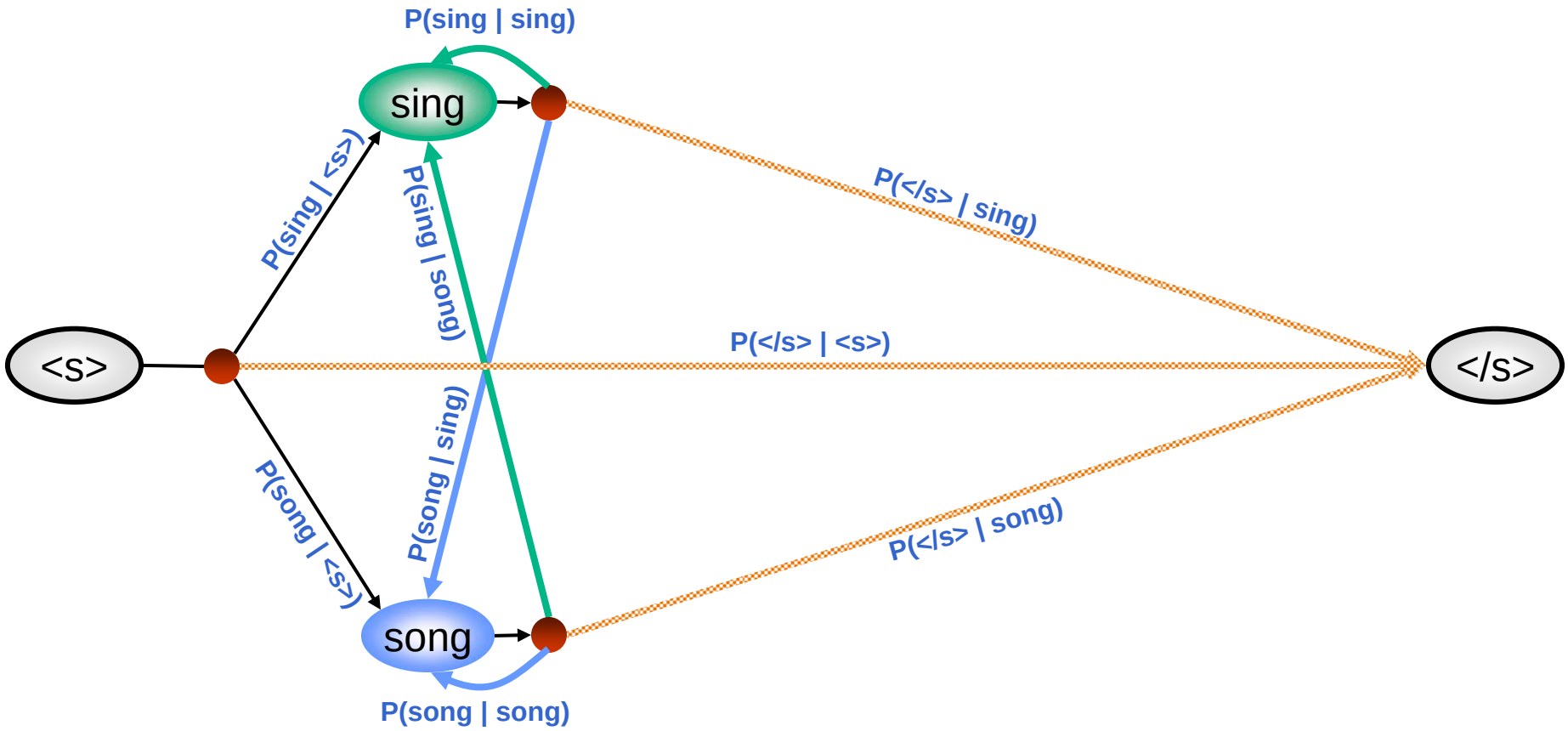


- The structure is recursive and can be collapsed


<s>
sing
song
P(sing|<s>)
P(song|<s>)
P(sing|sing)
P(song|sing)
P(sing|song)
P(song|song)
P(</s>|sing)
P(</s>|song)
P(</s>|<s>)
</s>


<s>
sing
song
P(sing | sing)
P(sing | <s>)
P(song | <s>)
P(song | sing)
P(</s>|sing)
P(</s>|<s>)
P(sing|sing)
P(song|sing)
P(</s>|sing)
P(sing|song)
P(song|song)
P(sing|song)
P(song|song)
P(</s>|song)
P(sing|sing)
P(song|sing)
P(sing|song)
P(song|song)
</s>

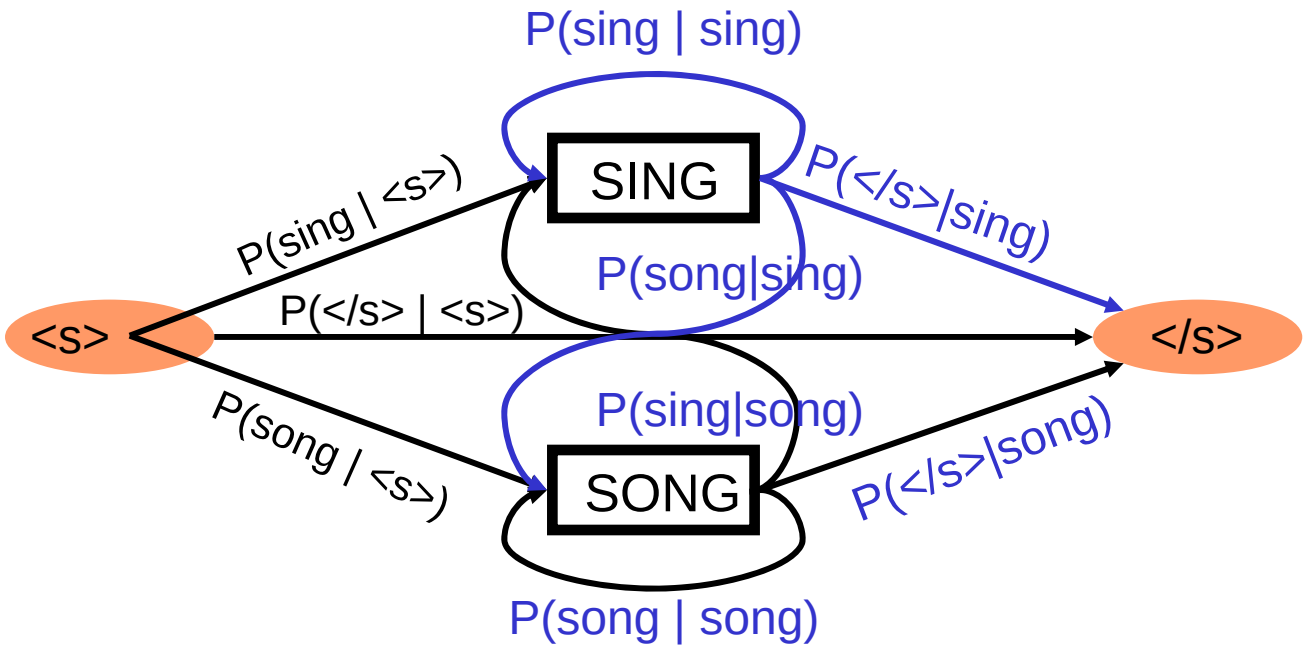

<s>
sing
song
P(sing | <s>)
P(song | <s>)
P(sing | sing)
P(song | sing)
P(sing | song)
P(song | song)
P(</s>|<s>)
P(</s>|sing)
P(</s>|song)
sing
song
P(sing|sing)
P(song|sing)
P(sing|song)
P(song|song)
sing
song
P(sing|sing)
P(song|sing)
P(sing|song)
P(song|song)
</s>


P(sing | sing)
sing
P(sing | <s>)
P(sing | song)
P(</s> | sing)
<s>
P(</s> | <s>)
</s>
P(song | <s>)
P(song | sing)
P(</s> | song)
song
P(song | song)

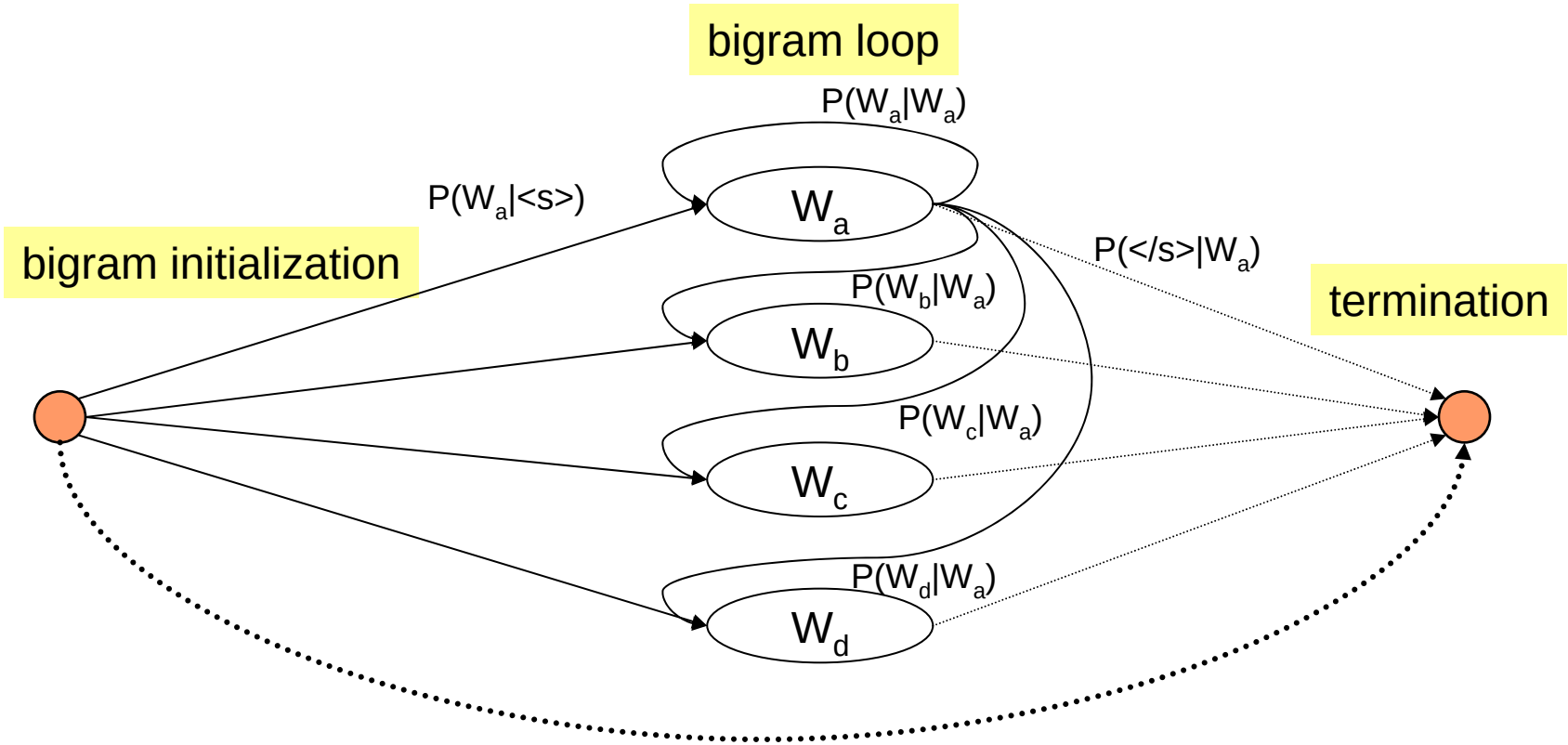
Language HMMs for Natural language: building graphs to incorporate bigram representations

Each word is an HMM

- Edges from “START” contain START dependent word probabilities
- Edges from “Even” contain “Even” dependent word probabilities
- Edges from “Odd” contain “Odd” dependent word probabilities



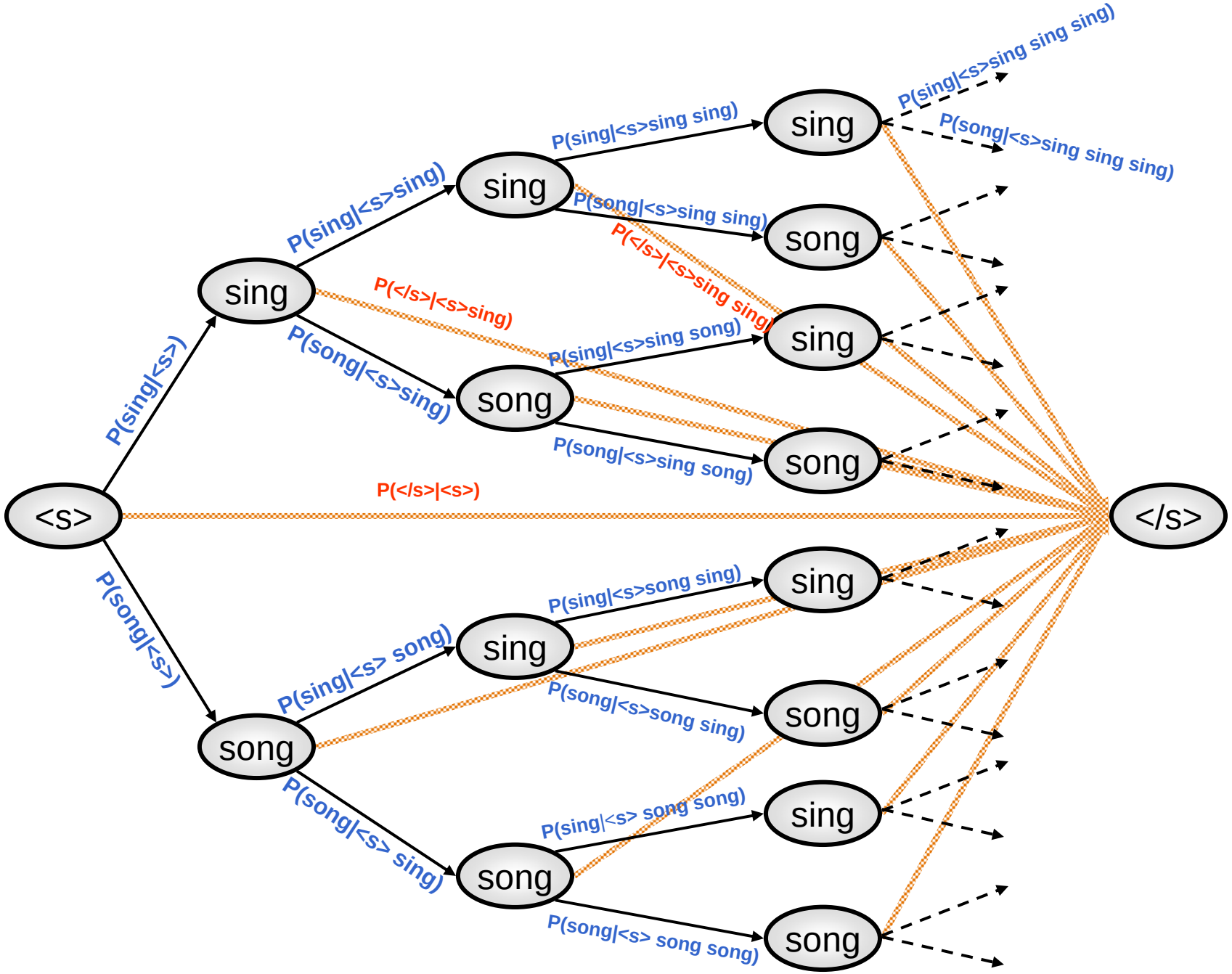
Language HMMs for Natural language: building graphs to incorporate bigram representations



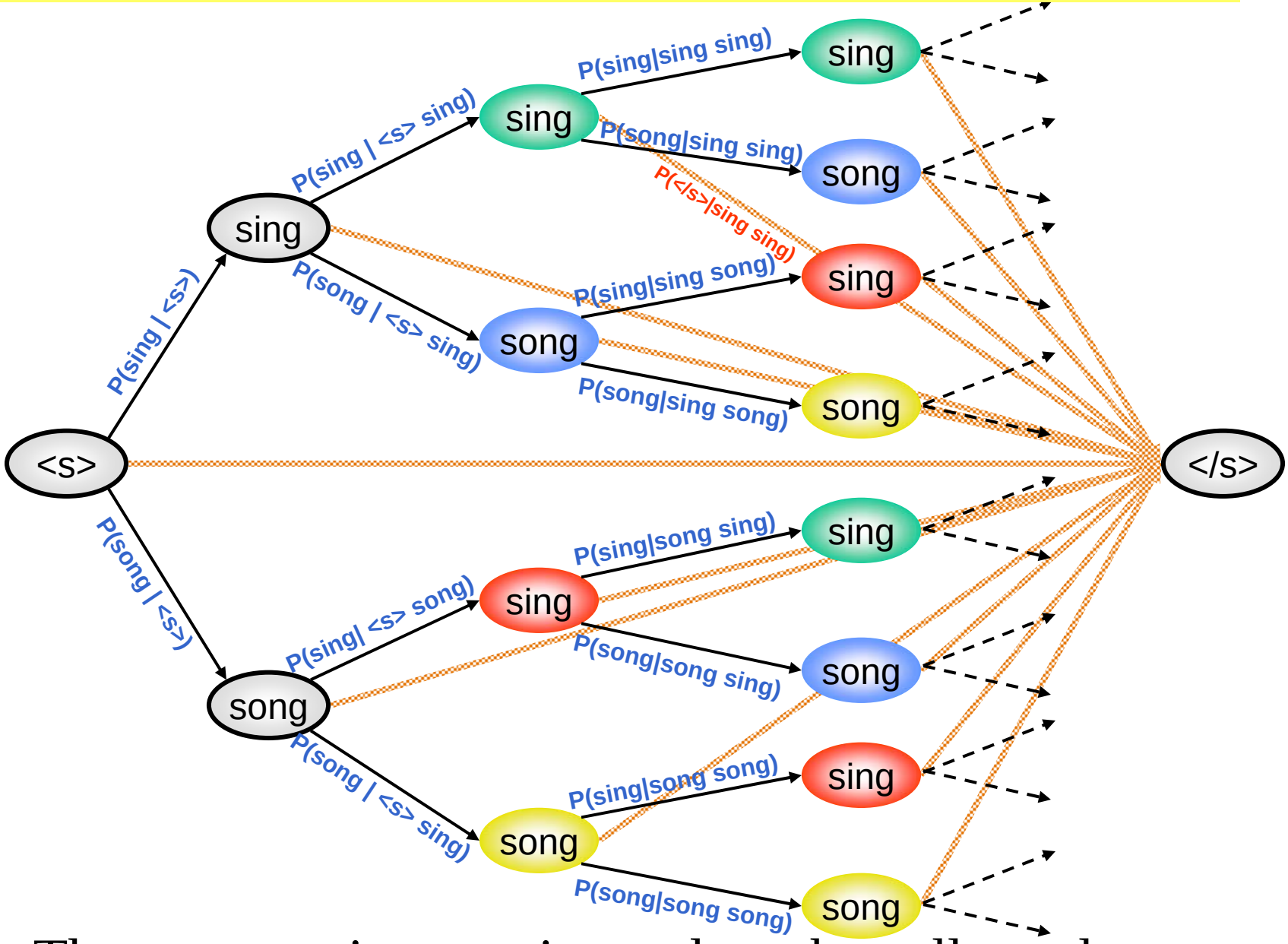
- The edges going out of any word "Word" have probabilities of the form P(Word2 | Word)

Language HMMs for Natural language: trigram representations

- The probability of a word depends on the previous two words
- P(when you wish upon a star) =
  P(when|<s>)P(you | <s>when)
  P(wish|when you)P(upon|you wish)P(a|wish upon)
  P(star| upon a)P(</s>|a star)
- Note that the very *first* word only has a bigram probability
  - P(when | <s>)
    - The first word only has the "start of sentence" marker as history
  - The *second* word actually is modelled by a trigram:
    P(you|<s>when)

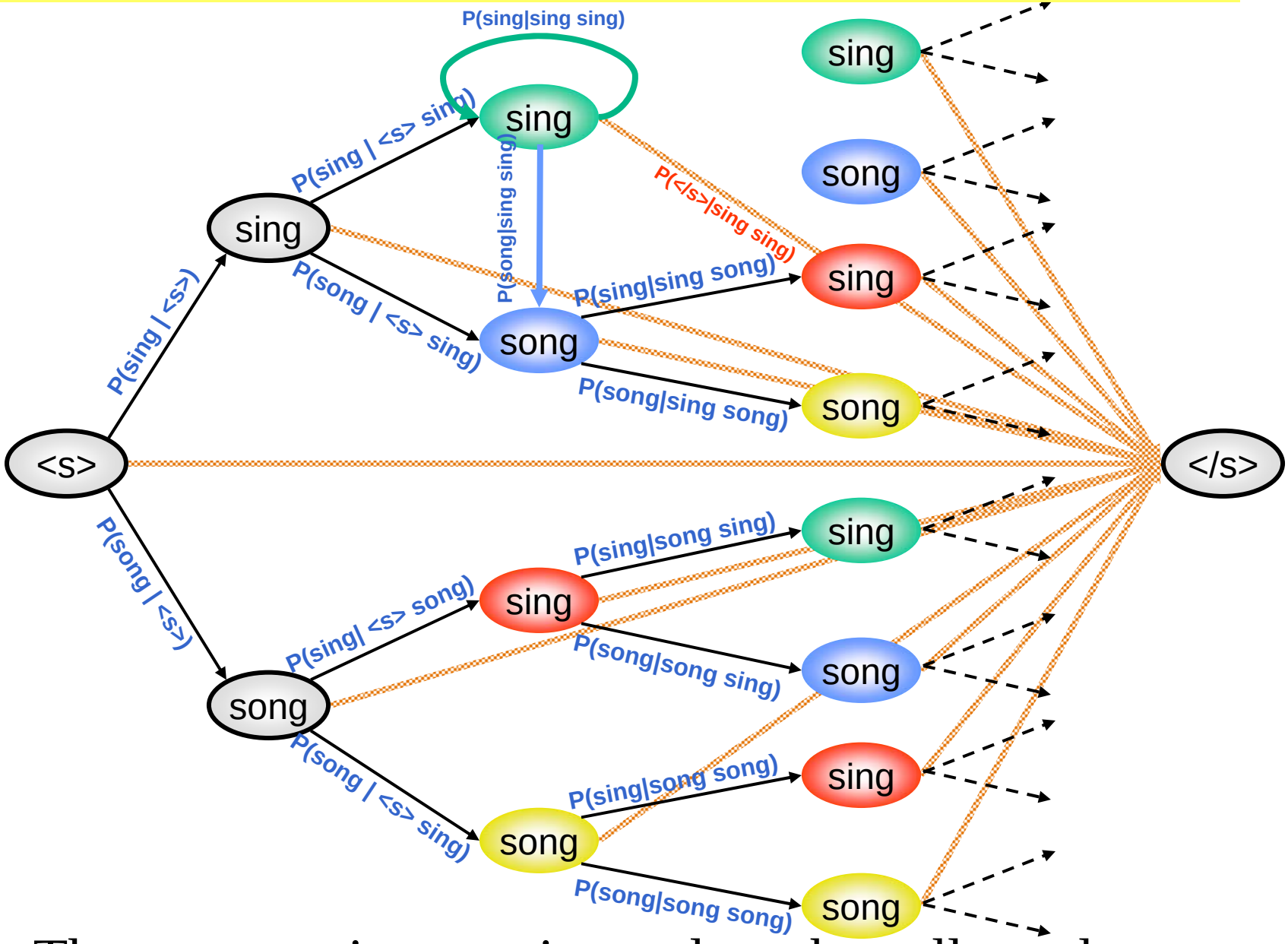

<s>
sing
song
sing
song
sing
song
sing
song
sing
song
sing
song
</s>
P(sing|<s>)
P(song|<s>)
P(sing|<s>sing)
P(song|<s>sing)
P(sing|<s> song)
P(song|<s> sing)
P(sing|<s>sing sing)
P(song|<s>sing sing)
P(sing|<s>sing song)
P(song|<s>sing song)
P(sing|<s>song sing)
P(song|<s>song sing)
P(sing|<s> song song)
P(song|<s> song song)
P(sing|<s>sing sing sing)
P(song|<s>sing sing sing)
P(</s>|<s>)
P(</s>|<s>sing)
P(</s>|<s>sing sing)

The two-word example as a full tree with a trigram LM

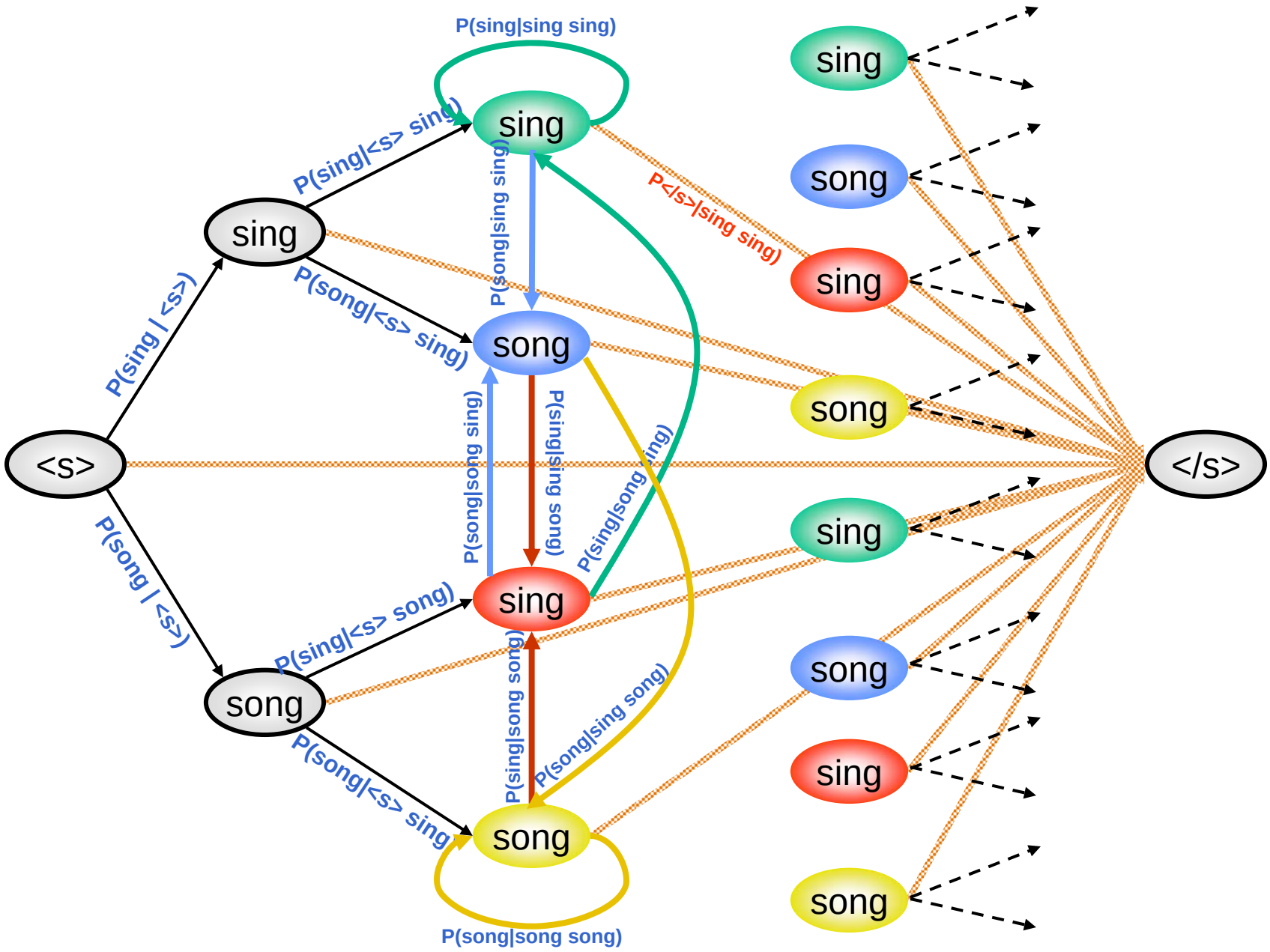


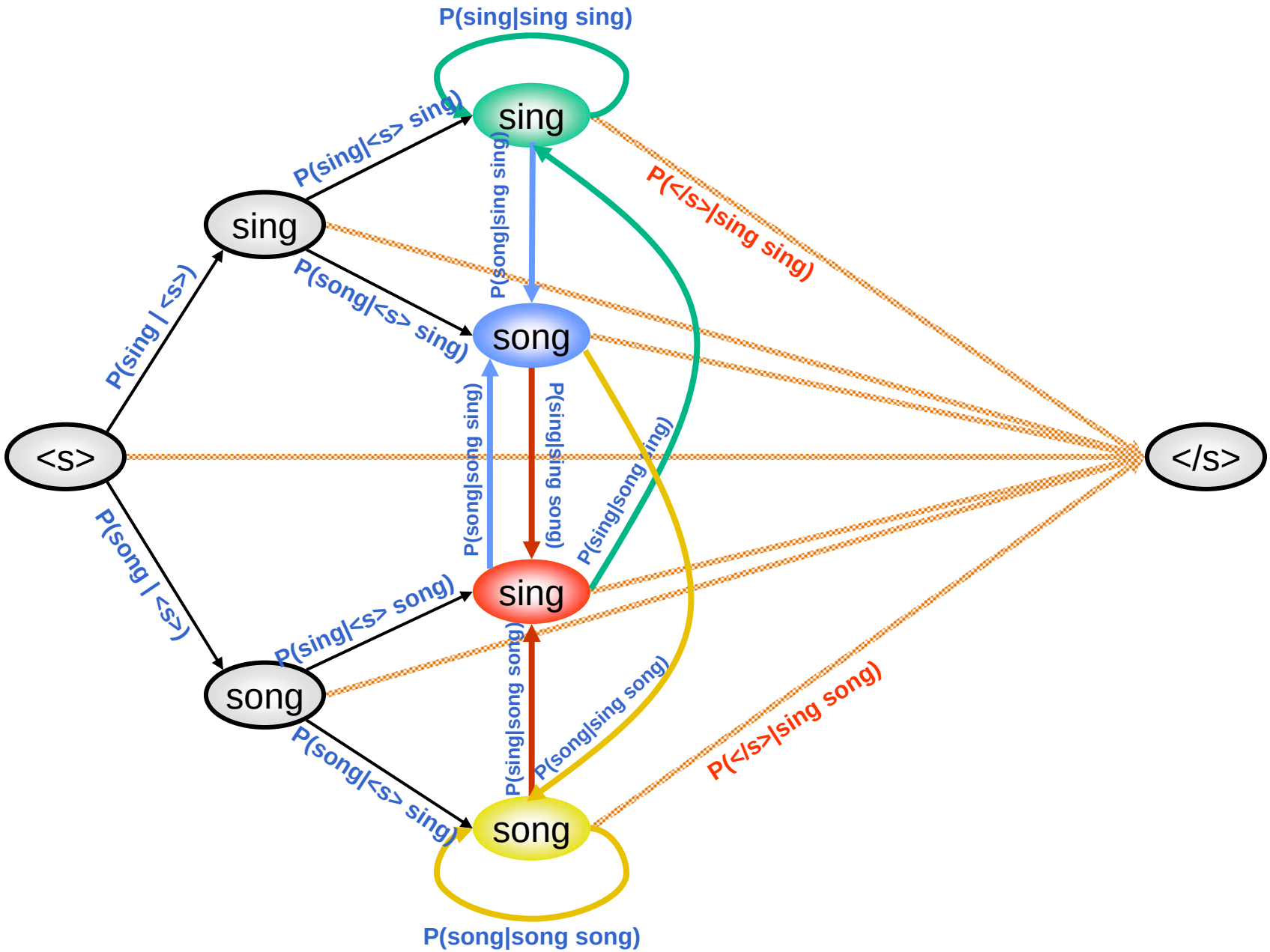
- The structure is recursive and can be collapsed

The two-word example as a full tree with a trigram LM



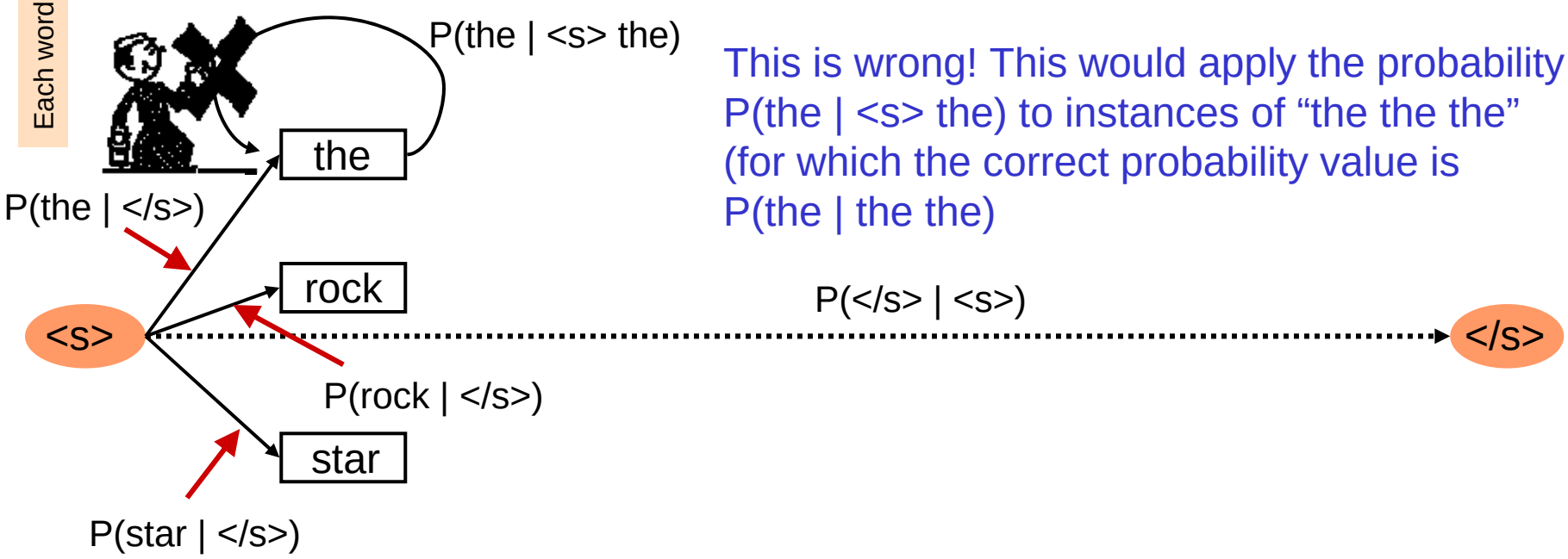
- The structure is recursive and can be collapsed


P(sing|sing sing)
sing
P(sing|<s> sing)
sing
P(song|sing sing)
P(song|<s> sing)
song
P(sing | <s>)
P(song | <s>)
<s>
P(song|song sing)
P(sing|sing song)
P(sing|<s> song)
sing
song
P(sing|song song)
P(song|sing song)
P(song|<s> sing)
song
P(song|song song)
P<s>|sing sing)
sing
song
sing
song
sing
song
sing
song
</s>


P(sing|sing sing)
sing
P(sing|<s> sing)
P(</s>|sing sing)
sing
P(song|sing sing)
P(song|<s> sing)
song
P(sing | <s>)
P(song|song sing)
P(sing|song sing)
P(single|sing song)
<s>
</s>
P(song | <s>)
P(sing|<s> song)
sing
P(sing|song song)
P(song|sing song)
P(</s>|sing song)
song
P(song|<s> sing)
song
P(song|song song)

Trigram representations

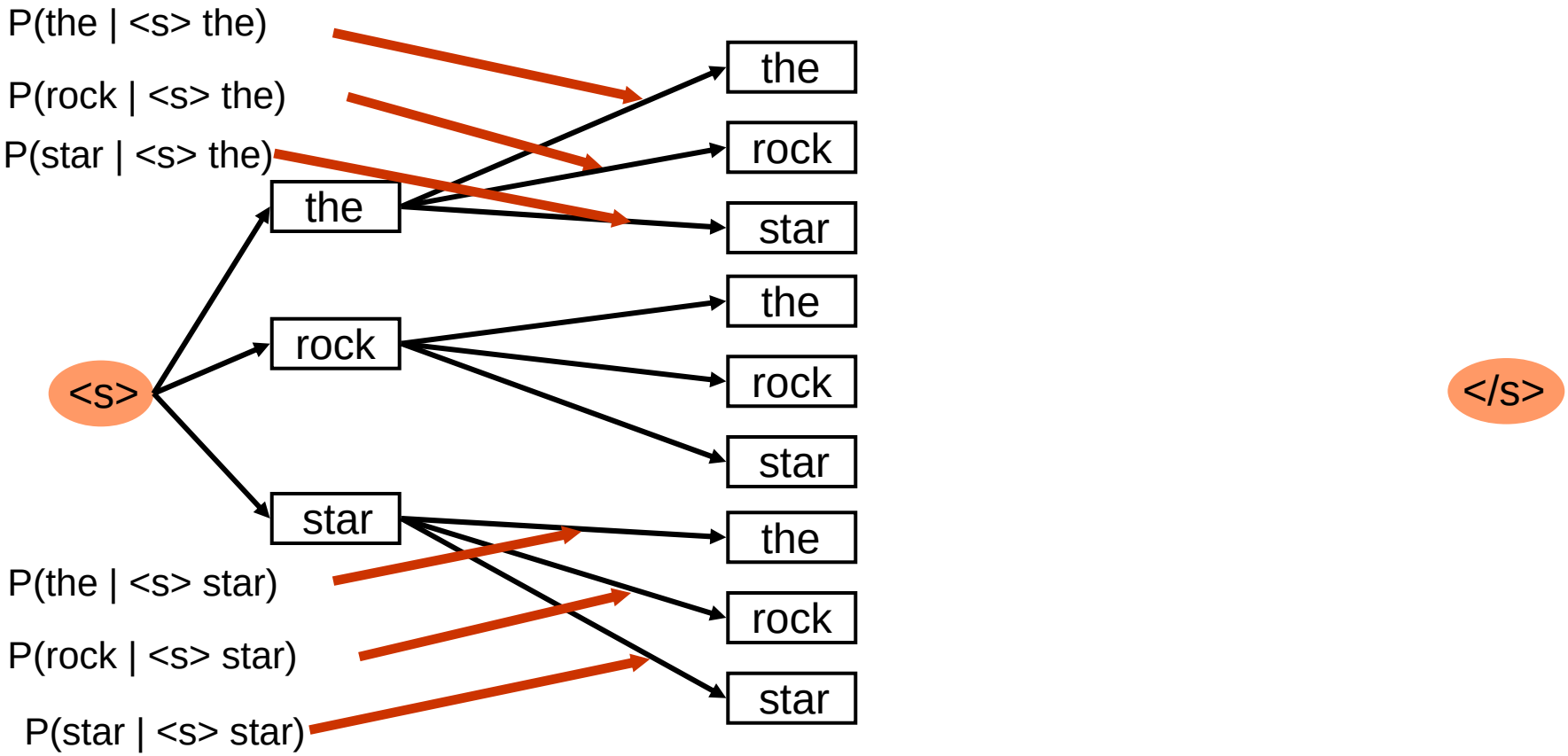
- Three word vocabulary “the”, “rock”, “star”
  - The graph initially begins with bigrams of <s>
  - There are edges from every node to “</s>”, that are not shown
  - Trigrams of “<s> the”..



This is wrong! This would apply the probability P(the | <s> the) to instances of “the the the” (for which the correct probability value is P(the | the the)

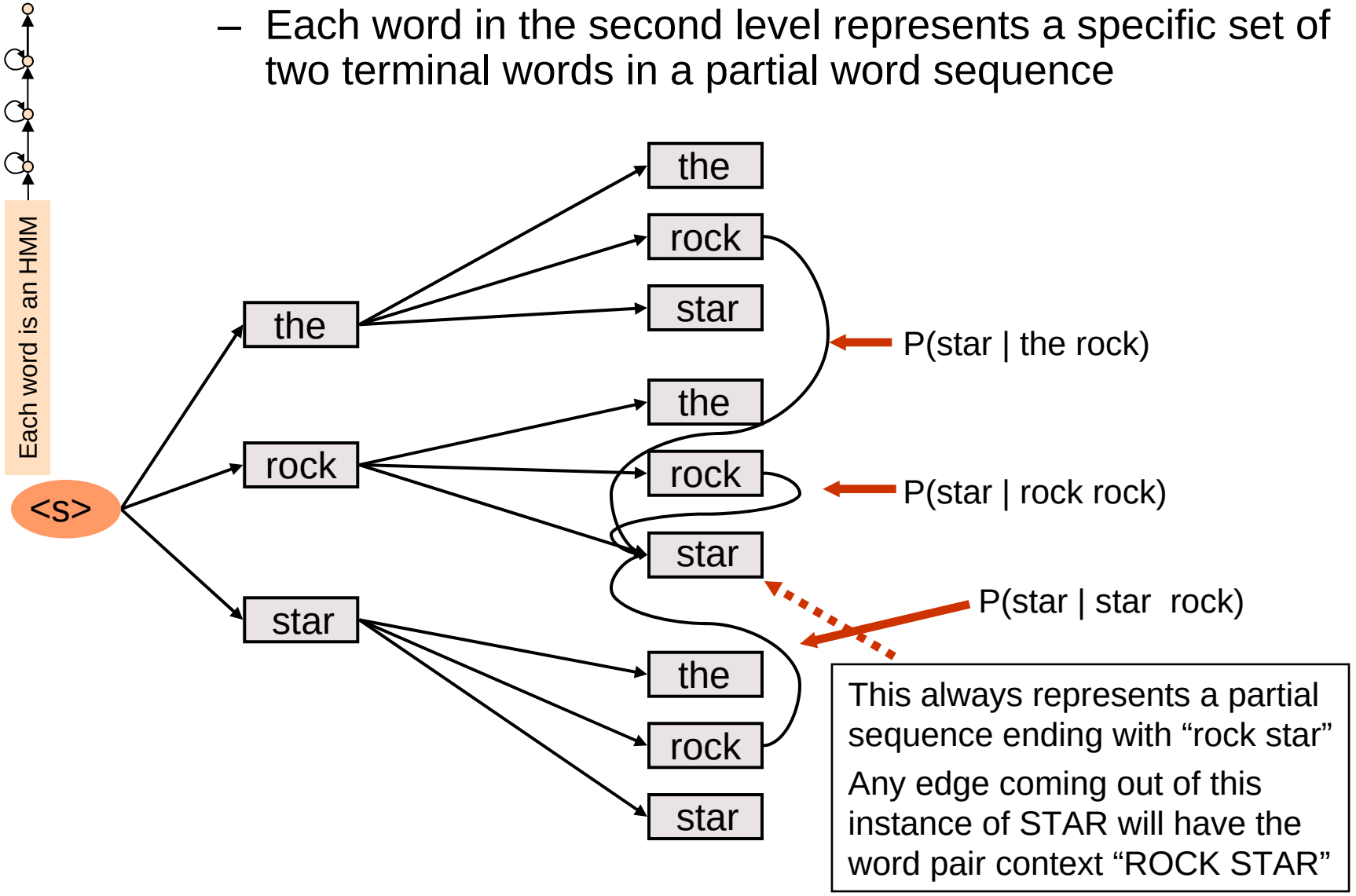
Trigram representations

- Trigrams for all "<s> word" sequences
  - A new instance of every word is required to ensure that the two preceding symbols are "<s> word"

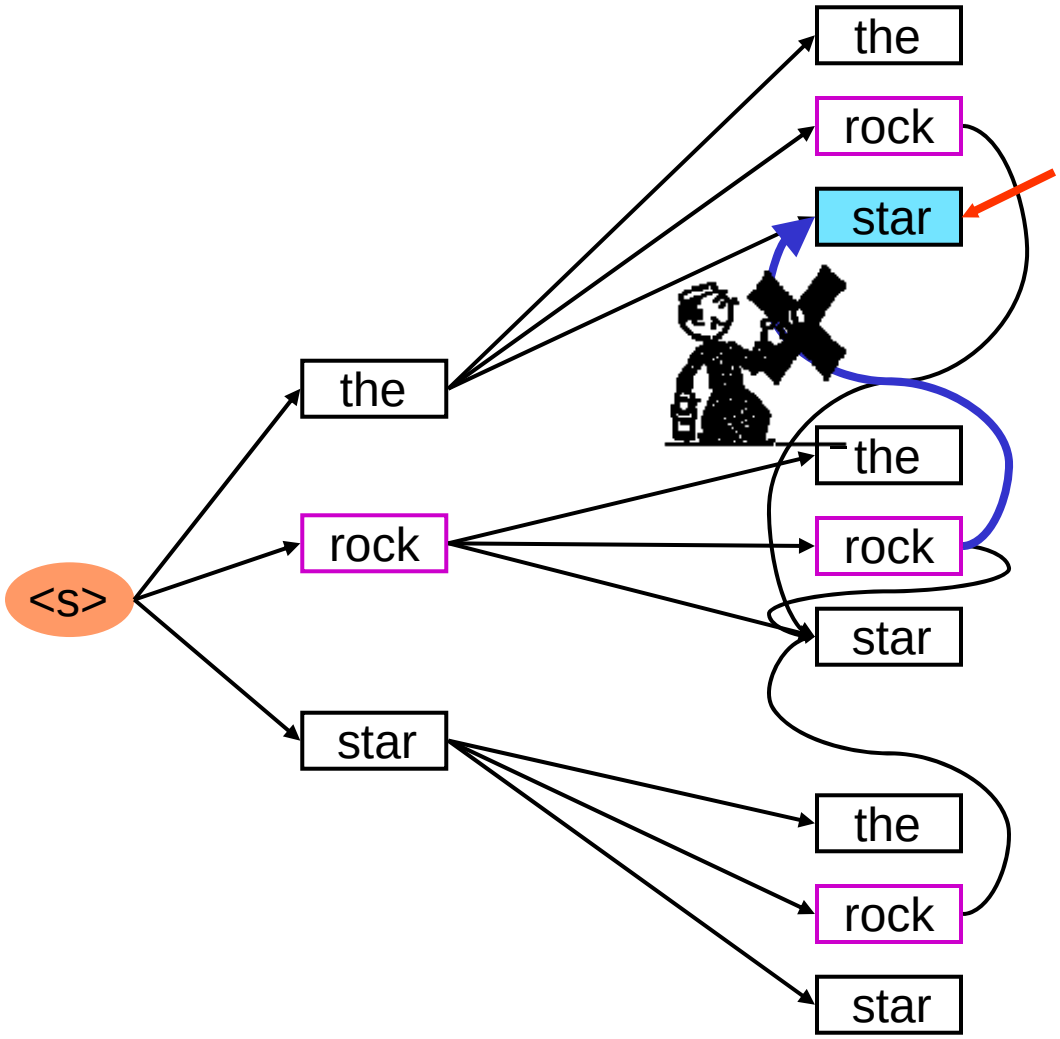


Trigram representations

– Each word in the second level represents a specific set of two terminal words in a partial word sequence



Trigram representations



Edges coming out of this wrongly connected STAR could have word pair contexts that are either "THE STAR" or "ROCK STAR". This is amibiguous. A word cannot have incoming edges from two or more different words

Generic N-gram representations

- The logic can be extended:
- A trigram decoding structure for a vocabulary of D words needs D word instances at the first level and D^2 word instances at the second level
  - Total of D(D+1) word models must be instantiated
  - Other, more expensive structures are also possible

- An N-gram decoding structure will need
  - $D + D^2 + D^3 \ldots D^{N-1}$ word instances
  - Arcs must be incorporated such that the exit from a word instance in the $(N-1)^{th}$ level always represents a word sequence with the same trailing sequence of N-1 words

To Build a Speech Recognizer

- Train word HMMs from many training instances
  - Typically one trains HMMs for individual phonemes, then concatenates them to make HMMs for words
  - Recognition, however is almost always done with WORD HMMs (and not phonemes as is often misunderstood)
- Train or decide a language model for the task
  - Either a simple grammar or an N-gram model
- Represent the language model as a compact graph
- Introduce the appropriate HMM for each word in the graph to build a giant HMM

- Use the Viterbi algorithm to find the best state sequence (and thereby the best word sequence) through the graph!