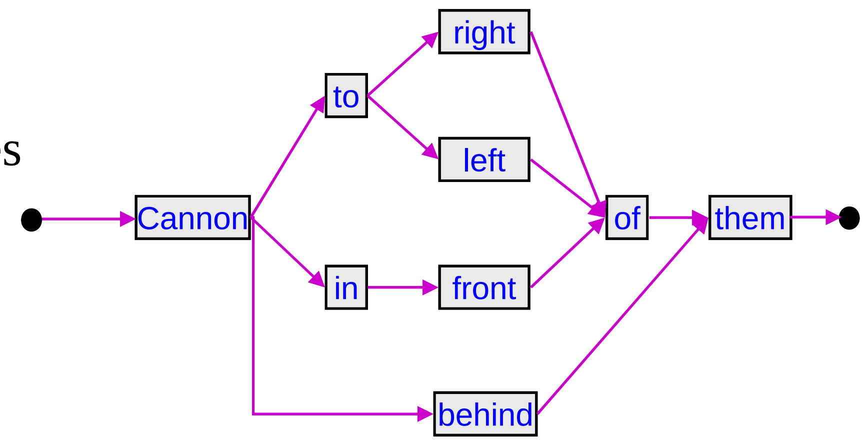
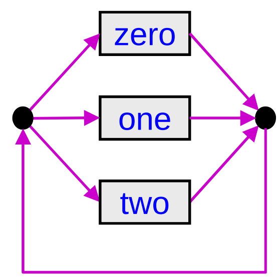
Exact and Approximate Search for Automatic Speech Recognition

Types of "Language Models"

- Finite state grammars
  - The set of all possible word sequences is represented as a graph



- Context free grammars
  - A set of context-free rules:
    - Digit := 0 | 1 | 2;
    - Number = Digit | Number Digit;
  - Typically converted into a finite state graph for recognition
  - Graph may be approximate
    - Some CFGs are not representible as finite-state Graphs and require push-down automata

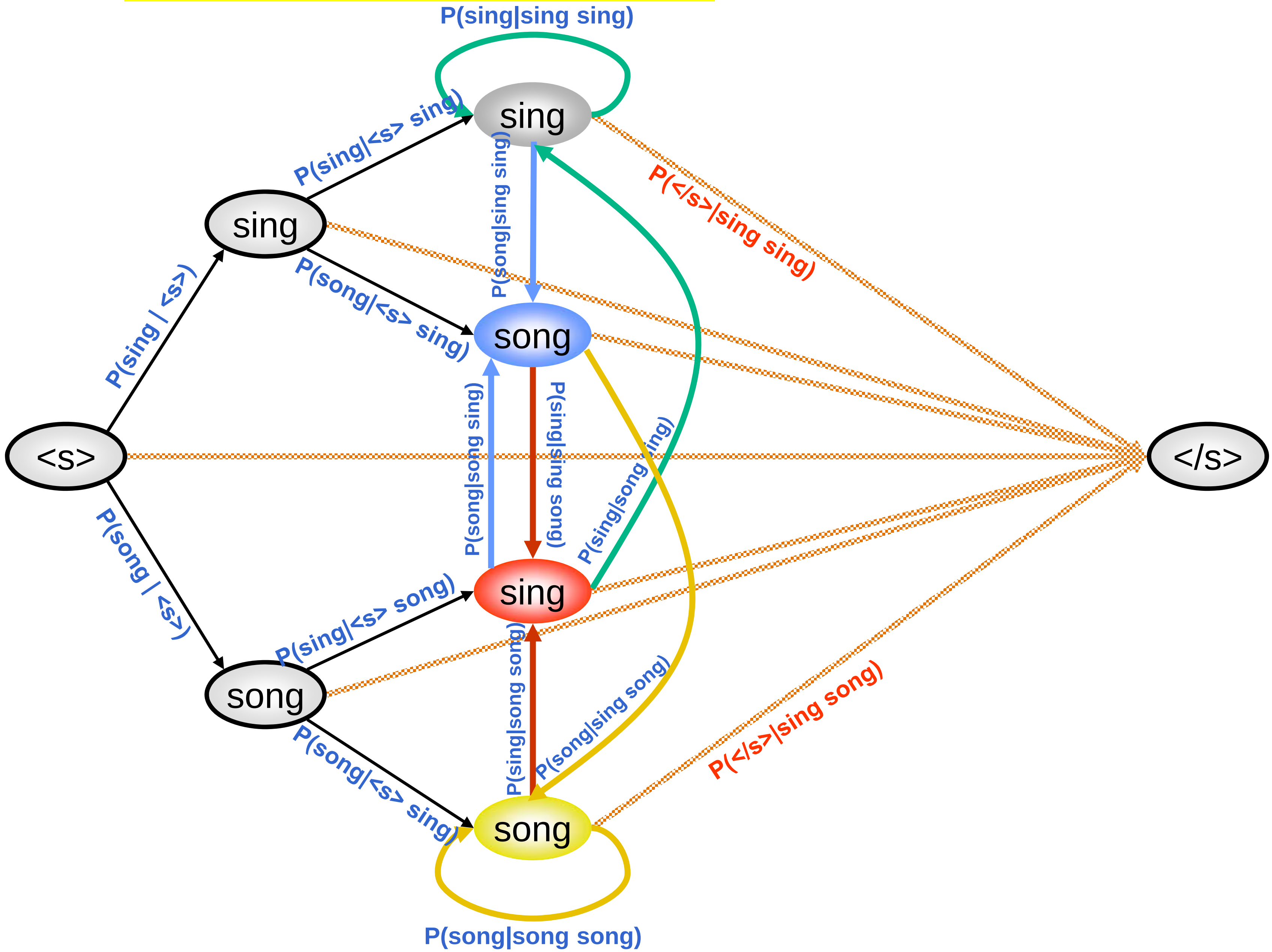


- N-gram language models

An Example Backoff Trigram LM

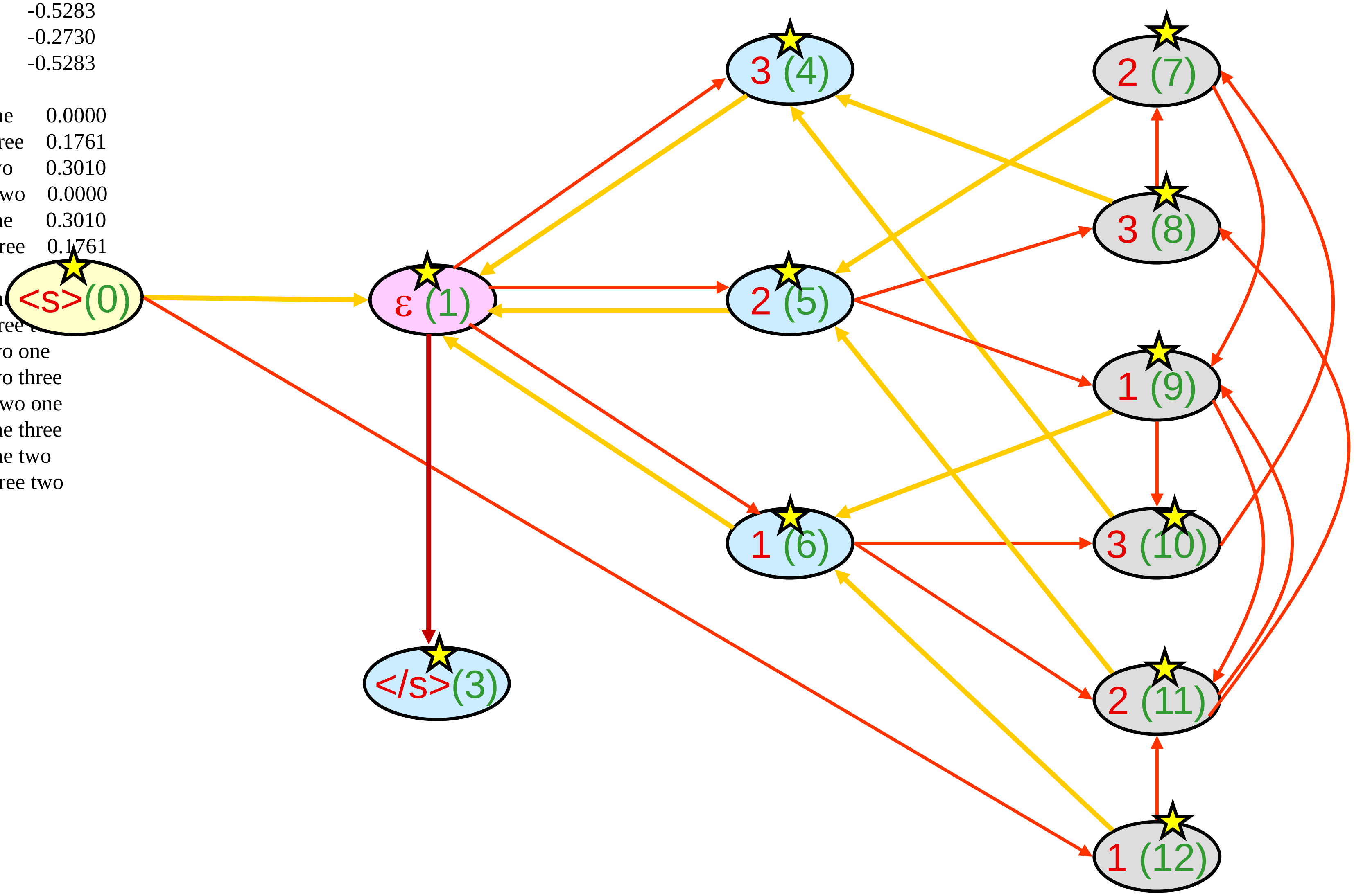
```
\1-grams:
-1.2041 <UNK>      0.0000
-1.2041 </s>       0.0000
-1.2041 <s>       -0.2730
-0.4260 one       -0.5283
-1.2041 three     -0.2730
-0.4260 two       -0.5283
\2-grams:
-0.1761 <s> one      0.0000
-0.4771 one three    0.1761
-0.3010 one two      0.3010
-0.1761 three two    0.0000
-0.3010 two one      0.3010
-0.4771 two three    0.1761
\3-grams:
-0.3010 <s> one two
-0.3010 one three two
-0.4771 one two one
-0.4771 one two three
-0.3010 three two one
-0.4771 two one three
-0.4771 two one two
-0.3010 two three two
```

A COMPLETE TRIGRAM GRAPH



A “Reduced” Trigram Graph

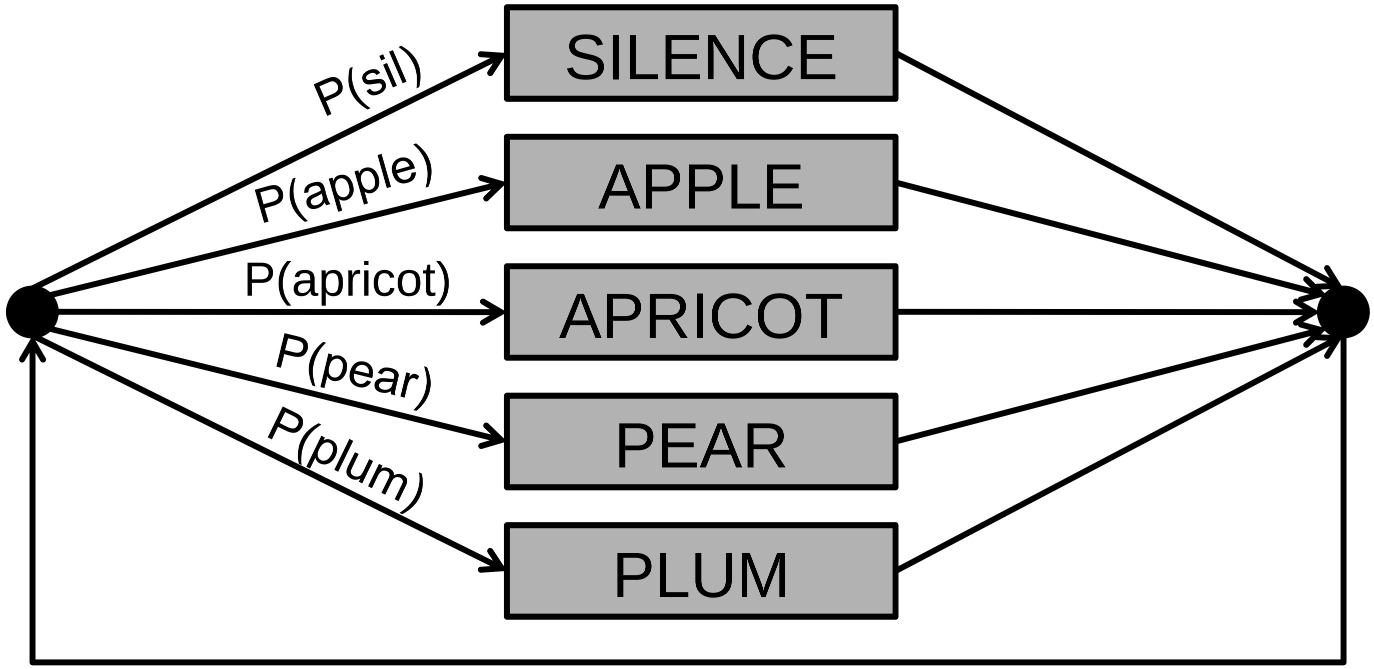
\1-grams:
-1.2041 <UNK> 0.0000
-1.2041 </s> 0.0000
-1.2041 <s> -0.2730
-0.4260 one -0.5283
-1.2041 three -0.2730
-0.4260 two -0.5283
\2-grams:
-0.1761 <s> one 0.0000
-0.4771 one three 0.1761
-0.3010 one two 0.3010
-0.1761 three two 0.0000
-0.3010 two one 0.3010
-0.4771 two three 0.1761
\3-grams:
-0.3010 <s> one
-0.3010 one three t
-0.4771 one two one
-0.4771 one two three
-0.3010 three two one
-0.4771 two one three
-0.4771 two one two
-0.3010 two three two



Ngrams: Can we do better

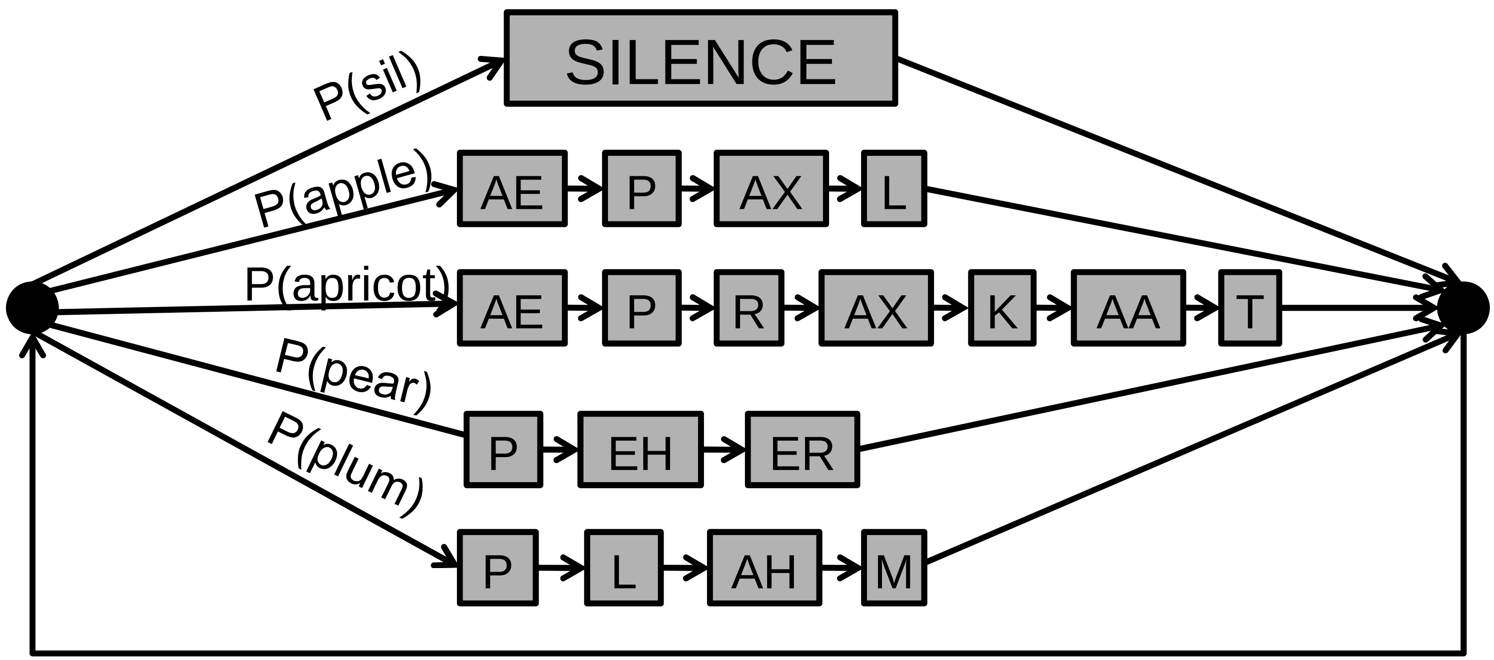
- Even reduced graphs can get very large
  - Rarely directly used for recognition
- Alternate strategies must be employed
  - Lextrees
    - For low-order Ngrams only
  - Approximate decoding strategies
  - Lextrees + approximate decoding strategies
- Minimization strategies
  - WFSTs: Using techniques from finite state automata theory

A Unigram Graph



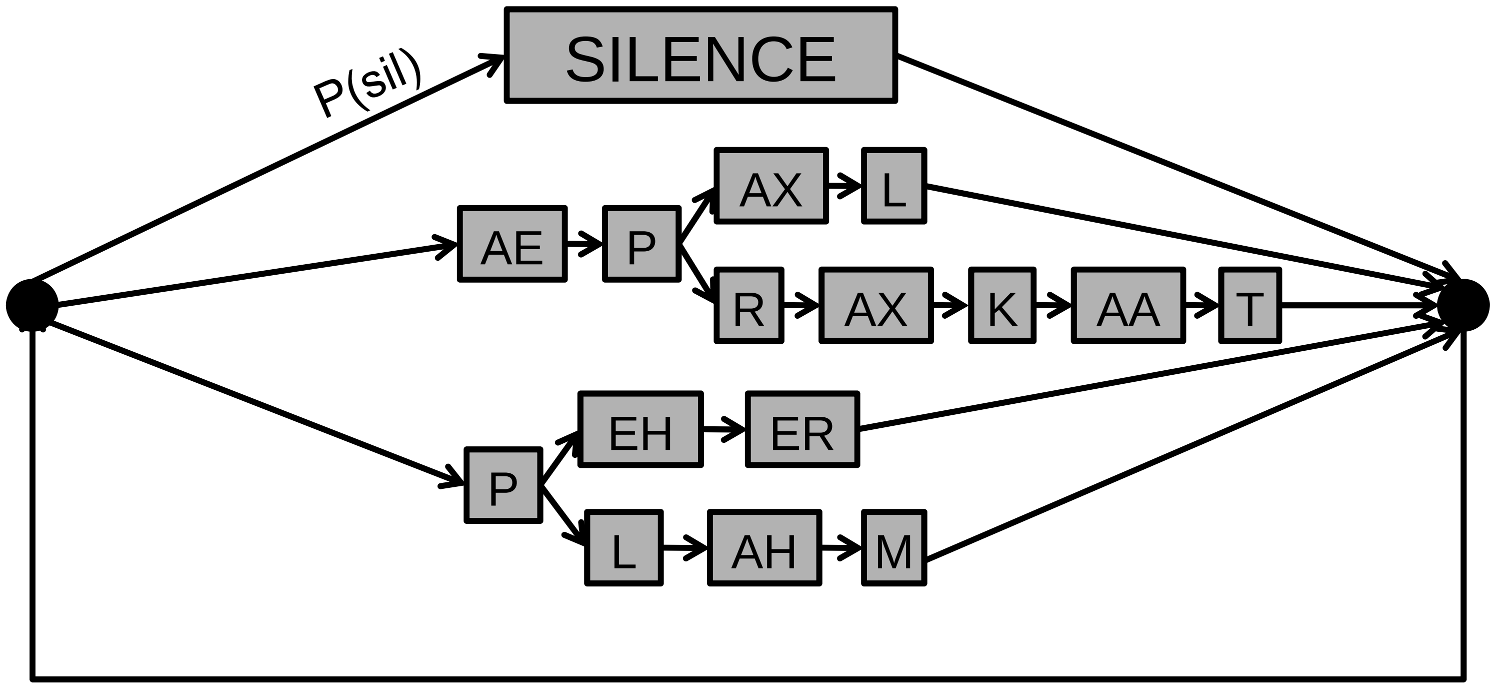
- Just a set of parallel word models with a loopback
- The ingoing edge into each word carries its LM probability

A Unigram Graph with words built from phonemes



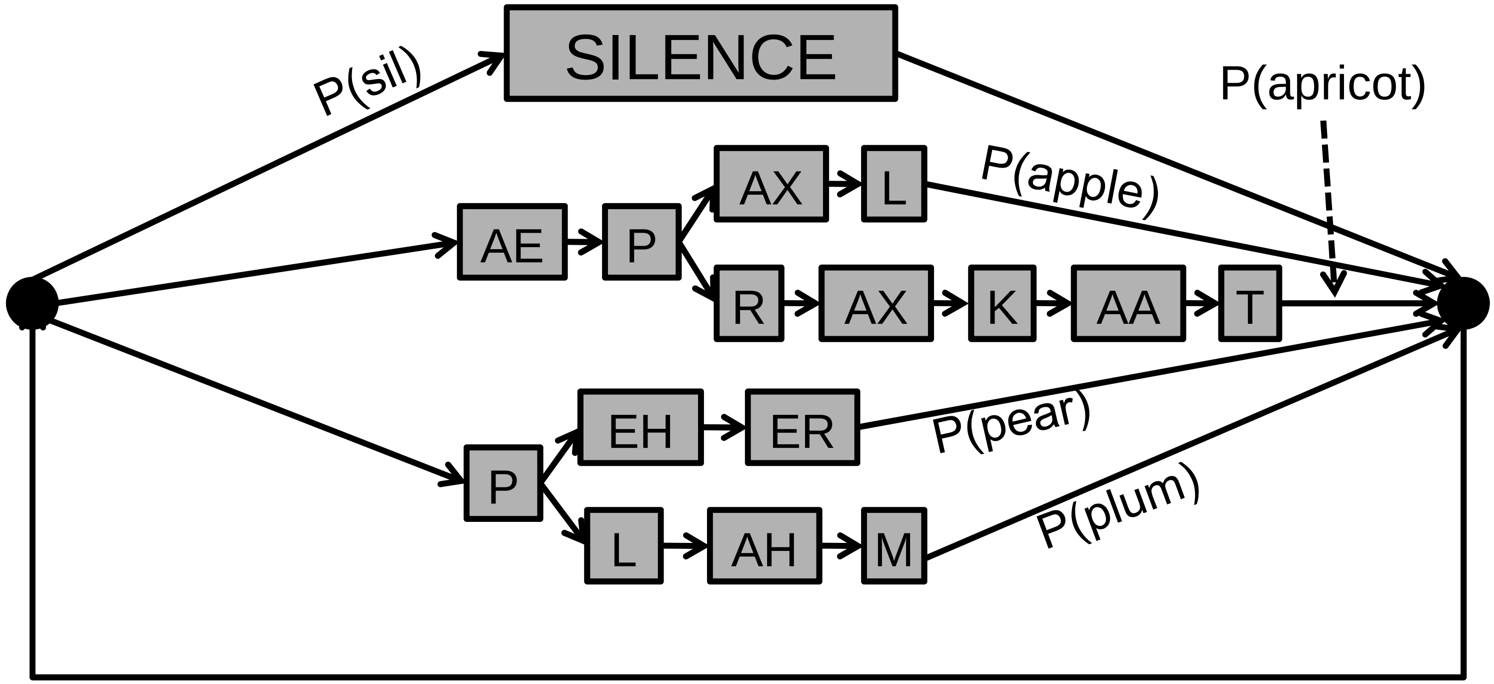
- Composing Word models from phoneme models
- Each rectangle is actually an HMM. The entire graph is a large HMM

A Unigram Lextree



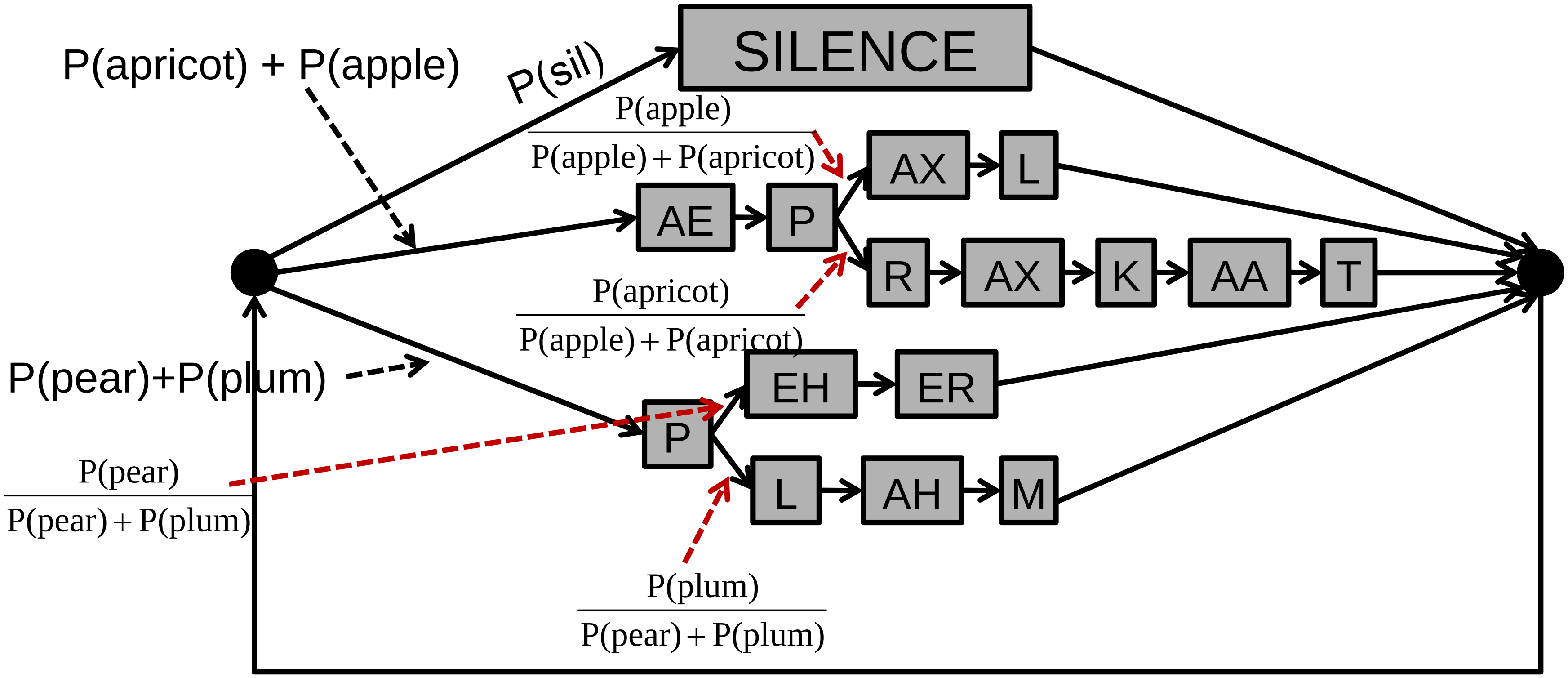
- Eliminate redundancy in the graph
- But where do word probabilities get introduced?
  - The identity of the word is not evident at entry!

A Unigram Lextree with trailing probabilities



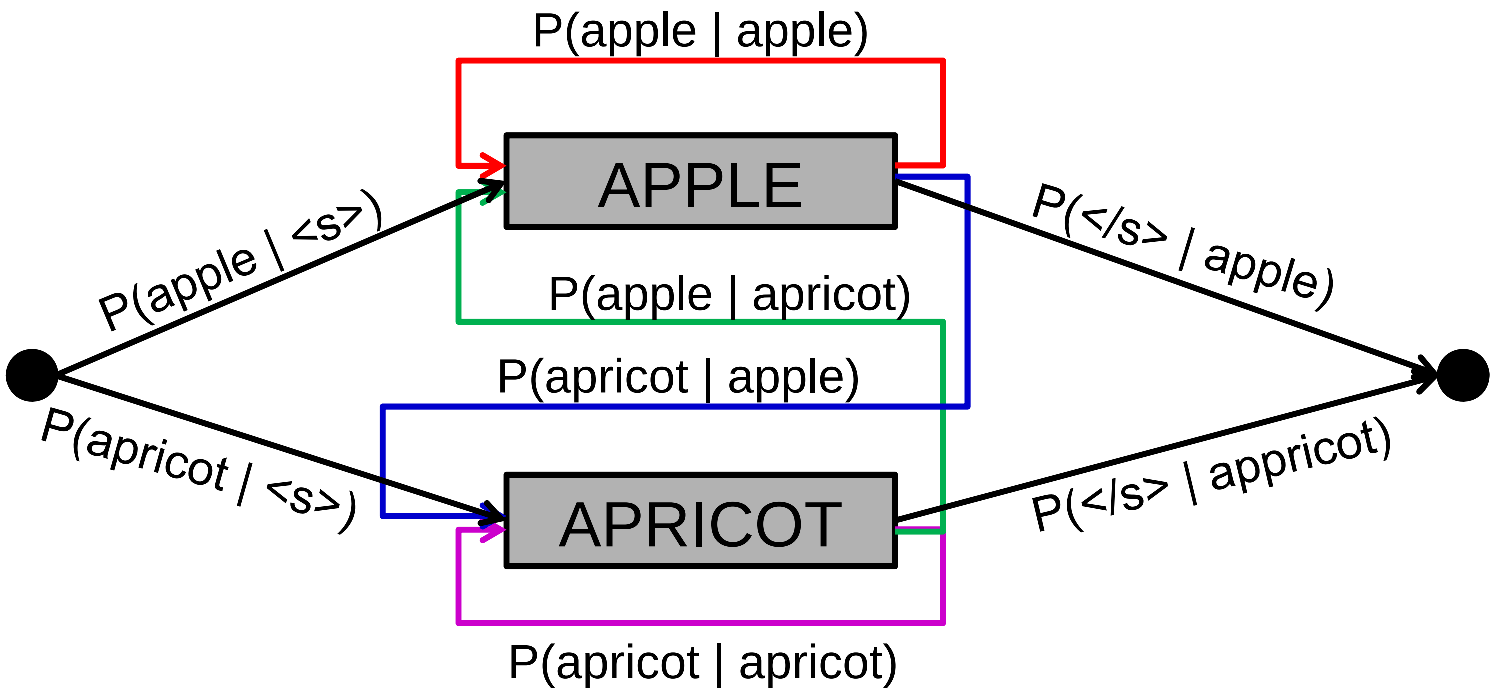
- Introduce word probabilities on the *exit* arcs
  - The word identity is evident at that point

A Unigram Lextree with spread probabilities



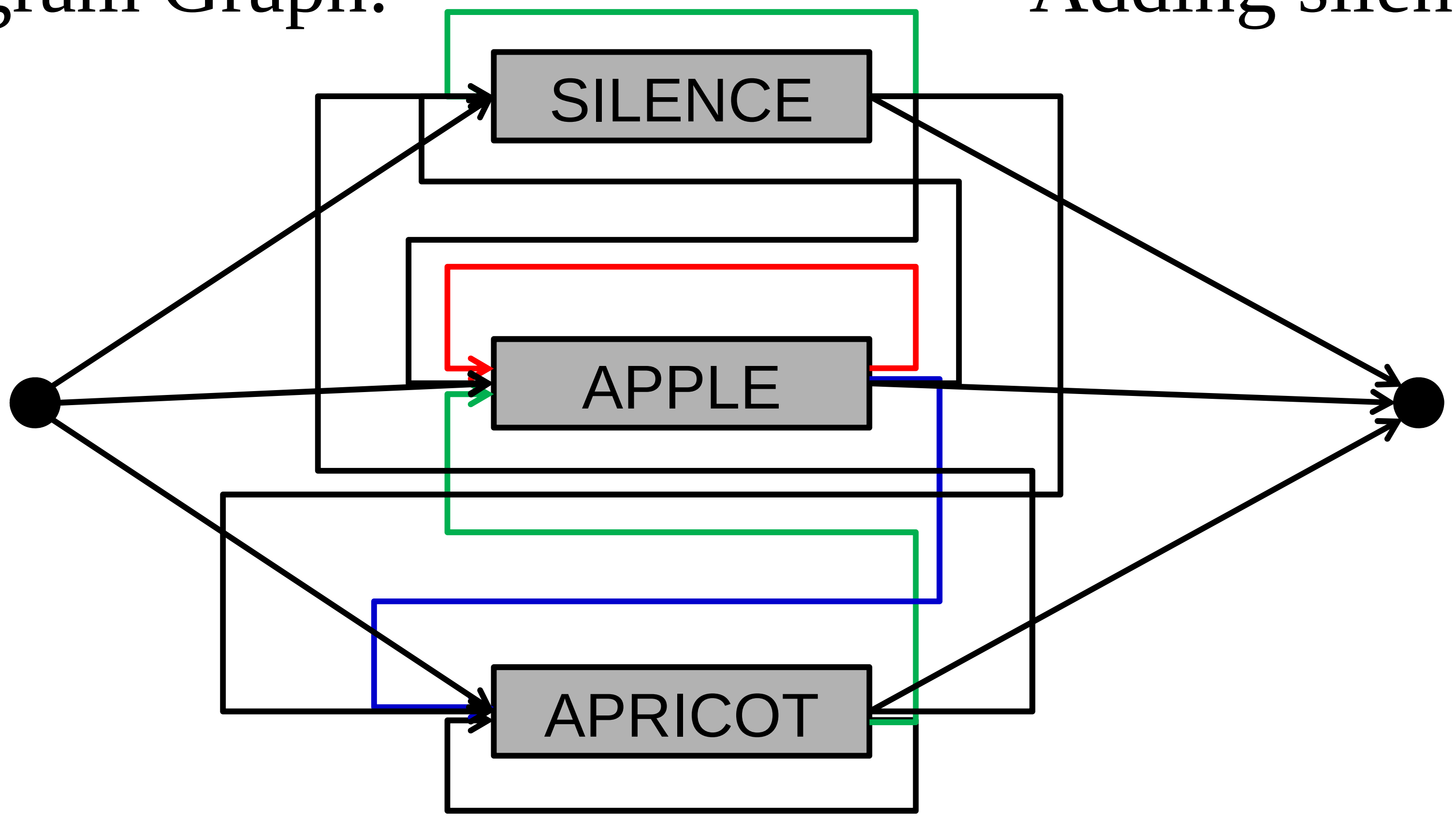
- Better still: Spread the probabilities
  - Any arc that first identifies a subset of words carries the conditional probability of that subset

A Bigram Graph



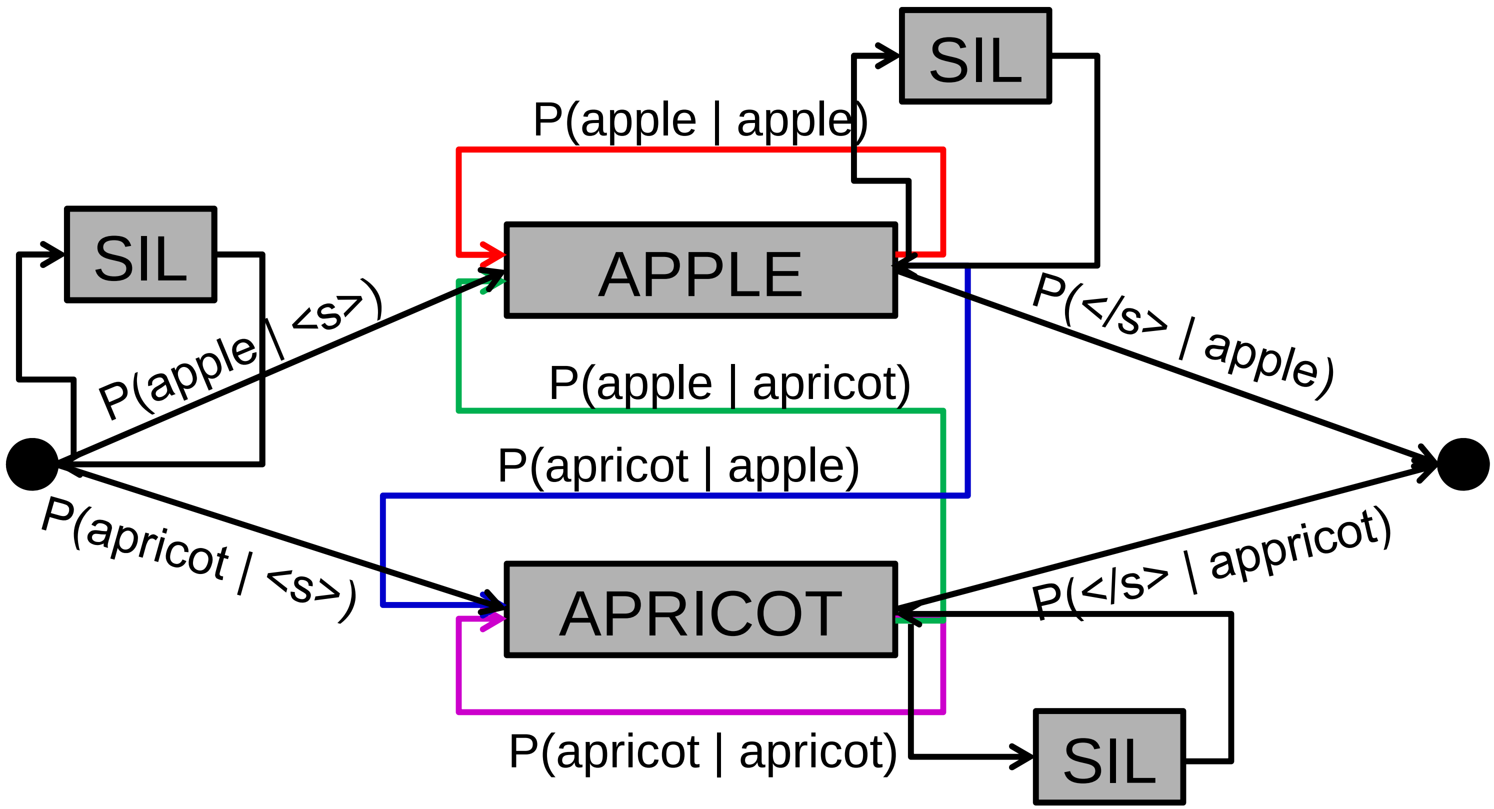
- Explicit connection from every word to every word
  - Connections carry bigram probabilities

A Bigram Graph: Adding silence



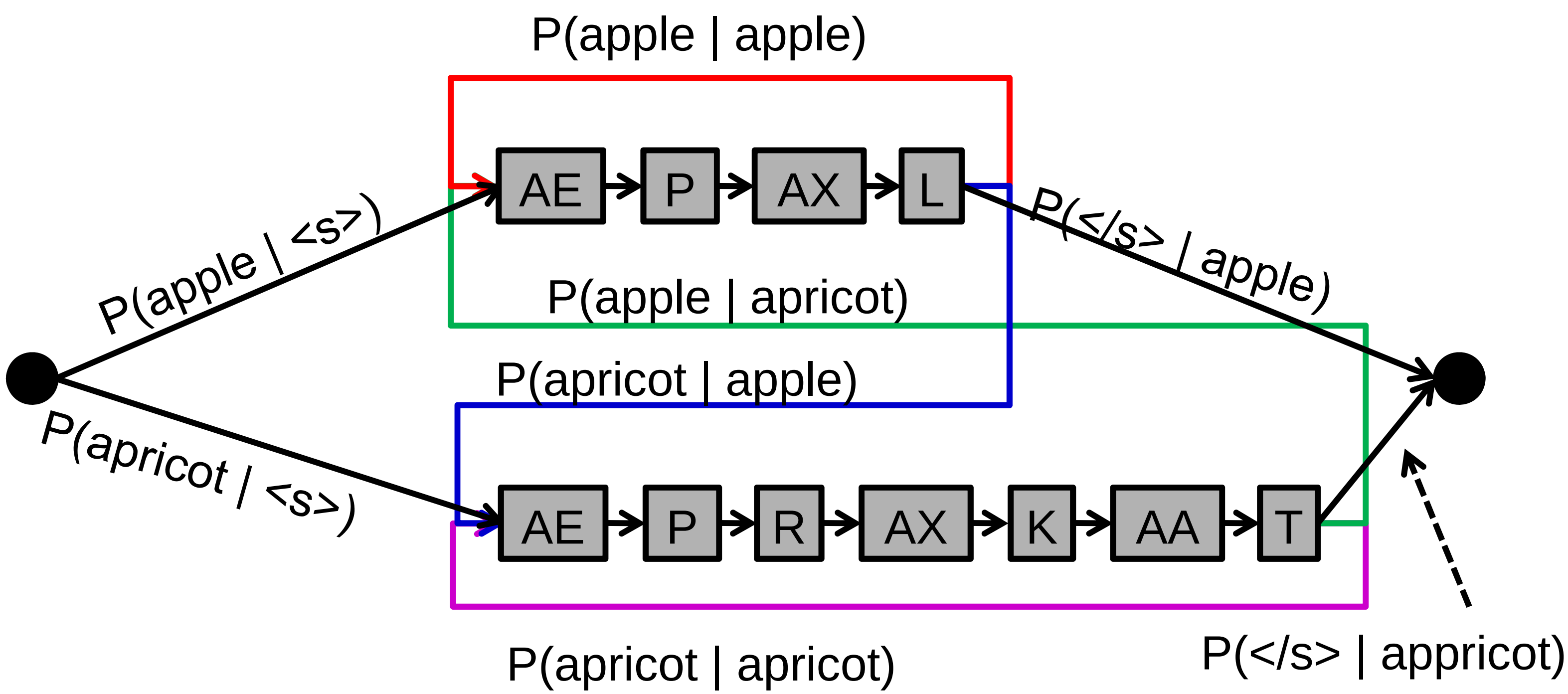
- Addition of looping silence is non-trivial
  - What will the probability be on the outgoing edges from silence
    - We do not have probabilities for P(word | silence), only P(word|word)
    - If a silence occurs between two words, we use the word before the silence as context

A Bigram Graph: Proper insertion of silences



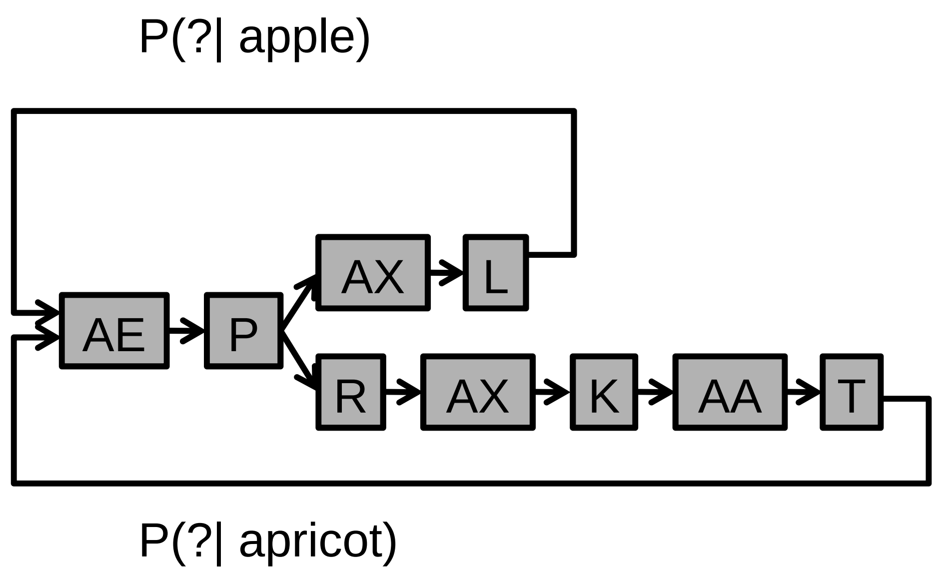
- An explicit silence model at the end of every word
  - We get an enormous number of copies of the silence model!

What about Lextrees



- Can this be collapsed to a lextree?

Probabilities on lextrees



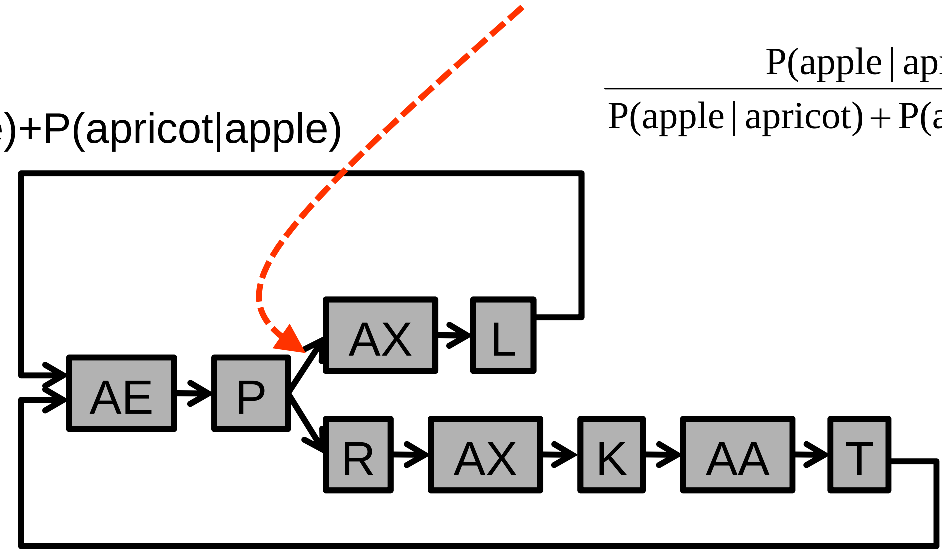
- Word identities are not known on entry
  - Only on word exit

Probabilities on lextrees

$$\frac{P(apple \mid apple)}{P(apple \mid apple) + P(apricot \mid apple)}$$

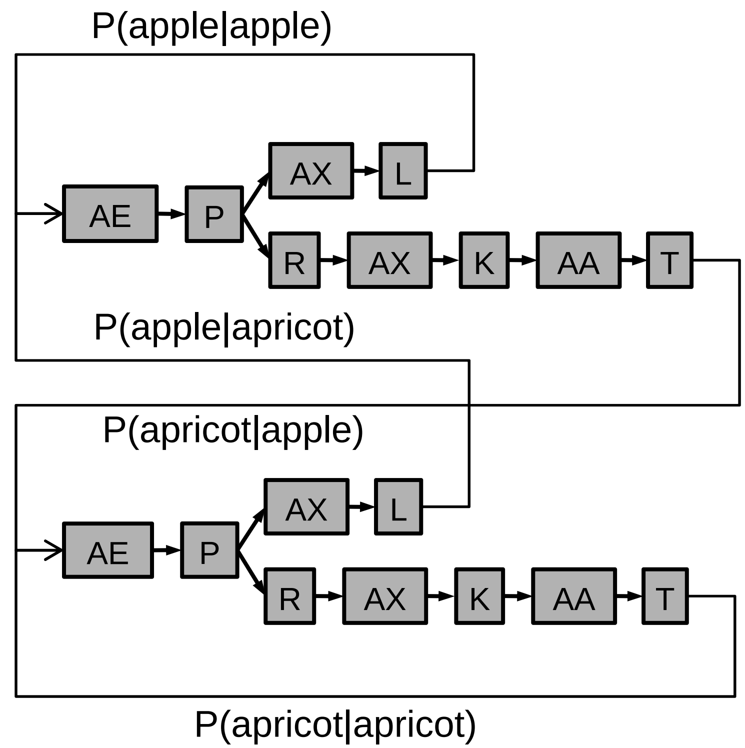
OR ?

$$\frac{P(apple \mid apricot)}{P(apple \mid apricot) + P(apricot \mid apricot)}$$



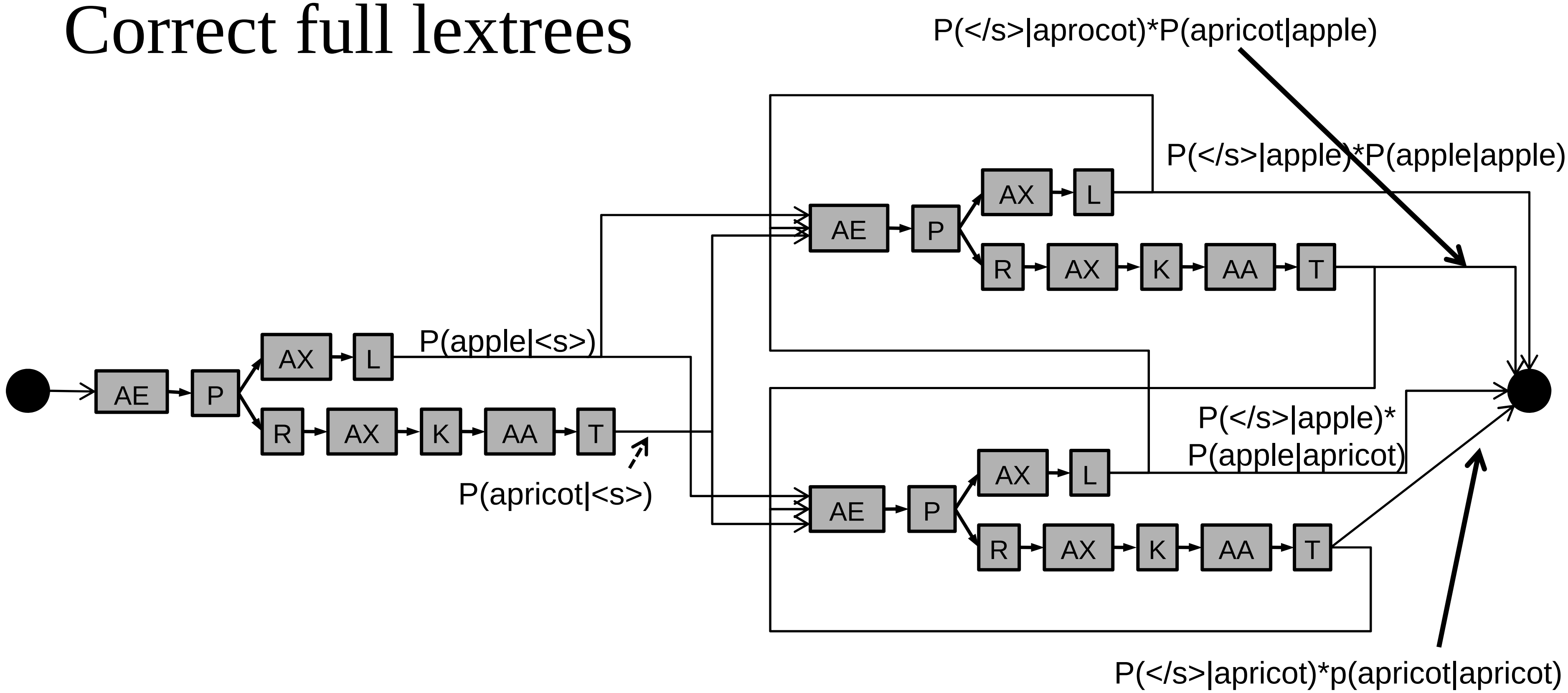
- Word identities are not known on entry
  - Only on word exit
- Word probabilities cannot be smeared
  - Both word histories lead into the same node
  - Uncertain which probability terms to use on inner connections

Correct Lextrees



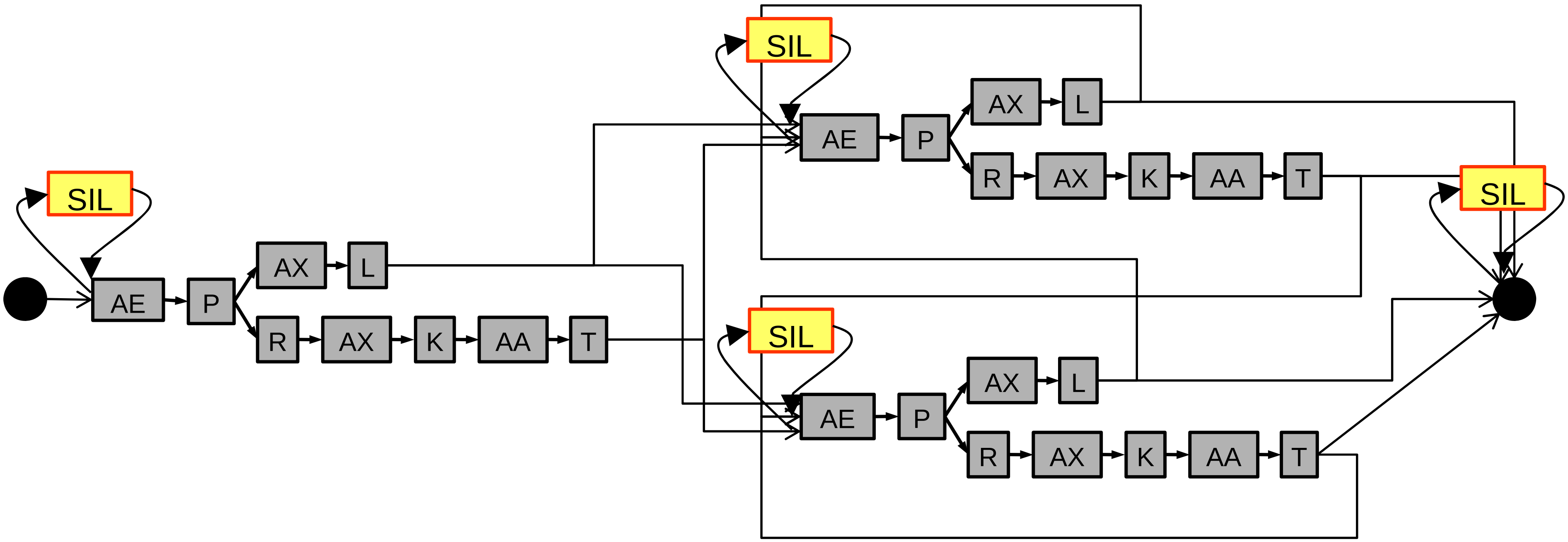
- Each edge carries the bigram probability of the *exited* word
  - This is different from the "flat" structure where the edges carried probabilities of words to be entered
  - All "Apple" exits enter lextree 1, all "apricot" exits enter lextree 2
- This graph is not complete: it ignores the first word in a sentence

Correct full lextrees



- The word entry bigrams need their own lextree!
  - Since neither of the second-level lextrees can represent a sentence-beginning context
    - Lextree 1 represents the “Apple” context (only exits from the word “apple” enter this lextree
    - Lextree 2 is the “apricot” context
- Why do transitions into the end of sentence have *products* of two probability terms?

Correct full lextrees with silence



- Fortunately, adding silence doesn't complicate this too much
- Add a looping silence at the beginning of each lextree
  - And one at the sentence terminator

Correct Structures are Limiting

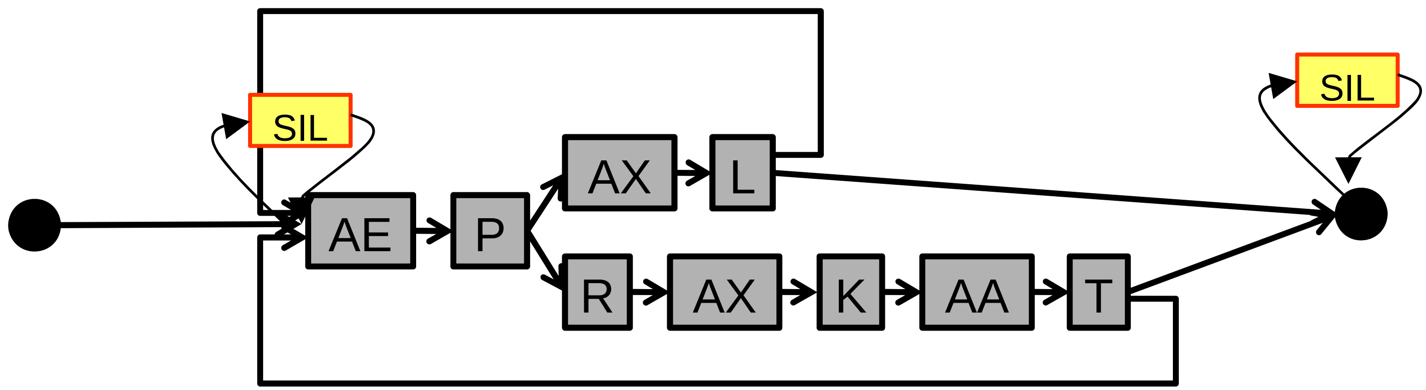
- The “correct” flat N-gram structure can get very large
  - $D + D^2 + .. + D^{N-1}$ word HMMs are required in the larger “Language” HMM
- Even the reduced N-gram structure can be very large
  - Reduced structures are not exact
    - Multiple paths exist for each N-gram
  - Reduced structures are nevertheless used very effectively by WFST-based strategies

- Lextrees result in significant compression for Unigram LMs
- But for N-gram LMs “correct” Lextree-based graphs are much larger than “flat” graphs
  - Need $D + D^2 + .. + D^{N-1}$ lextrees!!

Approximate Search Strategies

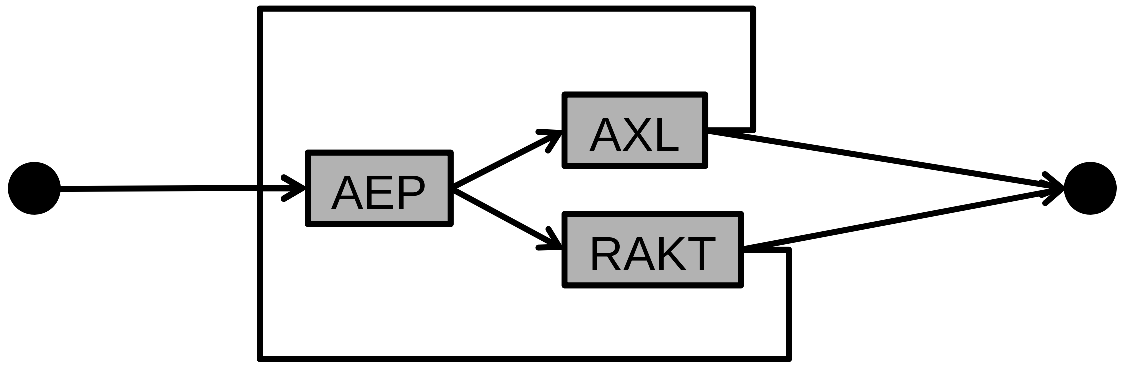
- Approximate search strategies are not guaranteed to result in the best recognition
  - Although, in practice they often approach the optimal recognition
- Efficiency is obtained by dynamically modifying graph parameters
  - I.e. language probabilities in the language HMM
- This is typically done by utilizing word histories
  - From a backpointer table
- The resulting improvement in efficiency can be very very large

Approximate search with a Unigram Lextree



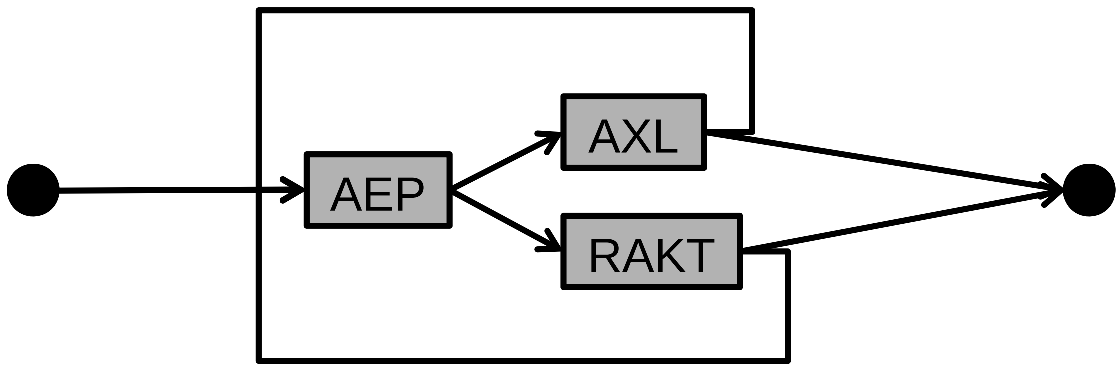
- Utilize the above lextree as the basic HMM structure
  - Note – no language model probabilities are loaded on the lextree
    - These will be imposed dynamically during search
  - In practice unigram probabilities may be factored into the lextree and factored out during search
    - We will ignore this option in the following explanation

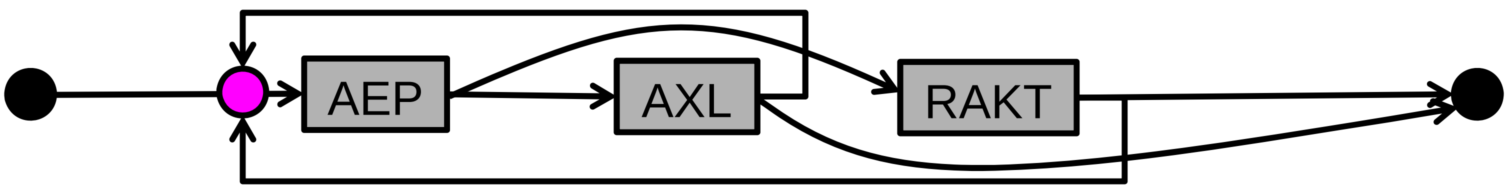
Approximate search with a Unigram Lextree



- We will use the simplified figure above in the following explanation
  - AEP is the concatenation of AE and P
  - AXL is the concatenation of AX and L
  - RAKT is the concatenation of R AX K AA and T
- Will not explicitly show silence models

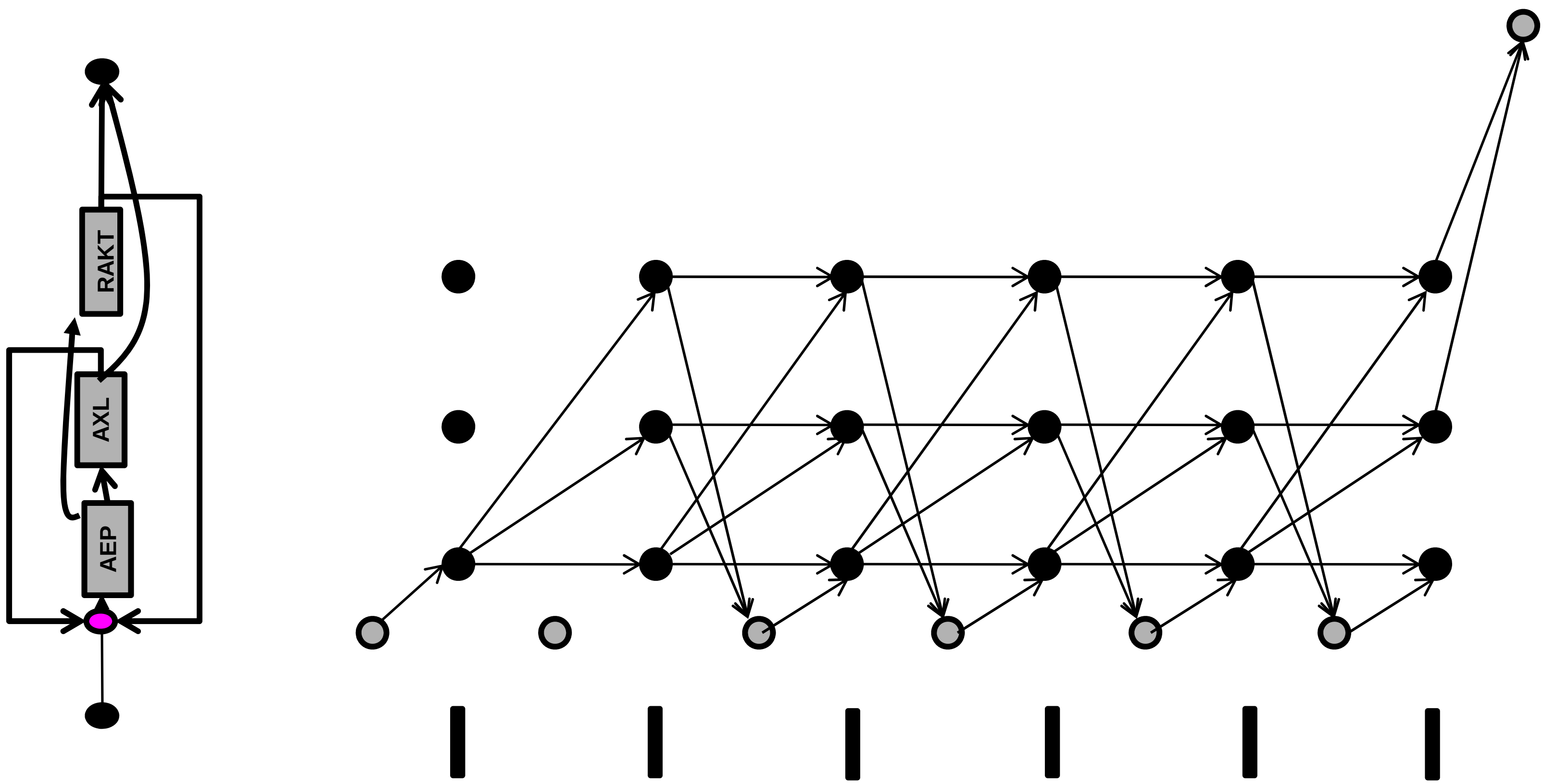
Approximate search with a Unigram Lextree





- A Linear Representation that is useful to draw a trellis
  - Note: Each box is actually an HMM (representing a sequence of phonemes)
  - For simplicity we will assume each box has only one state

Approximate search Trellis

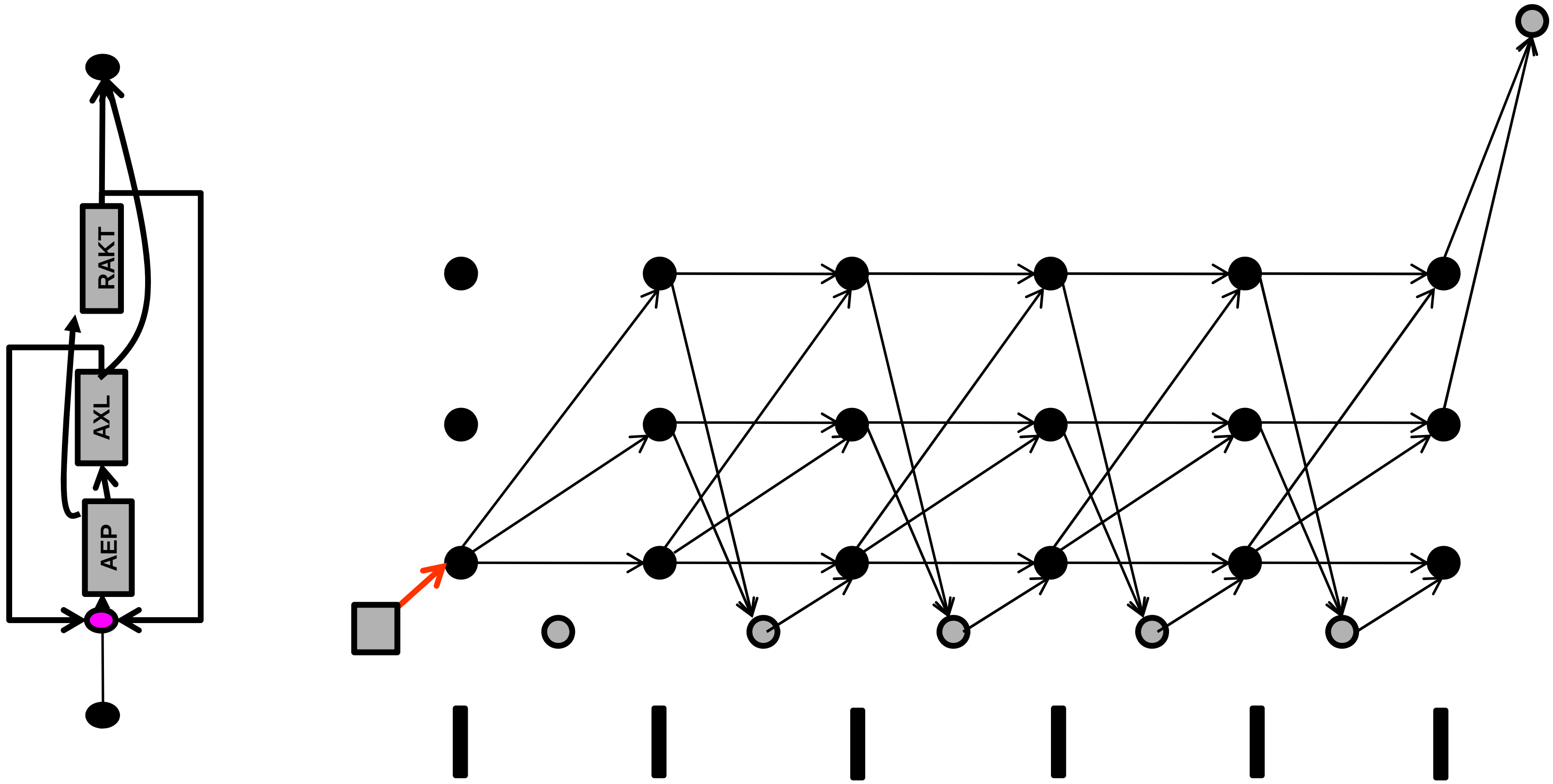


◆ A normal unigram trellis, but with no LM probabilities at word transitions

Id,time,parent,score,word

0,0,-1,0,<s>

Approximate search Trellis

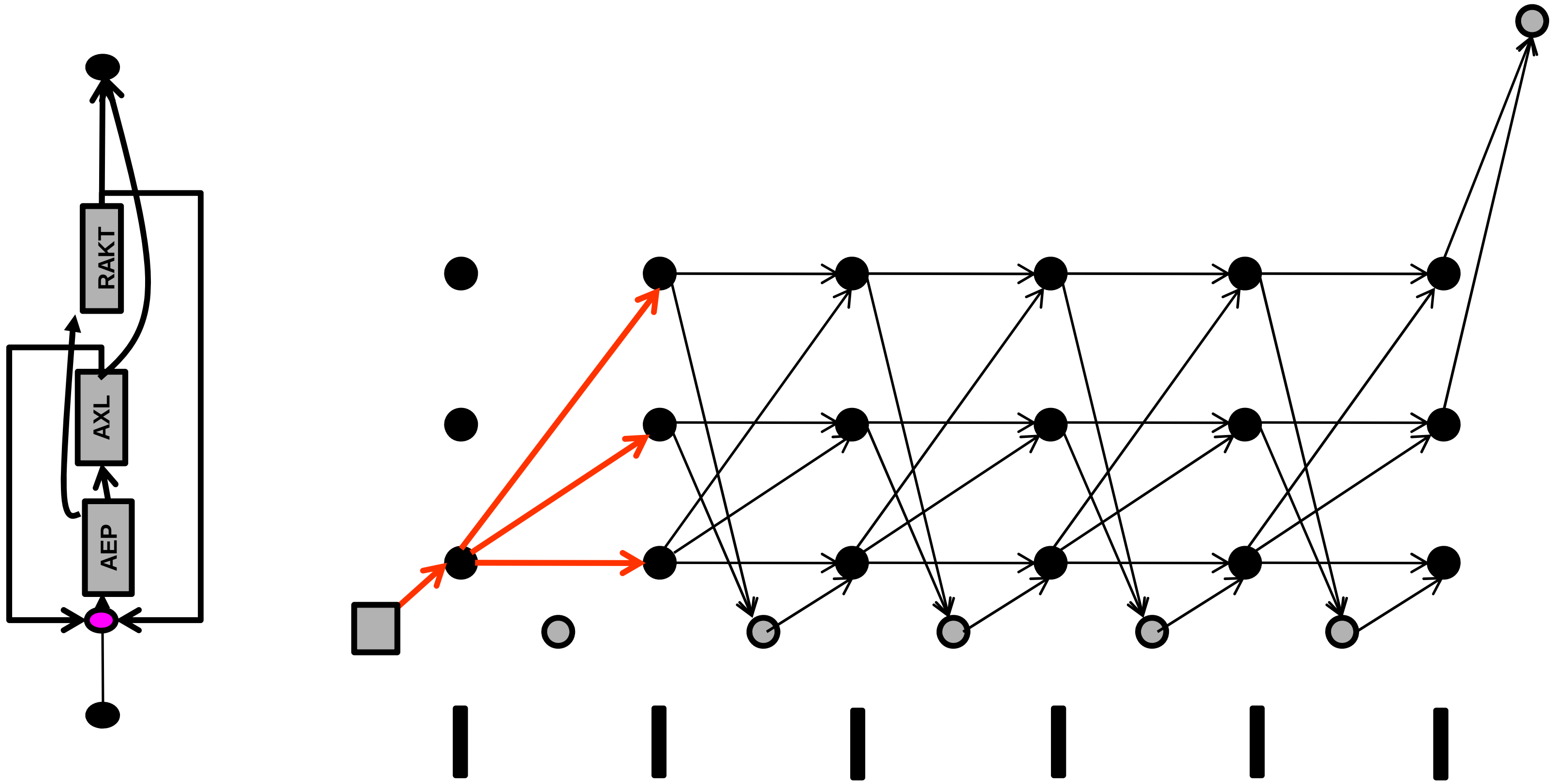


- A normal unigram trellis, but with no LM probabilities at word transitions

Id,time,parent,score,word

0,0,-1,0,<s>

Approximate search Trellis

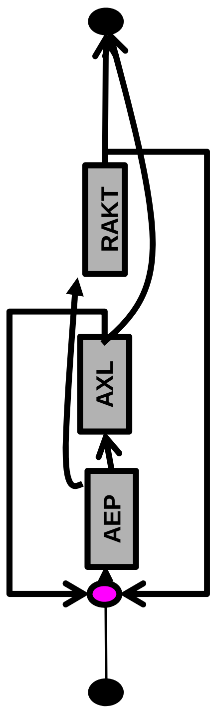


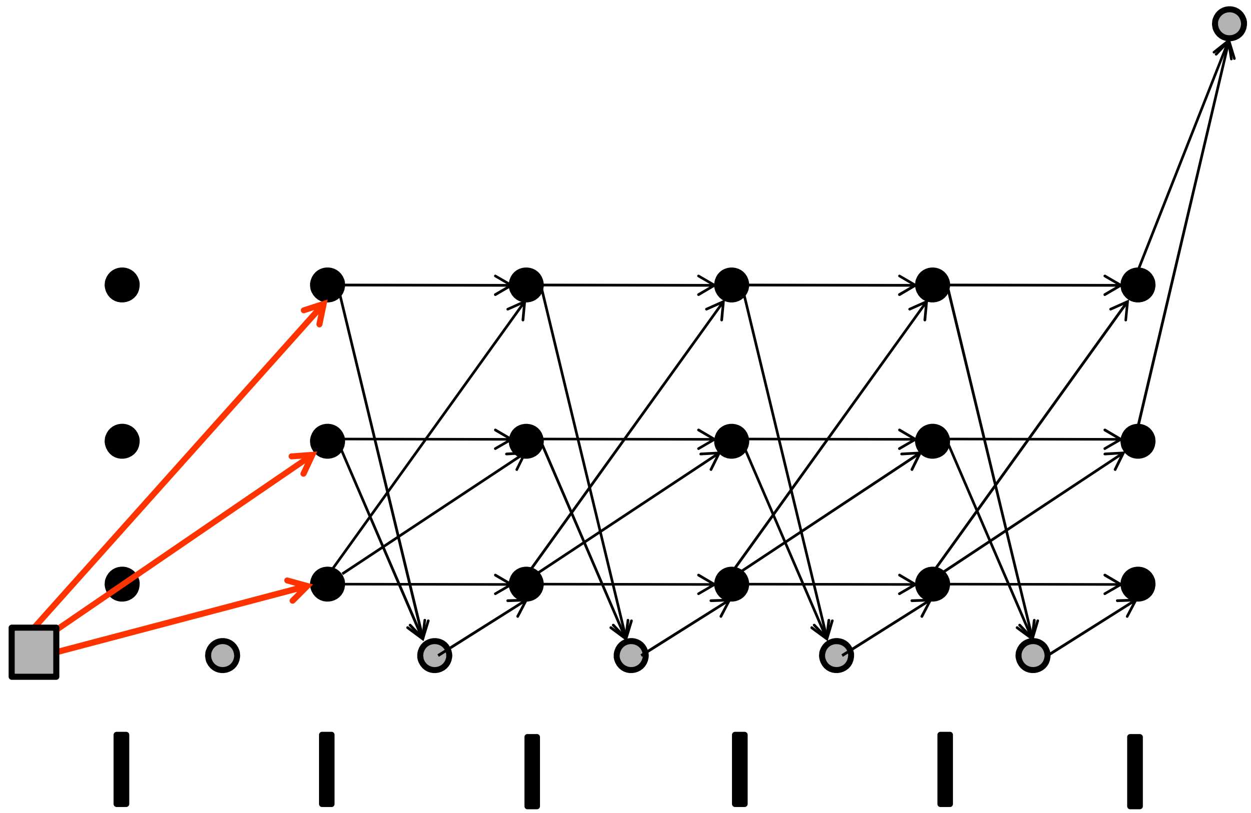
- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

Id,time,parent,score,word

0,0,-1,0,<s>

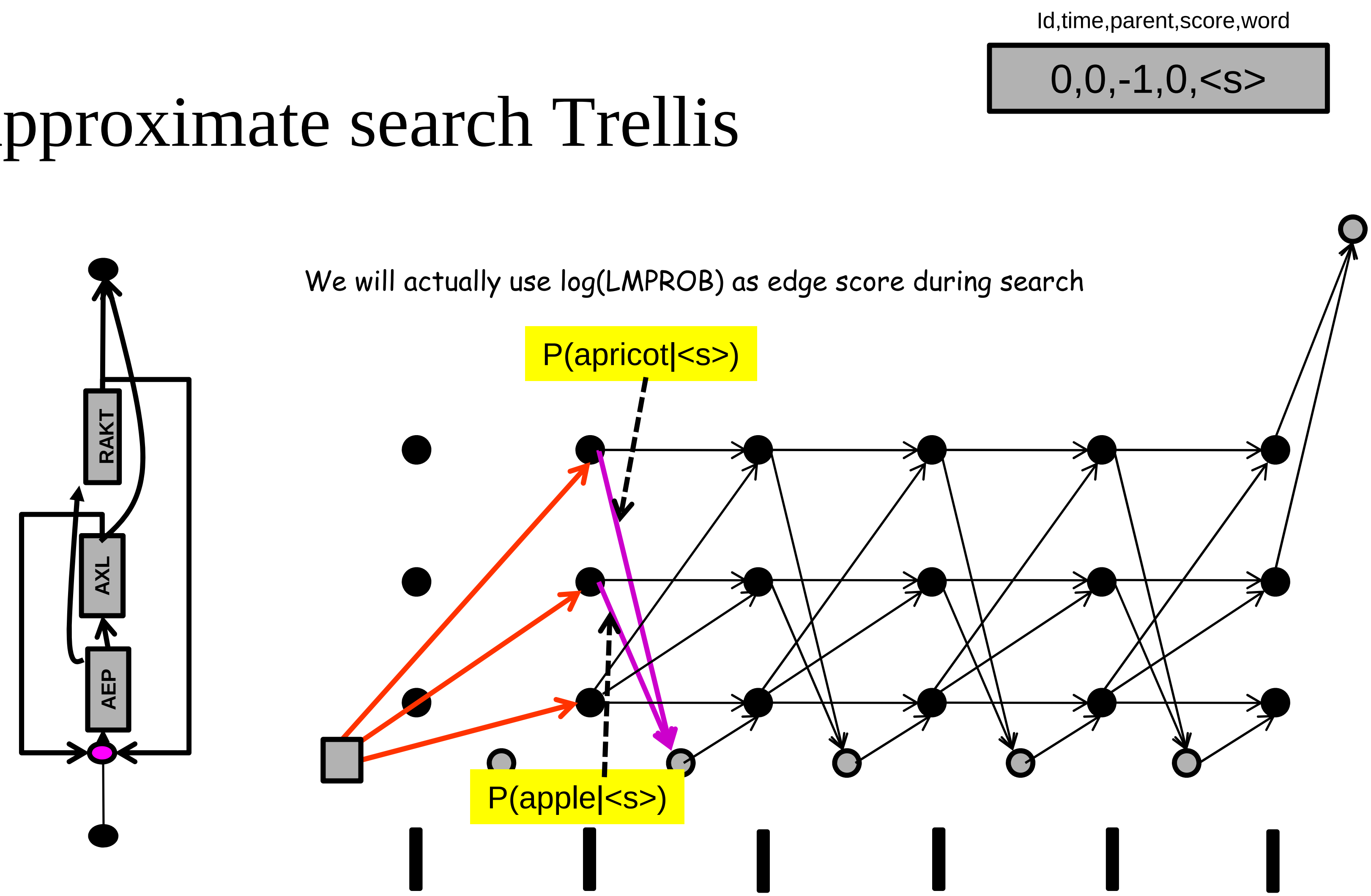
Approximate search Trellis





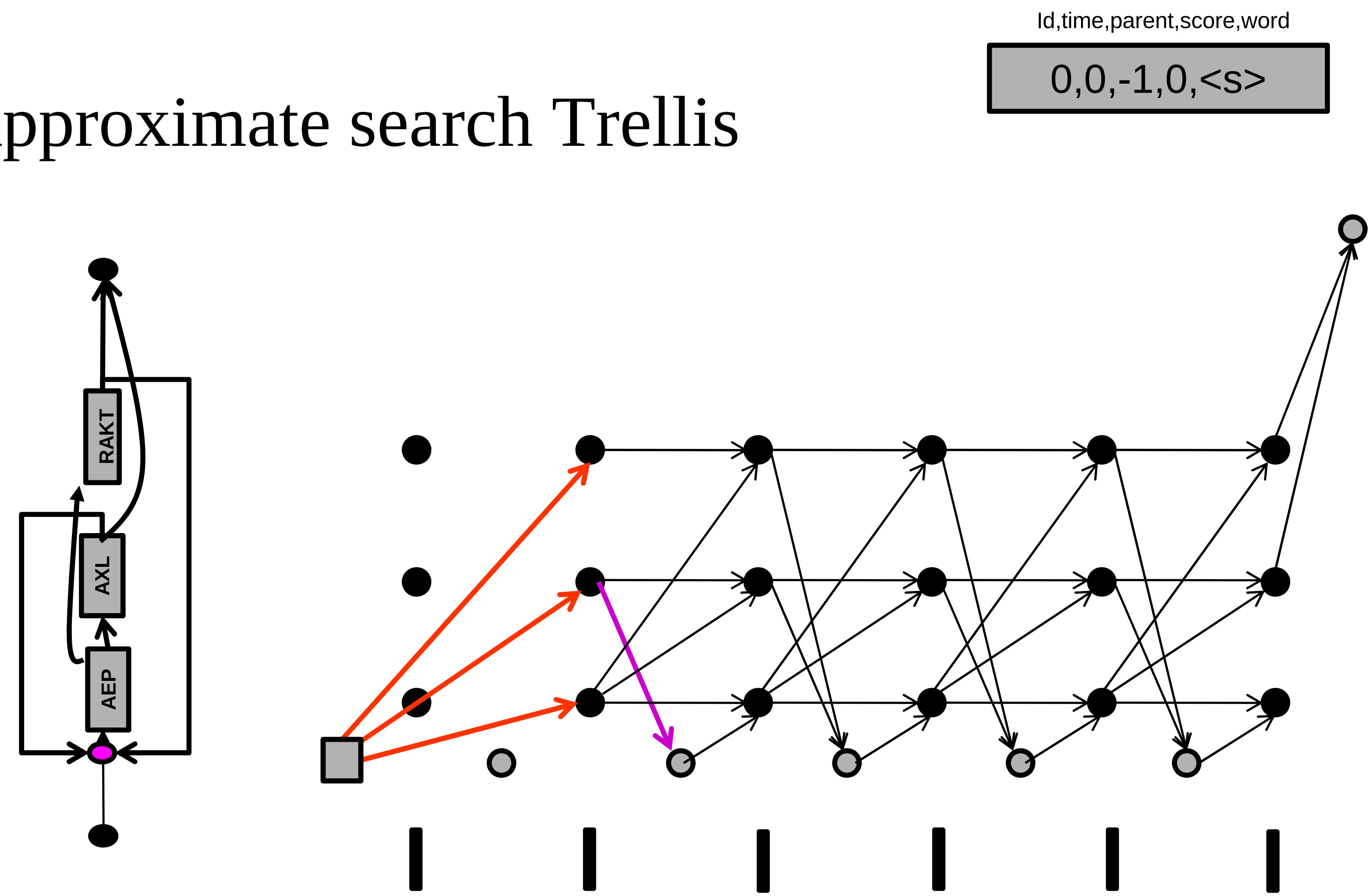
- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

Approximate search Trellis



- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

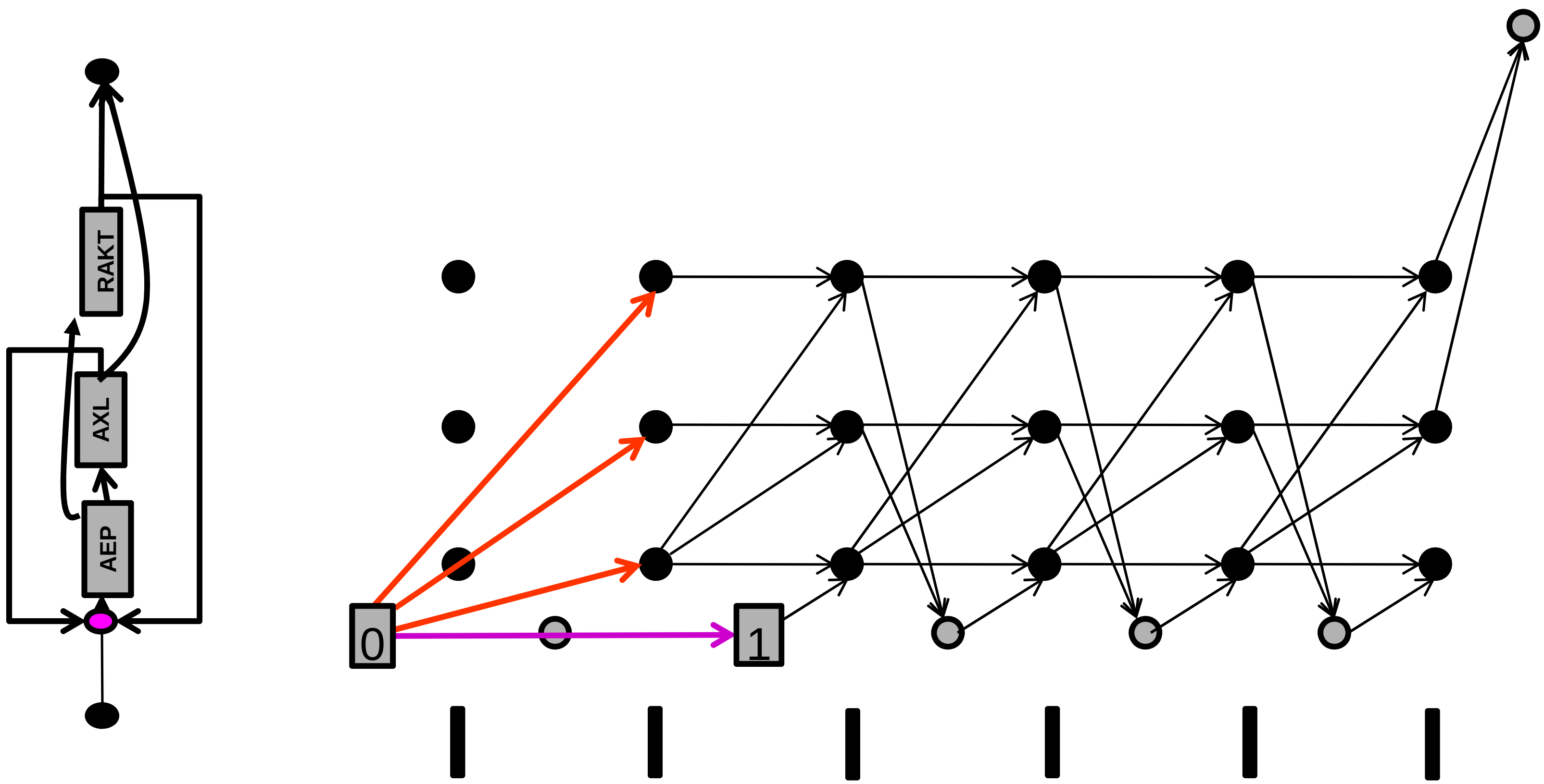
Approximate search Trellis



- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

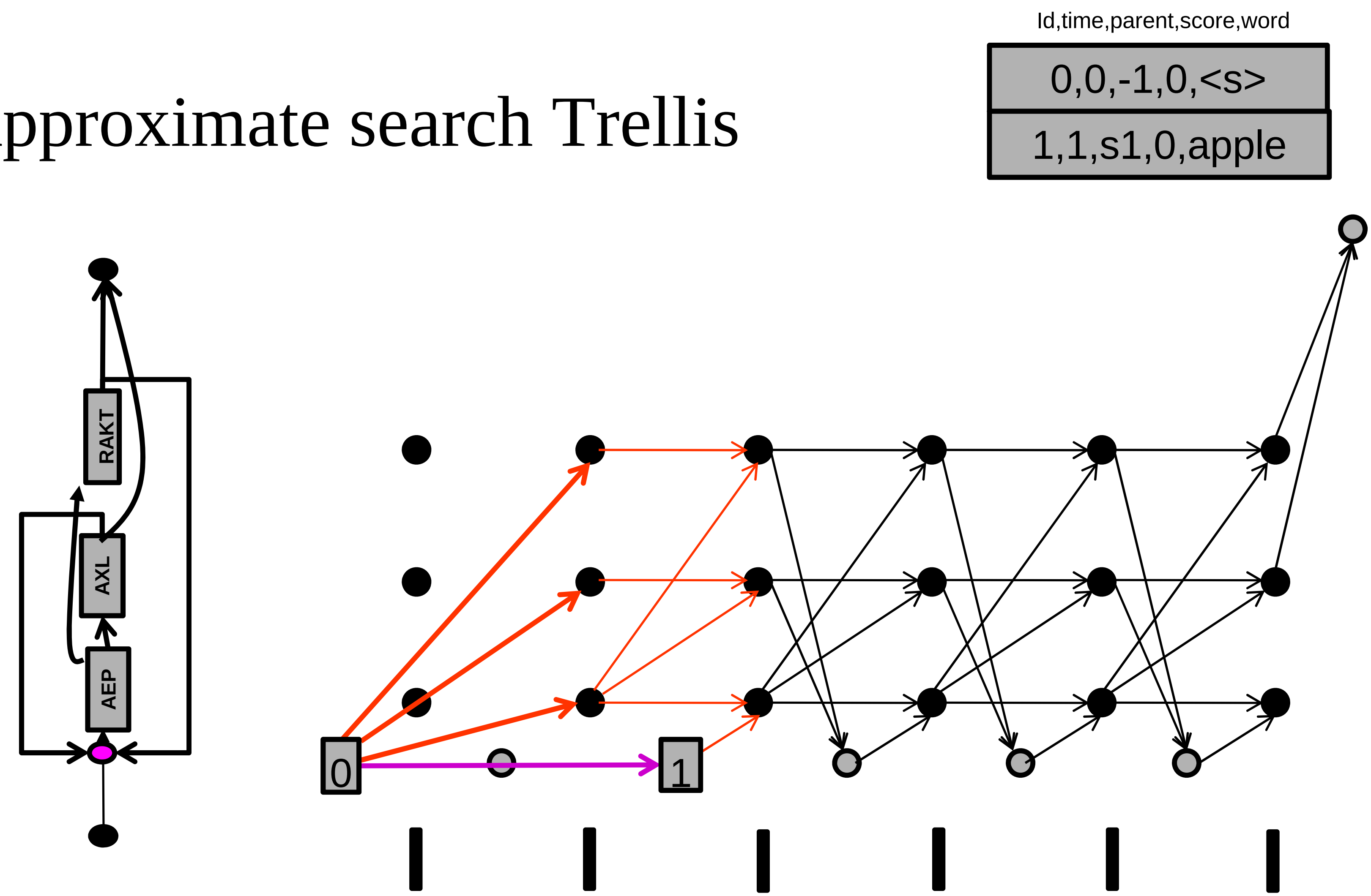
Approximate search Trellis

Id,time,parent,score,word
0,0,-1,0,<s>
1,1,s1,0,apple



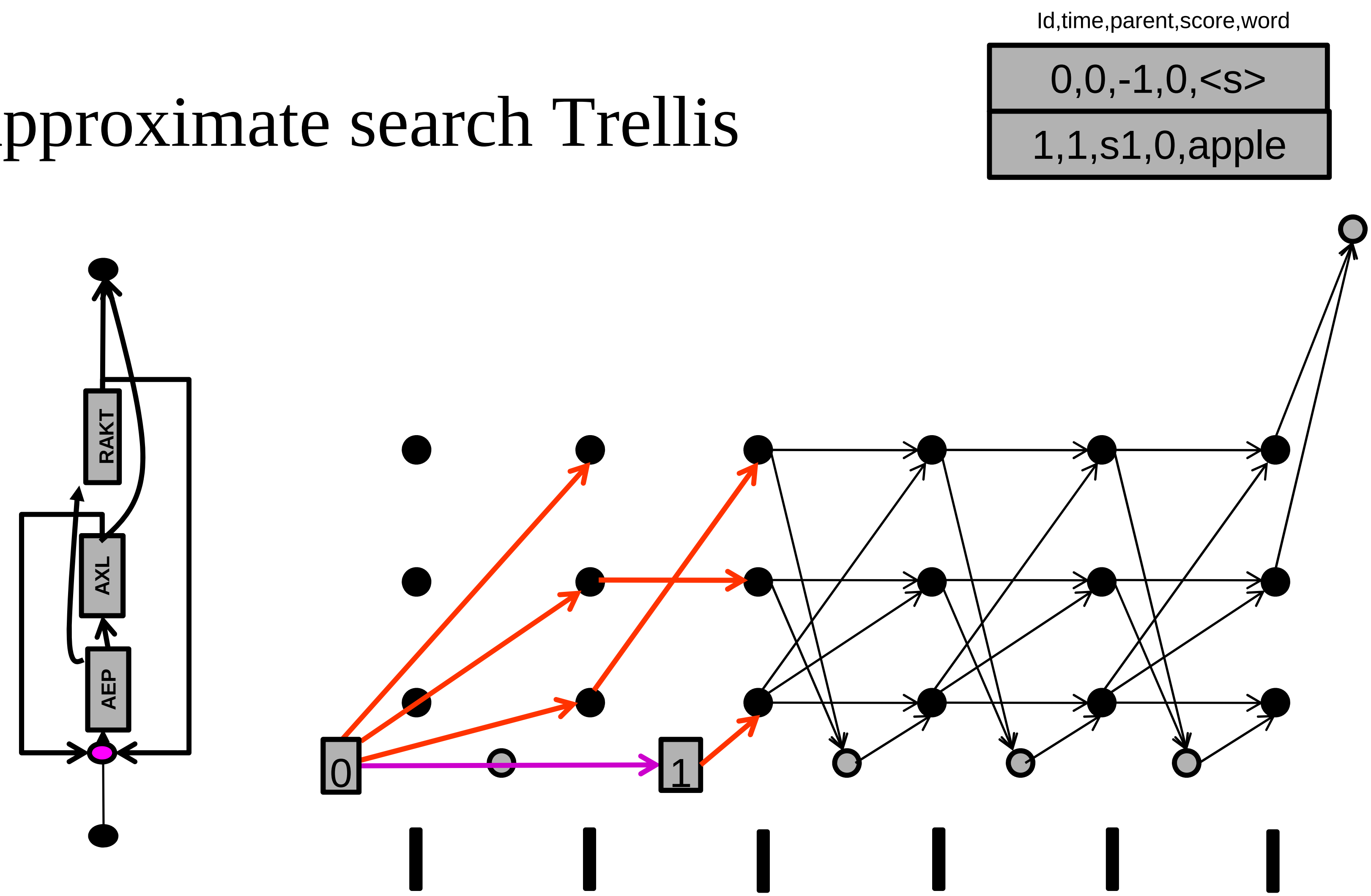
- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

Approximate search Trellis



- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

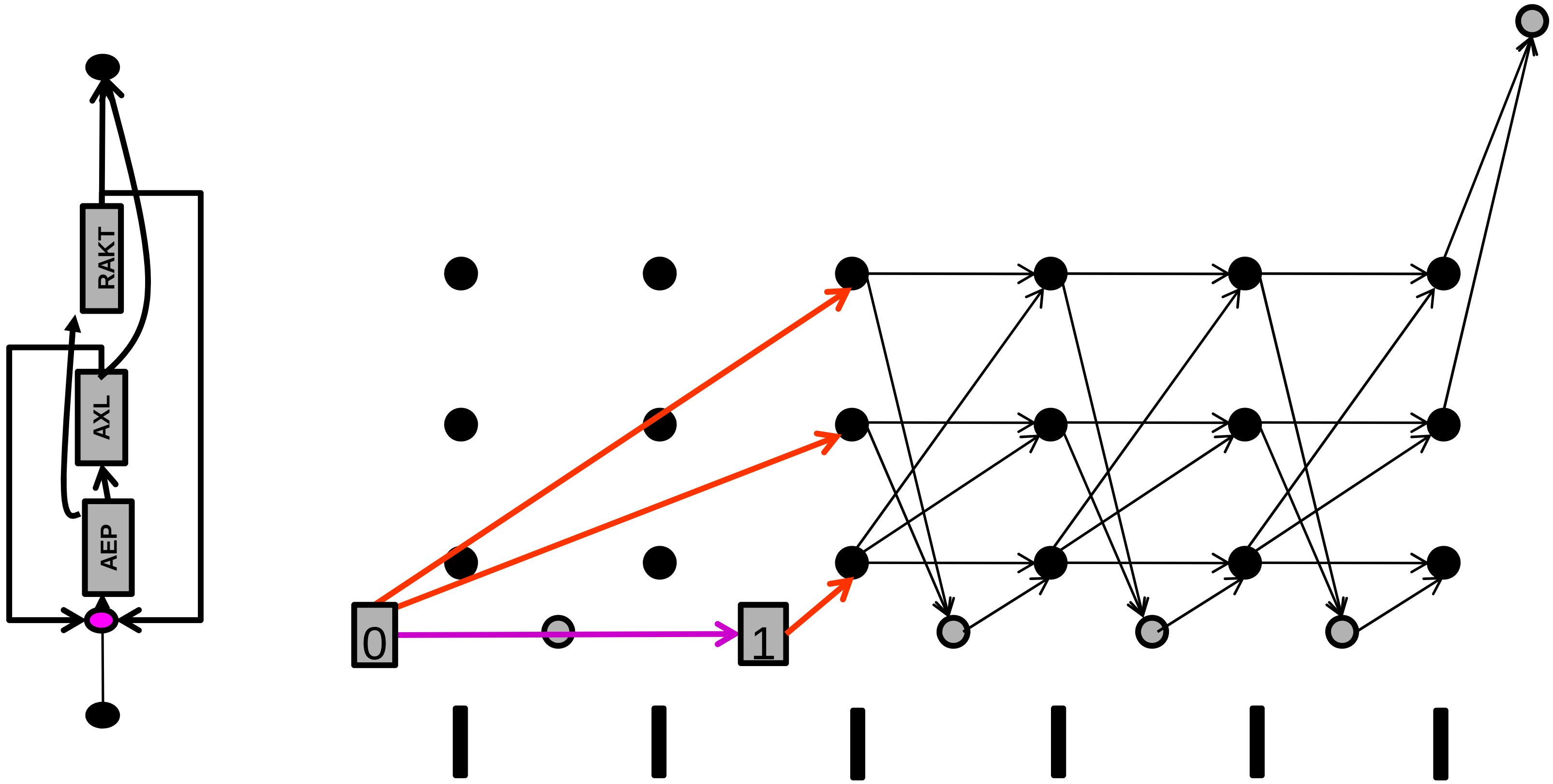
Approximate search Trellis



- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

Approximate search Trellis

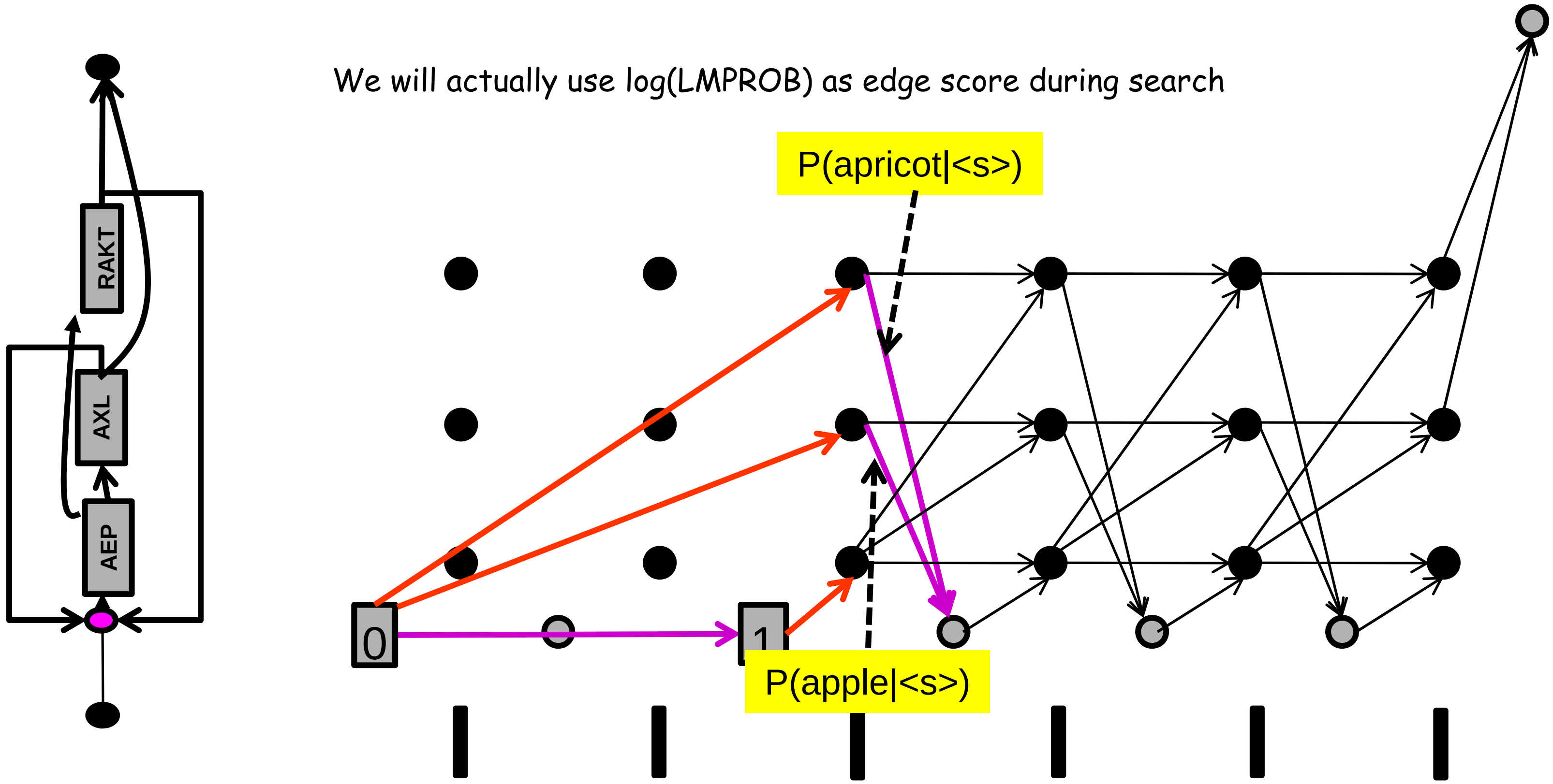
Id,time,parent,score,word
0,0,-1,0,<s>
1,1,s1,0,apple



- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

Approximate search Trellis

Id,time,parent,score,word
0,0,-1,0,<s>
1,1,s1,0,apple

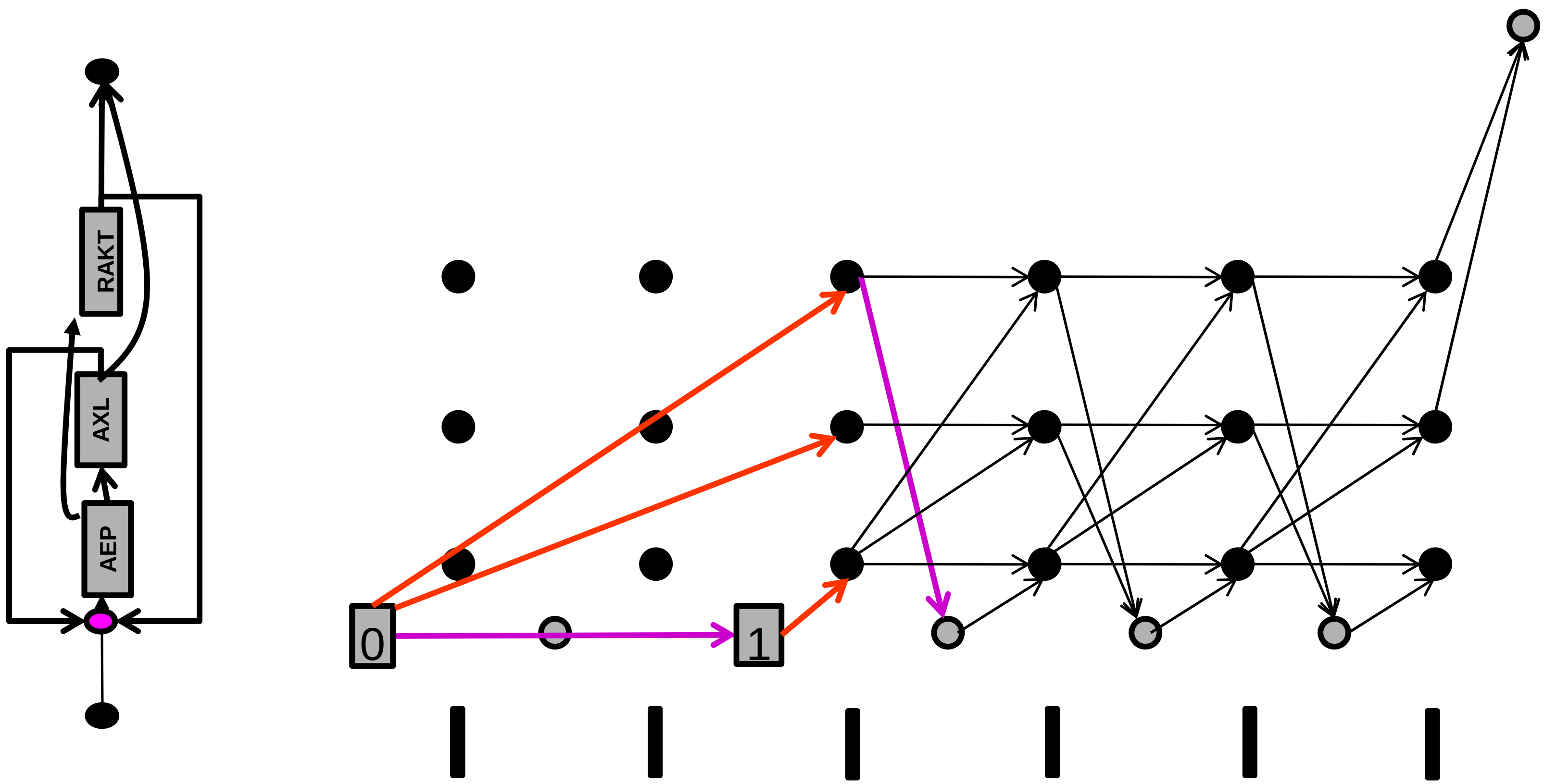


- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

Approximate search Trellis

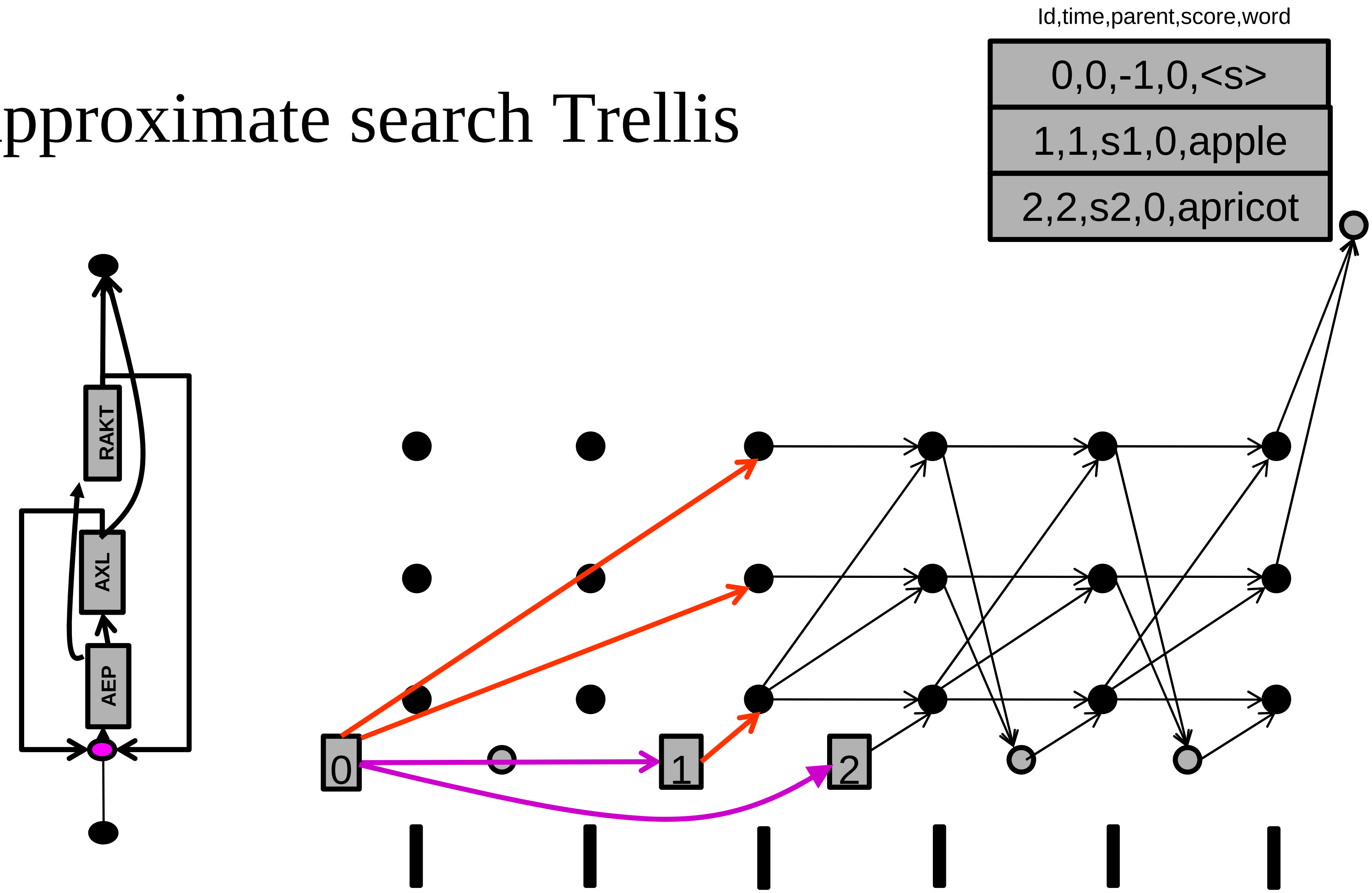
Id,time,parent,score,word

0,0,-1,0,<s>
1,1,s1,0,apple



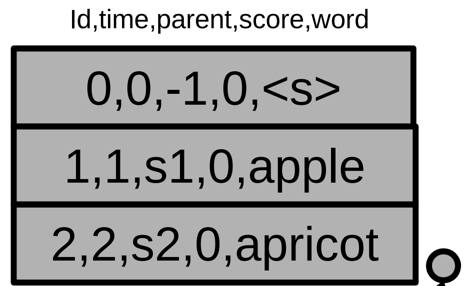
- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

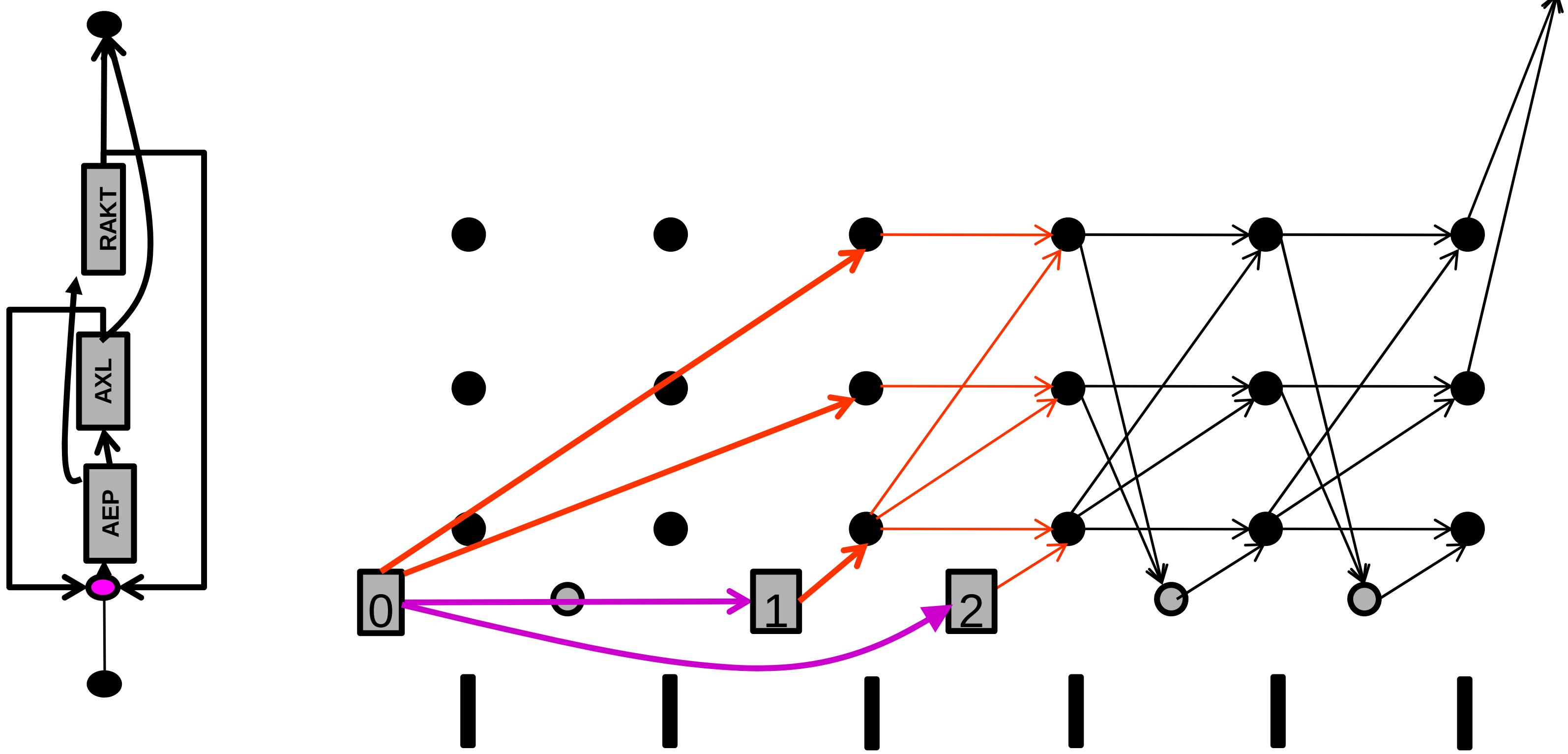
Approximate search Trellis



- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

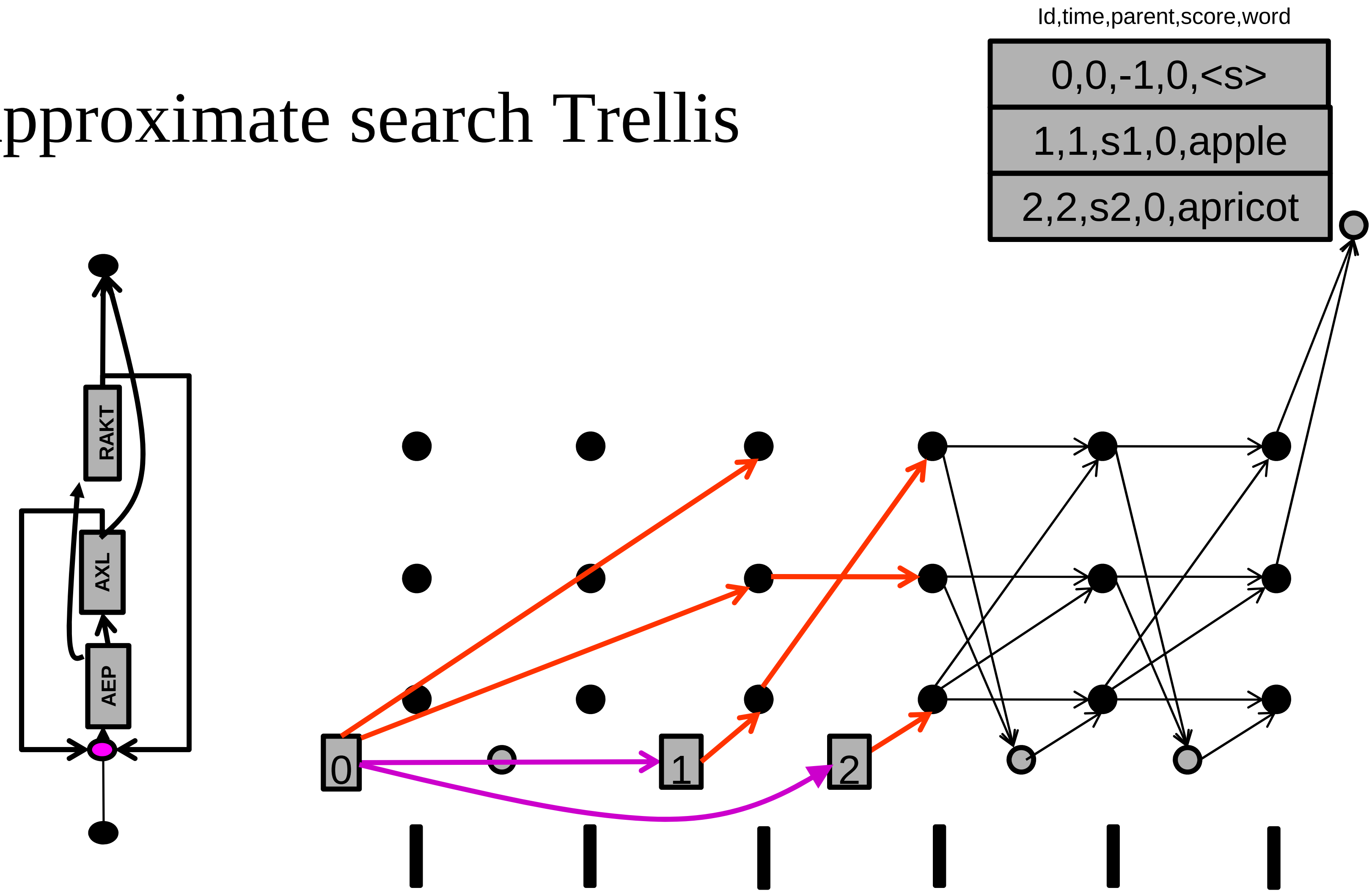
Approximate search Trellis





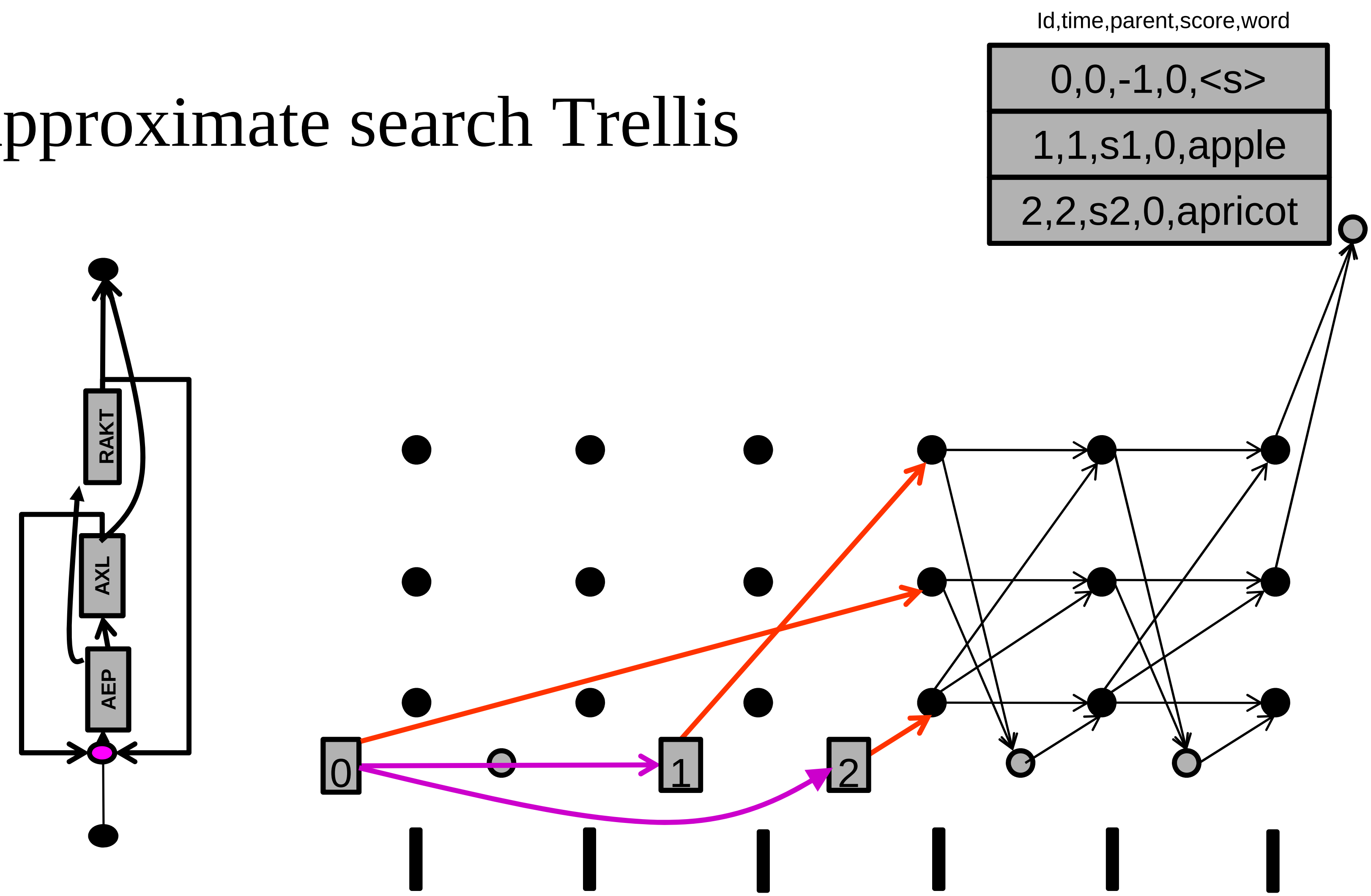
- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

Approximate search Trellis



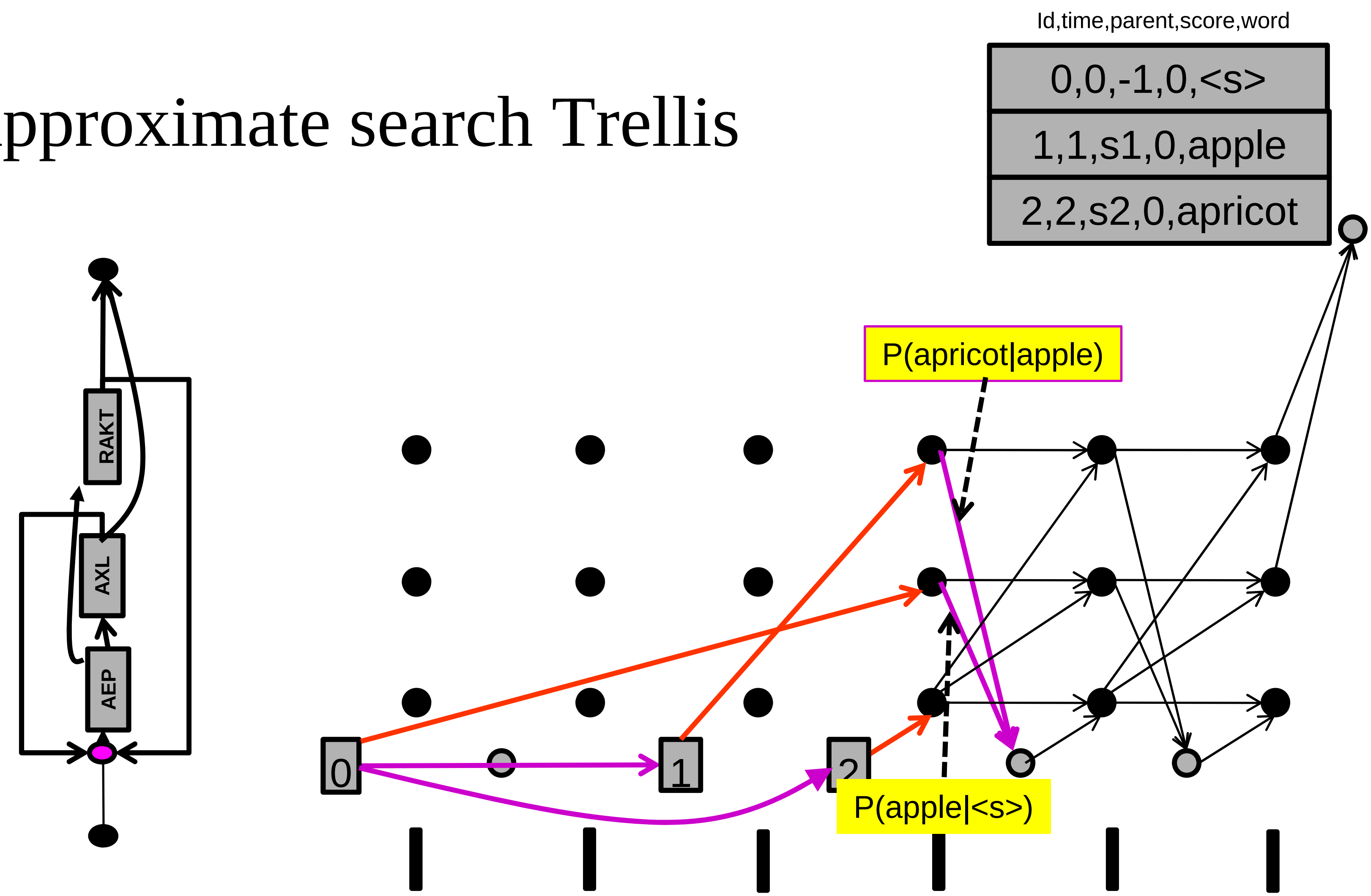
- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

Approximate search Trellis



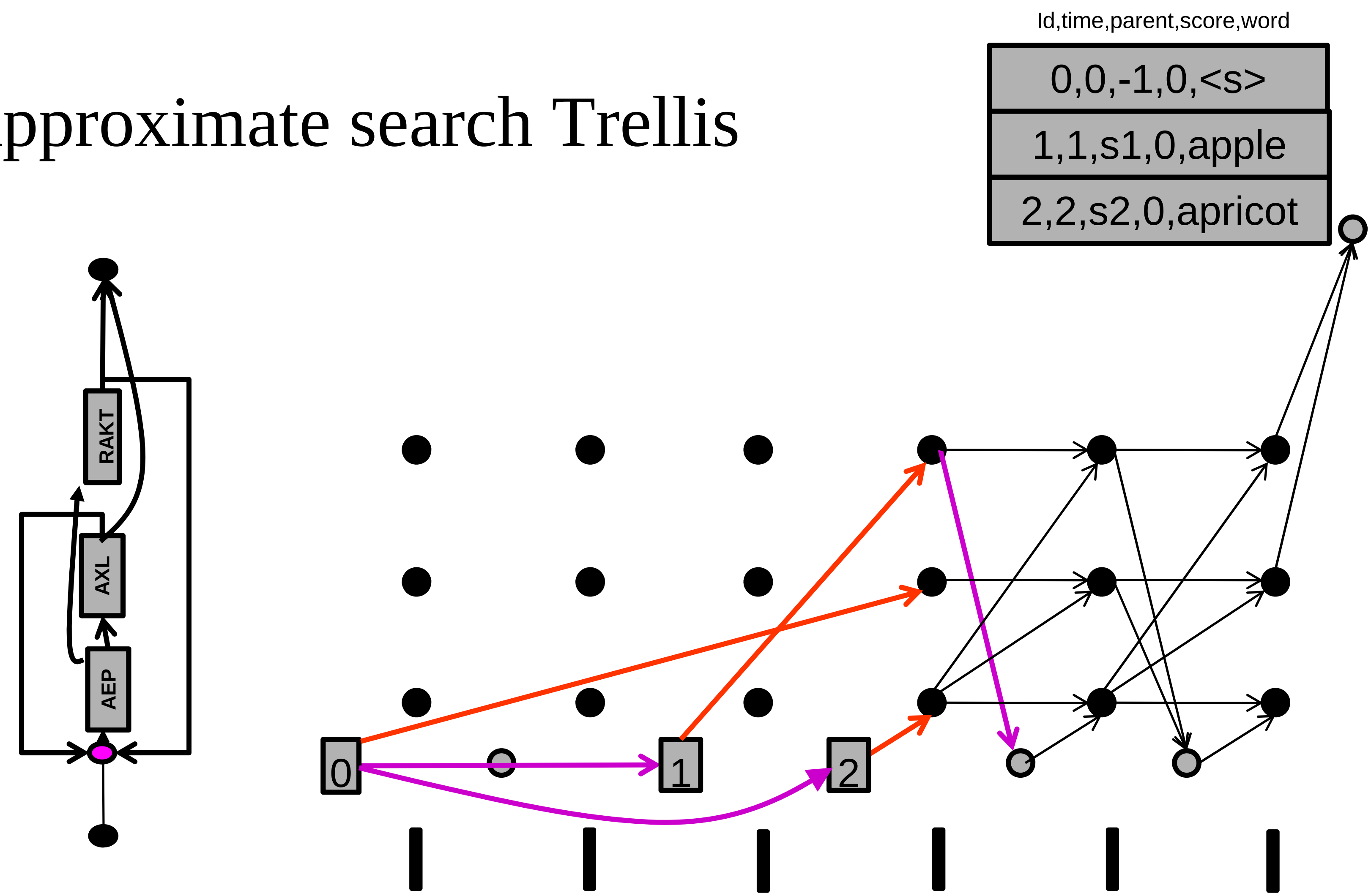
- Search follows usual rules except that at word transitions we look up the word history to apply LM probabilities

Approximate search Trellis



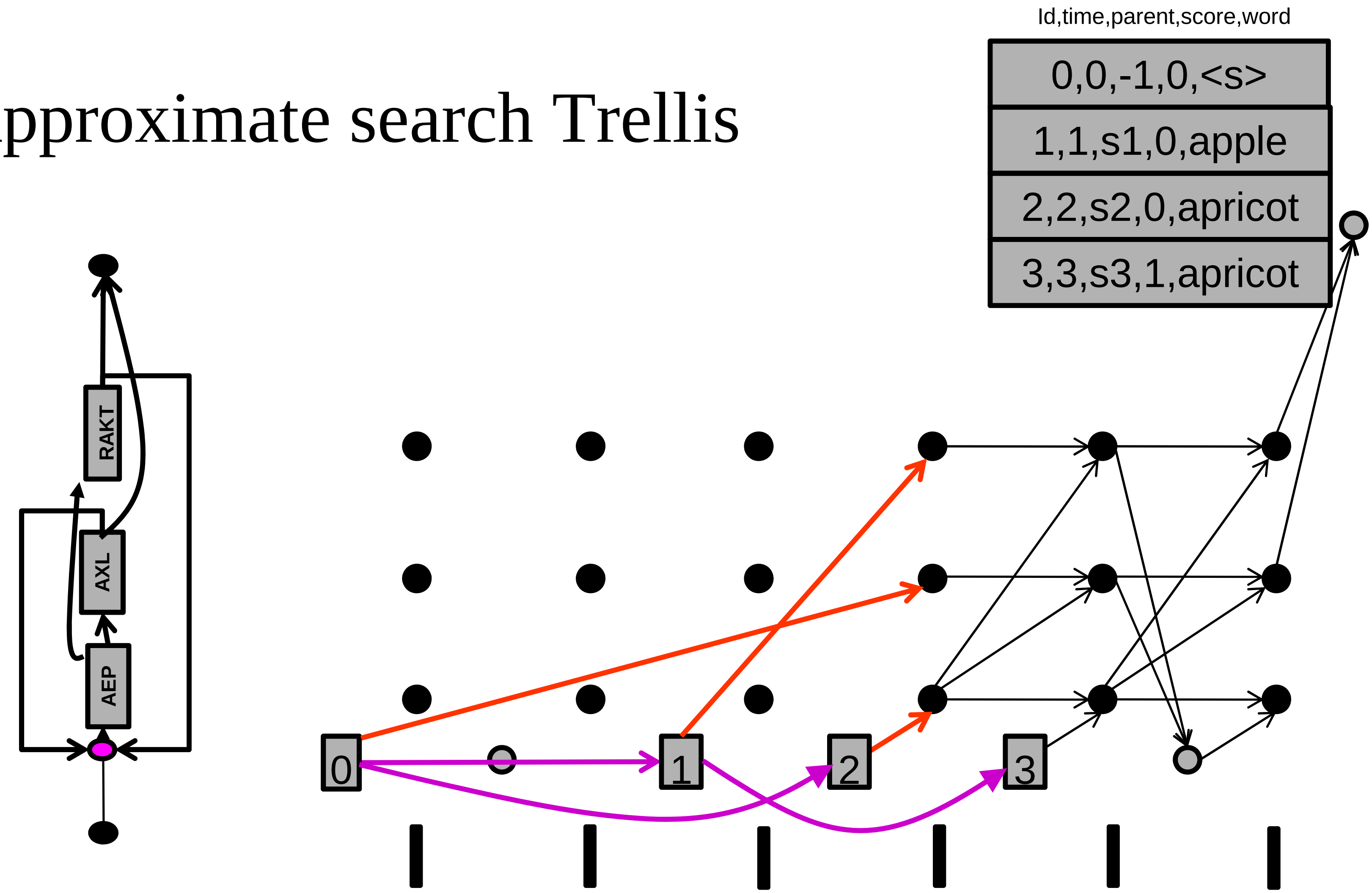
- The transition out of “Apricot” carries the probability P(Apricot|Apple) because the parent of the current state is the word “apple”
- This information is retrieved from the backpointer table

Approximate search Trellis



- Search rules do not change – the best incoming entry is retained

Approximate search Trellis

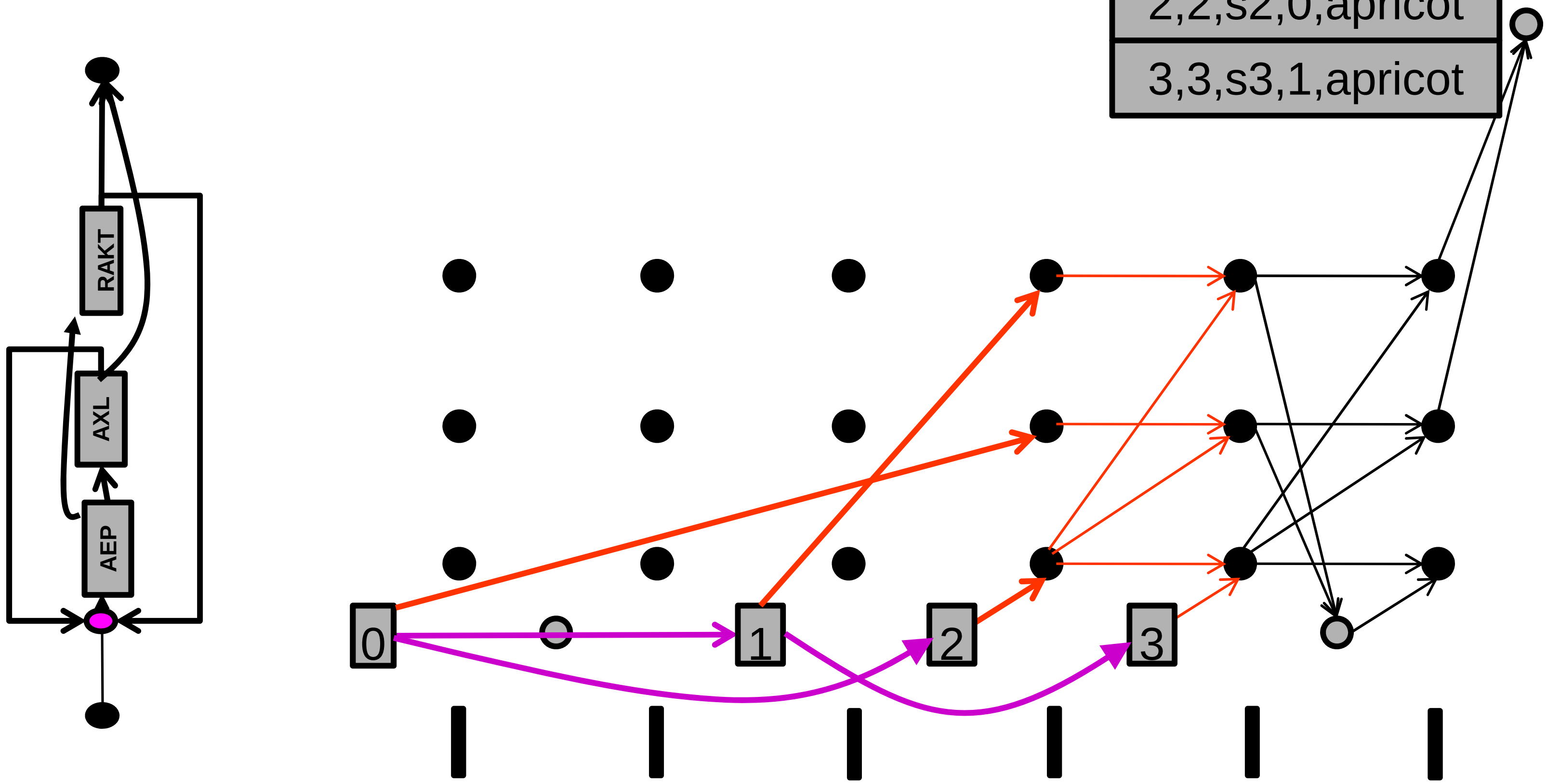


- Search rules do not change – the best incoming entry is retained

Approximate search Trellis

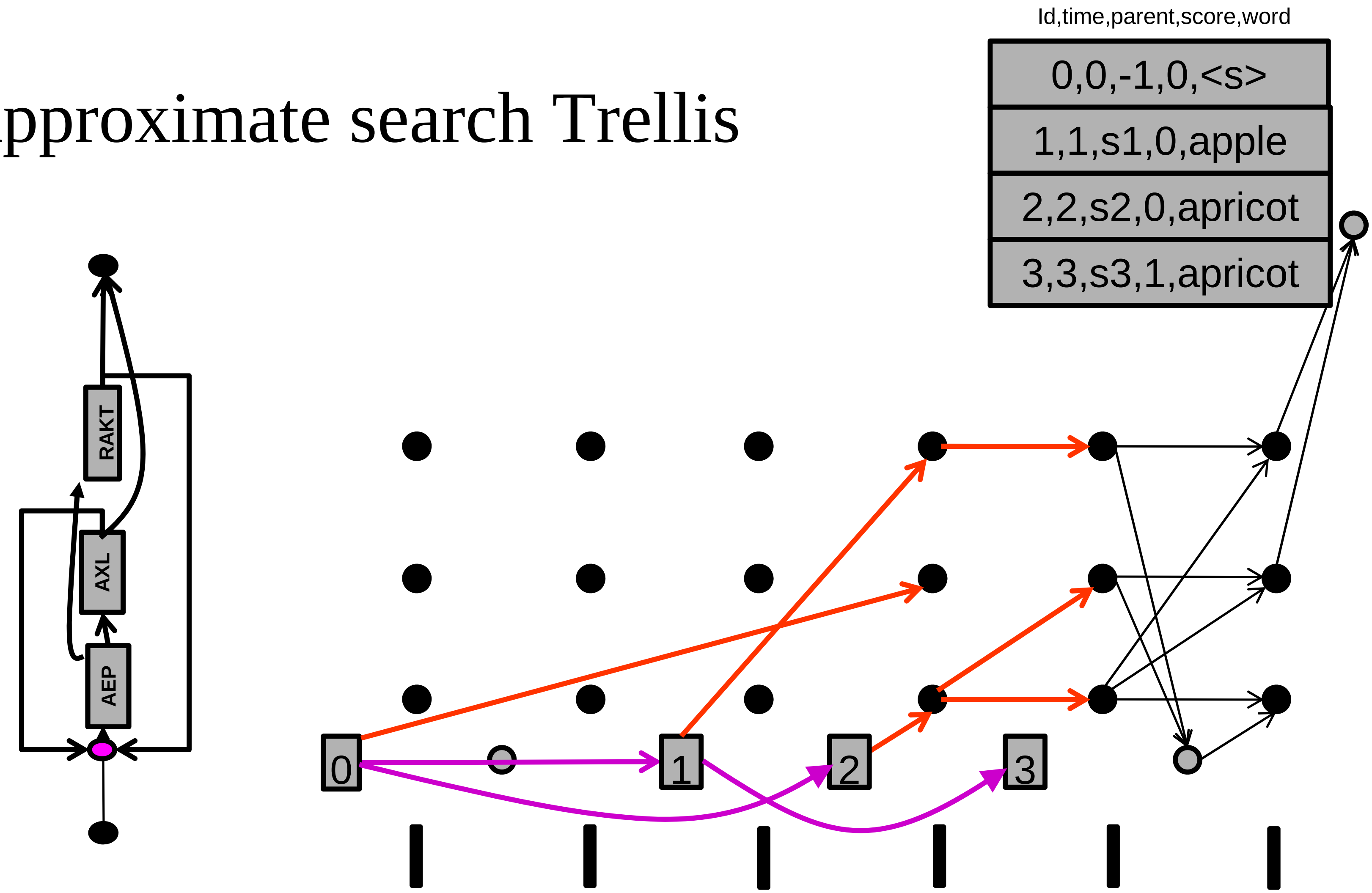
Id,time,parent,score,word

0,0,-1,0,<s>
1,1,s1,0,apple
2,2,s2,0,apricot
3,3,s3,1,apricot



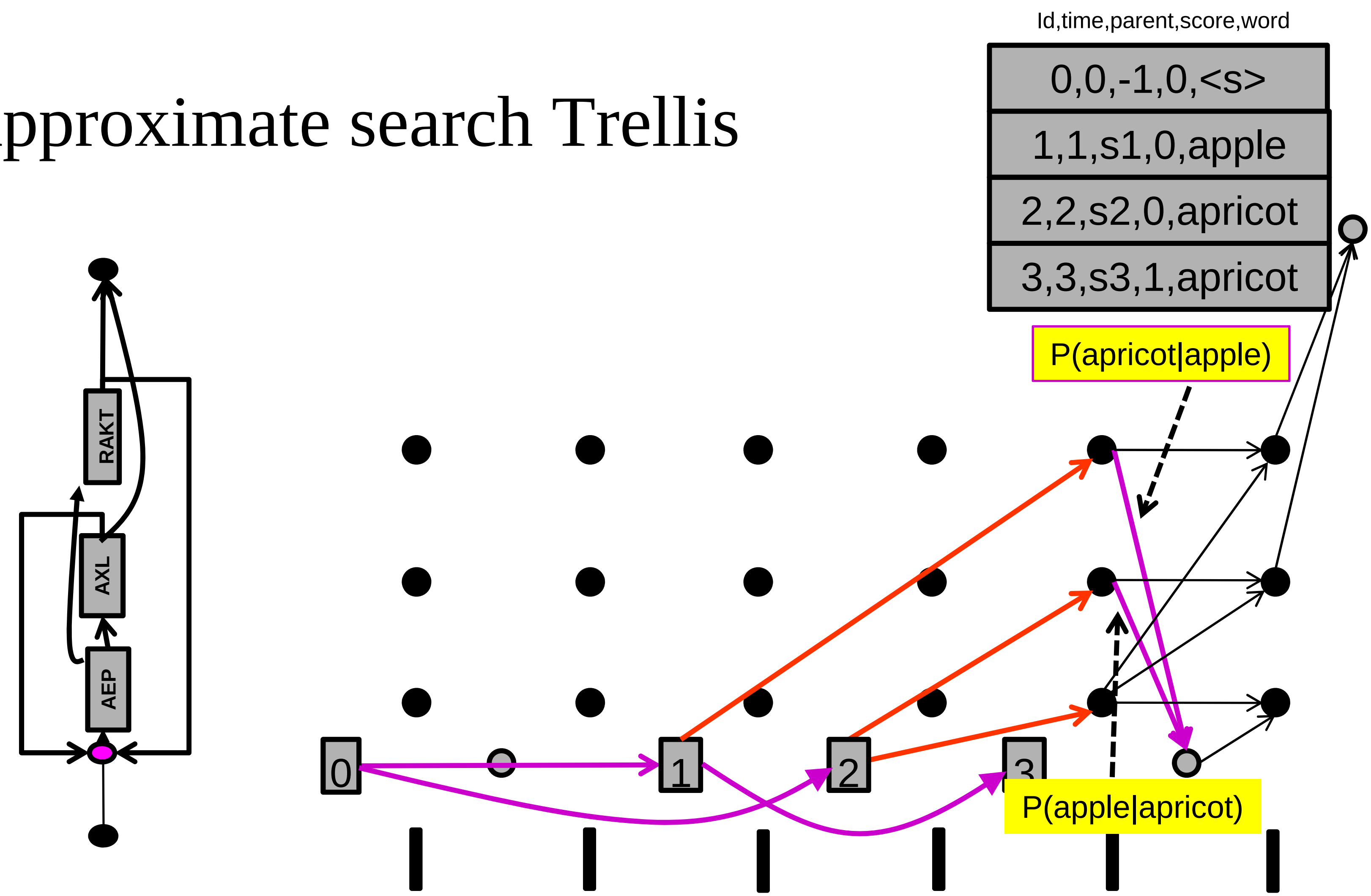
- Search rules do not change – the best incoming entry is retained

Approximate search Trellis



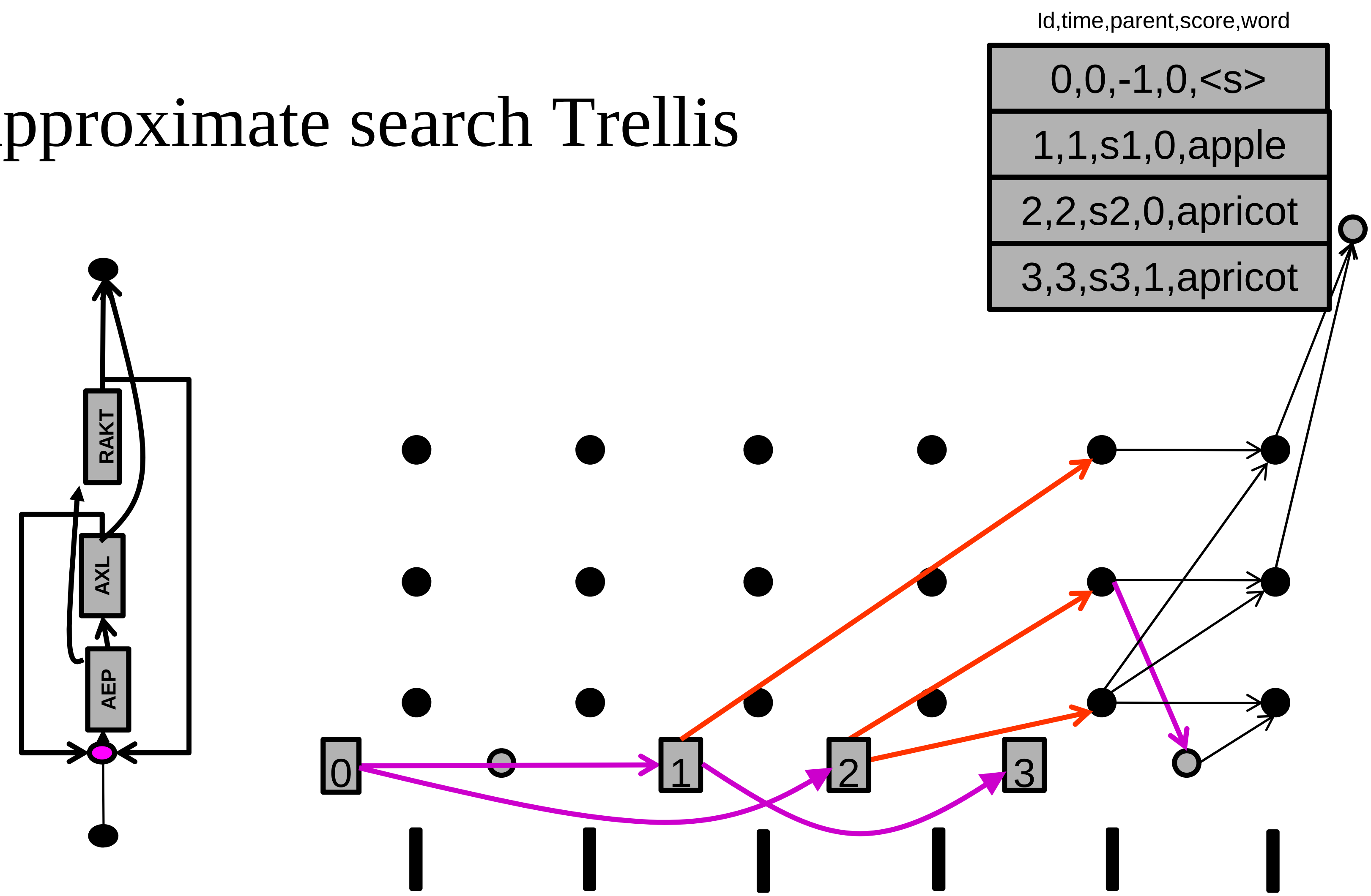
- Search rules do not change – the best incoming entry is retained

Approximate search Trellis



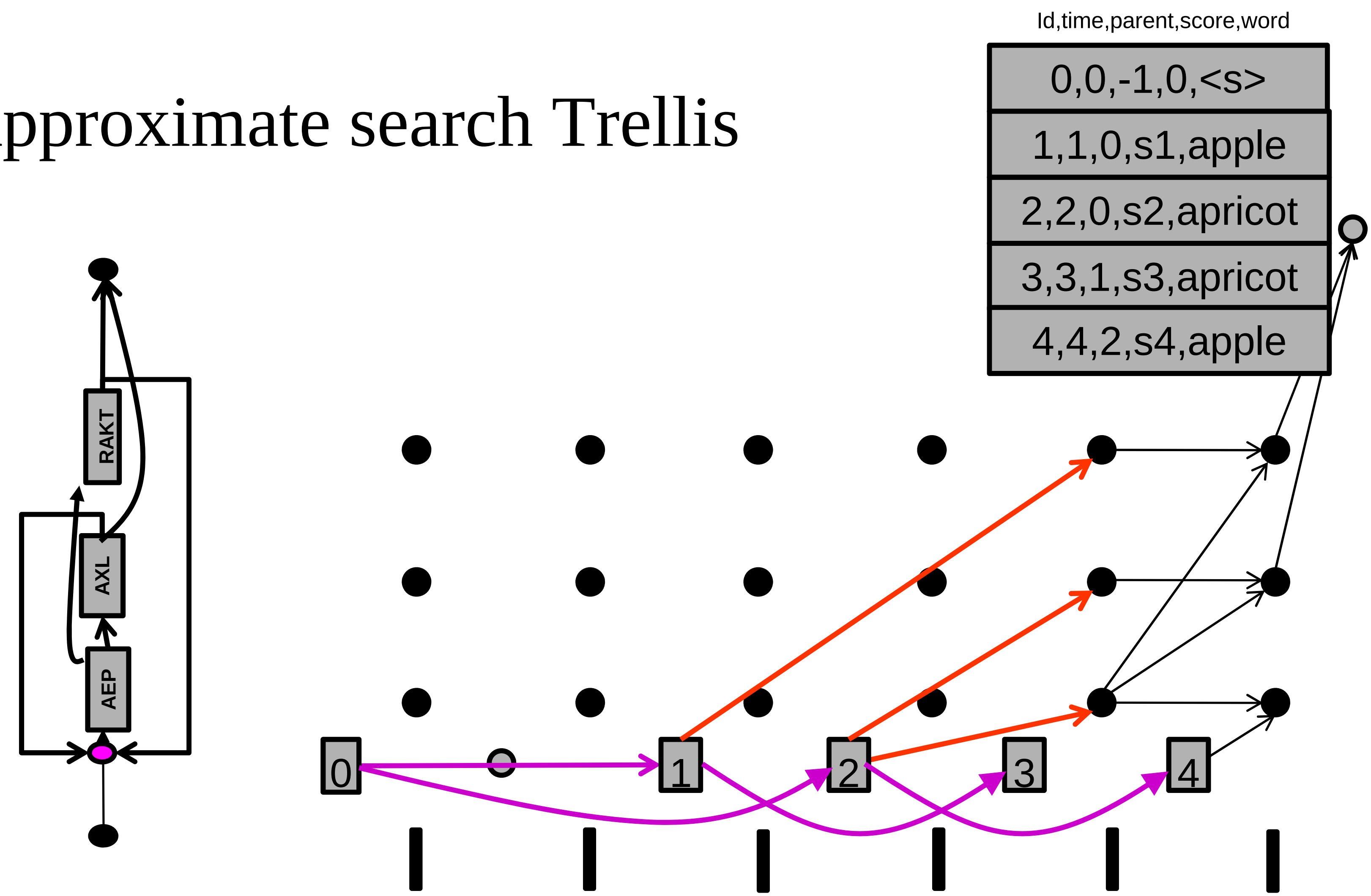
- Note the conditioning word in the bigram probabilities applied

Approximate search Trellis



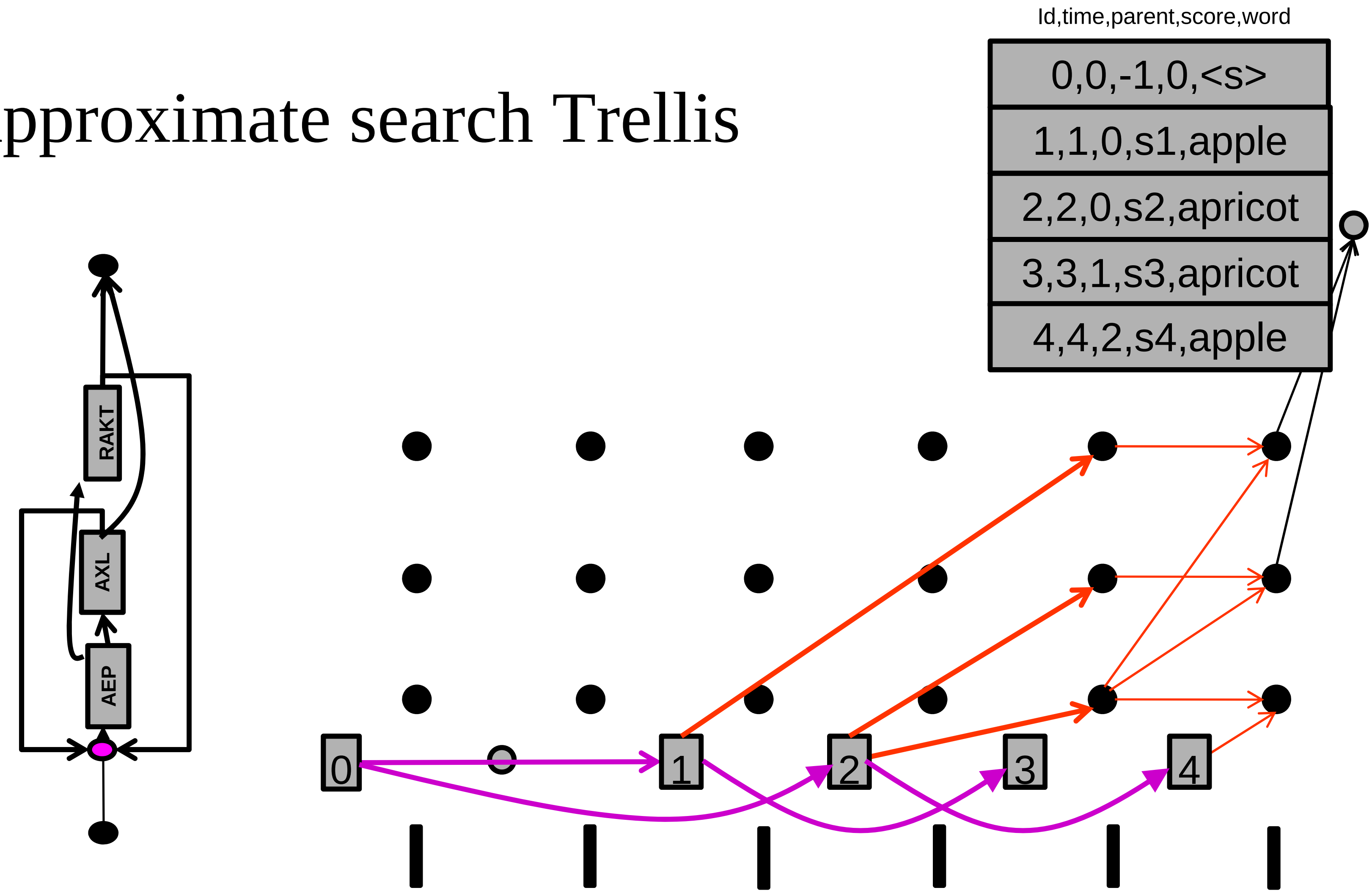
- The winner remains as before

Approximate search Trellis



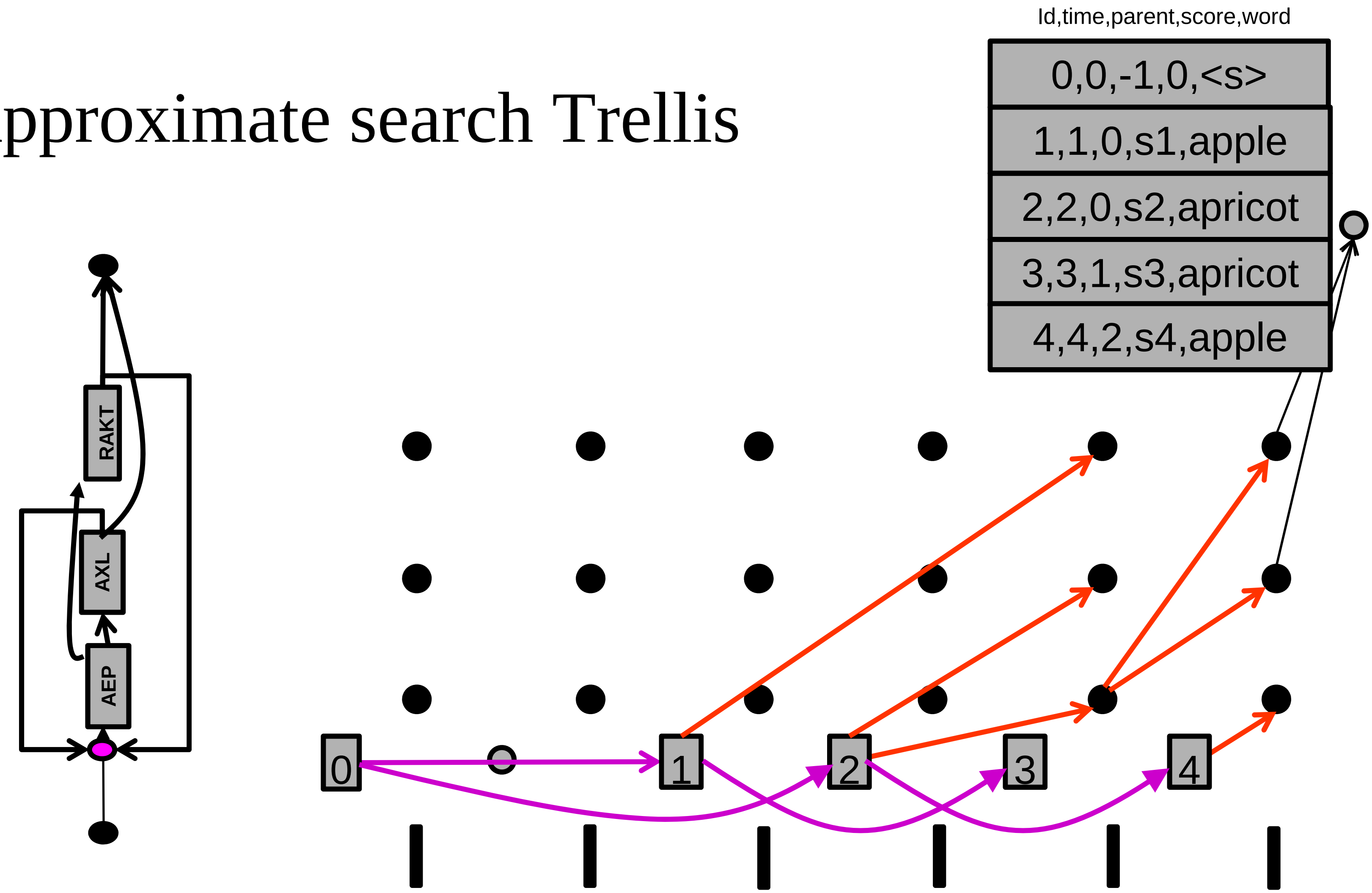
- The winner remains as before

Approximate search Trellis



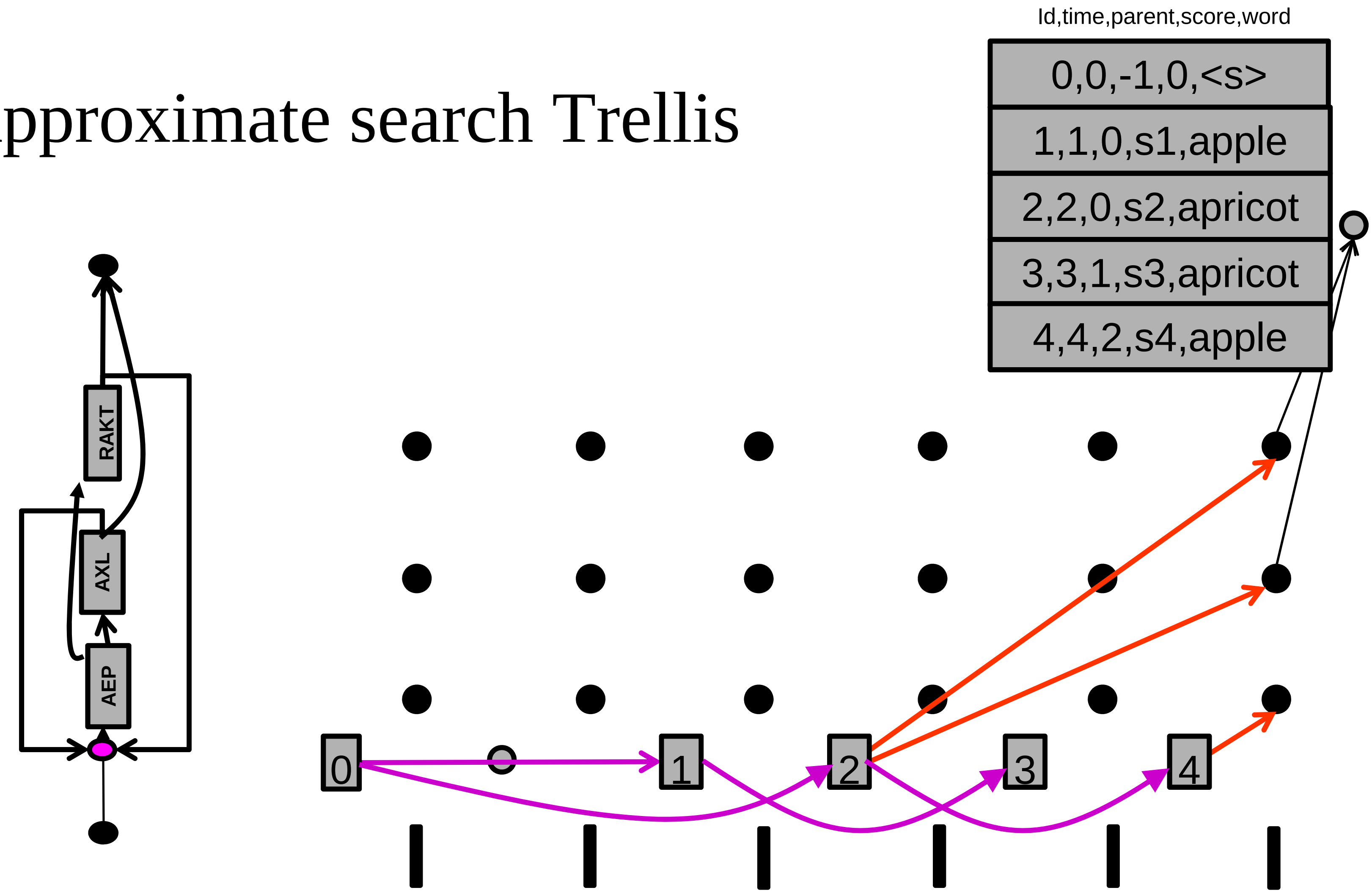
- Lets follow this to the end

Approximate search Trellis

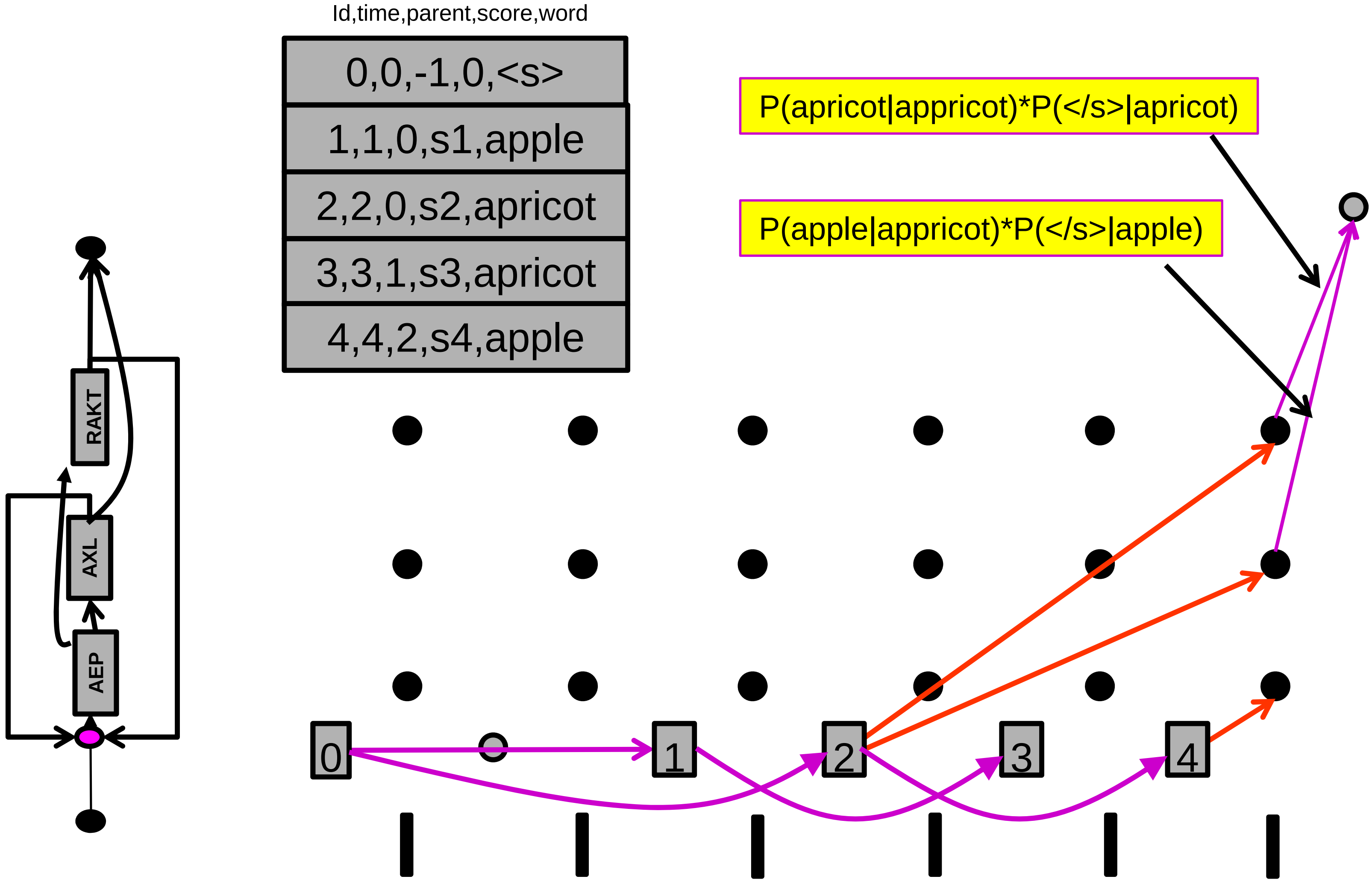


- Lets follow this to the end

Approximate search Trellis



- Lets follow this to the end



- Note the probabilities being applied to the final transition into sentence ending!

Approximate structures with lextrees

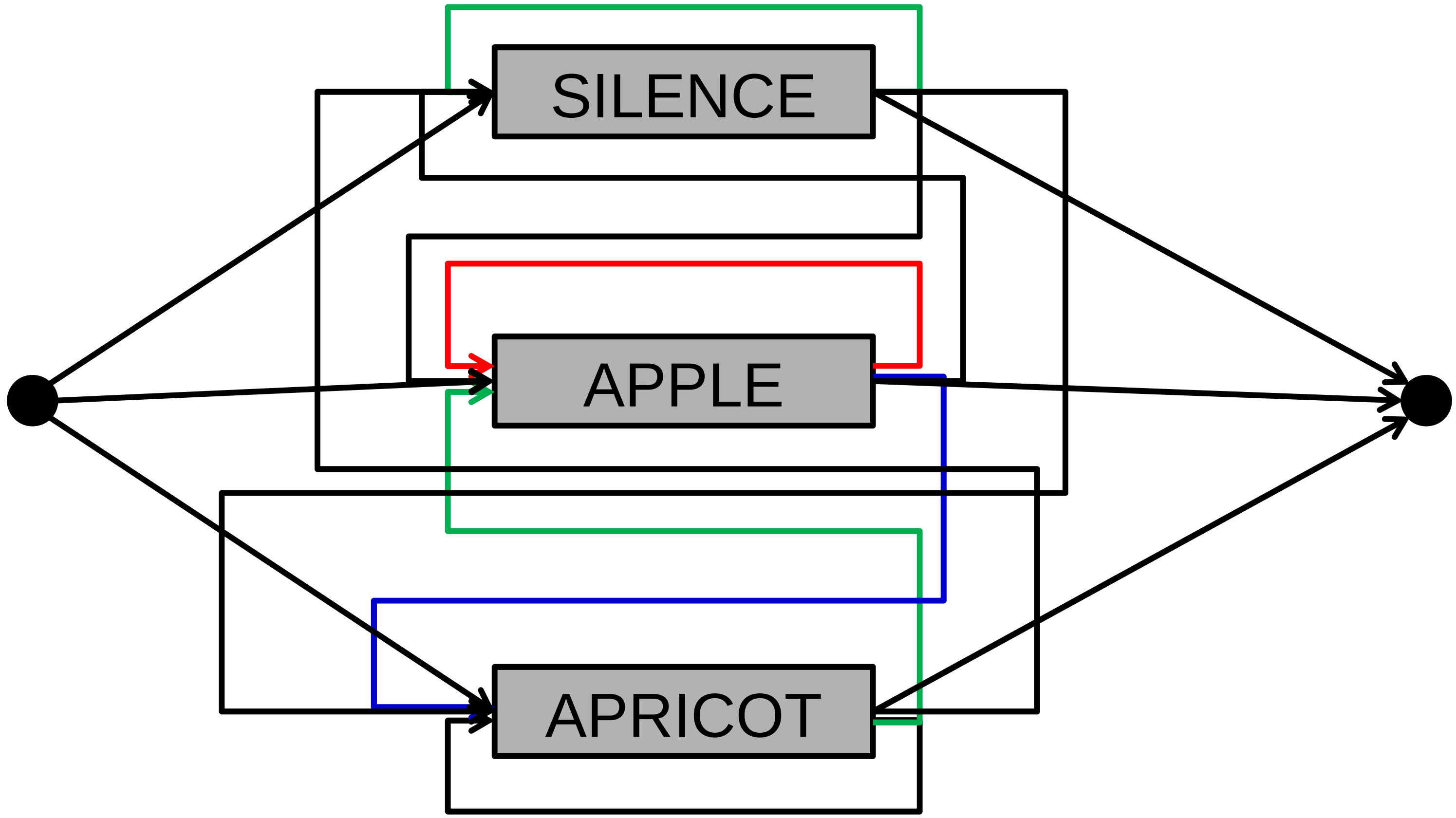
- Can use trigram probabilities instead of bigrams without modifying search strategy
  - Determine previous TWO words and apply appropriate LM trigram probability during search
  - Can in fact handle ANY left-to-right language model

- The approximate structure shown earlier is suboptimal
  - Although highly popular, particularly for embedded systems

- A better approximation is obtained using *multiple* lextrees
  - Typically 3-5 lextrees
  - The distinction between the lextrees is in the *time* of entry: incoming arcs into the j-th (of K) lextrees only activate if T%K = j
    - i.e. each lextree can be entered only once every K frames
    - Other similar heuristics may be applied

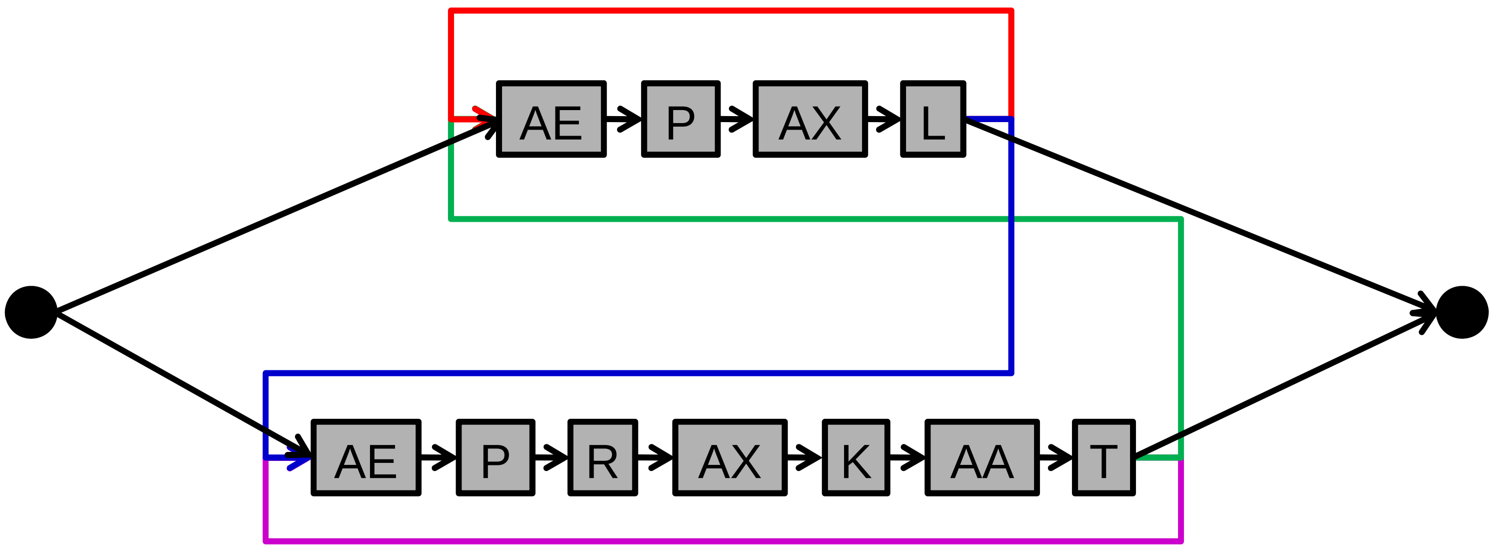
- A still better approximation is obtained using a *flat bigram search* structure

Approximate decode with flat bigram structure



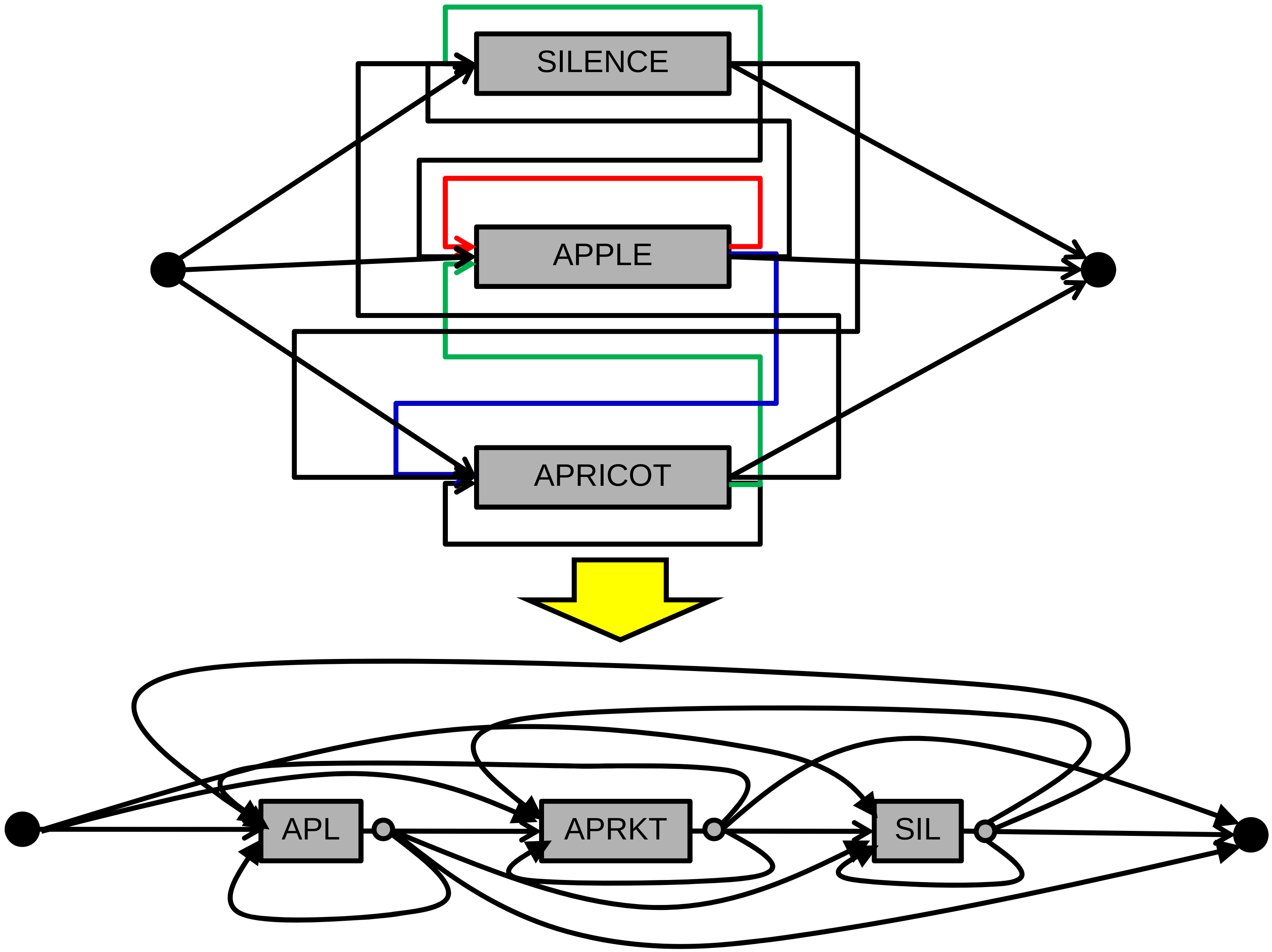
- A better (but more complex) approximate search uses the flat bigram structure shown above
- Note the manner in which silence is inserted
  - Very simple
- Once again, no LM probabilities are introduced at this stage

A closer look at the flat bigram



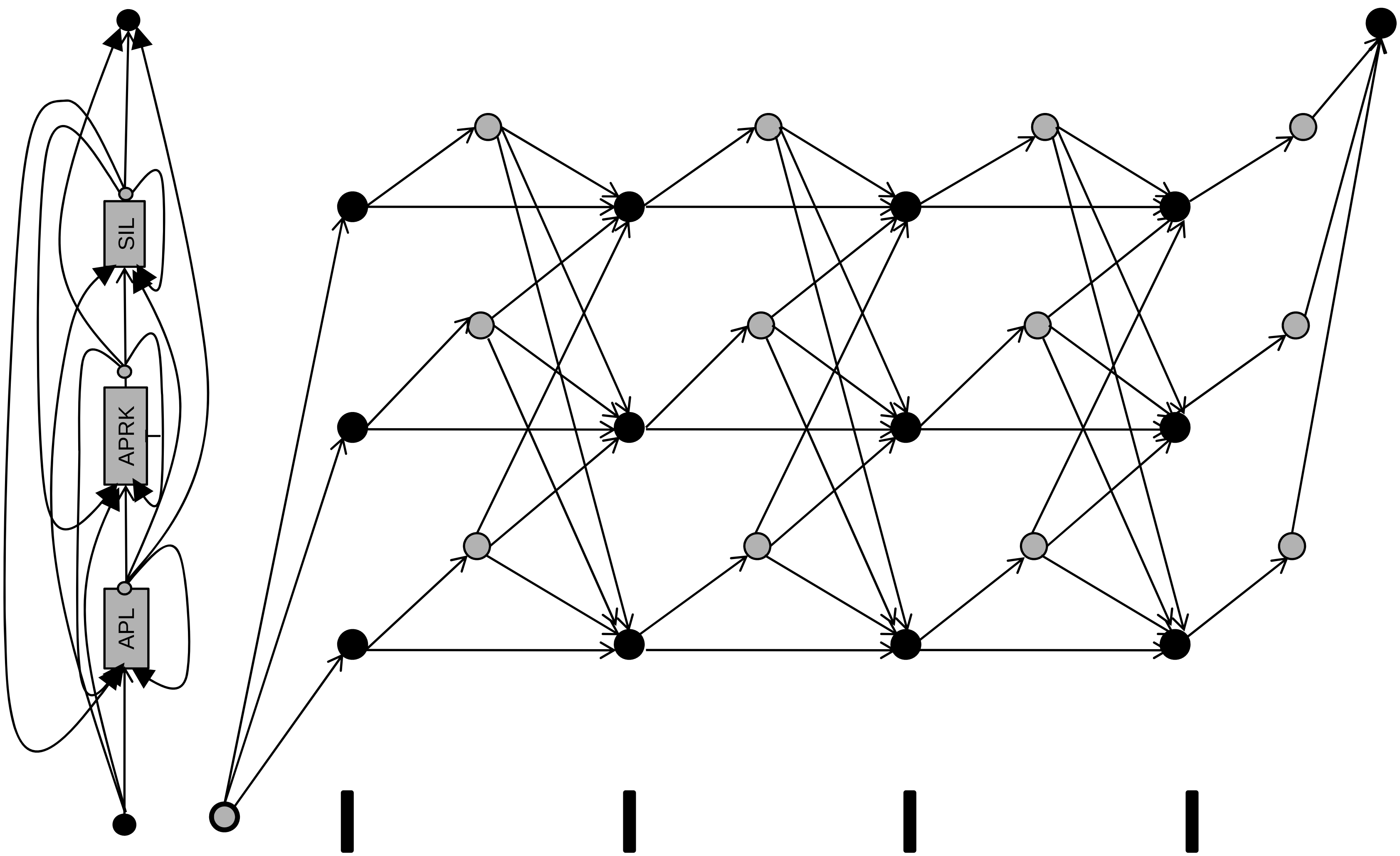
- Not showing silence above to keep it simple
  - But in reality, silence will be included
  - Note: No LM probabilities included
- We take no advantage of the fact that phonemes are shared, however
  - We want to be able to determine word identity at the entry to a word
  - In the following slides we will not show the phonetic breakup of words to keep figures simple

The flat bigram structure



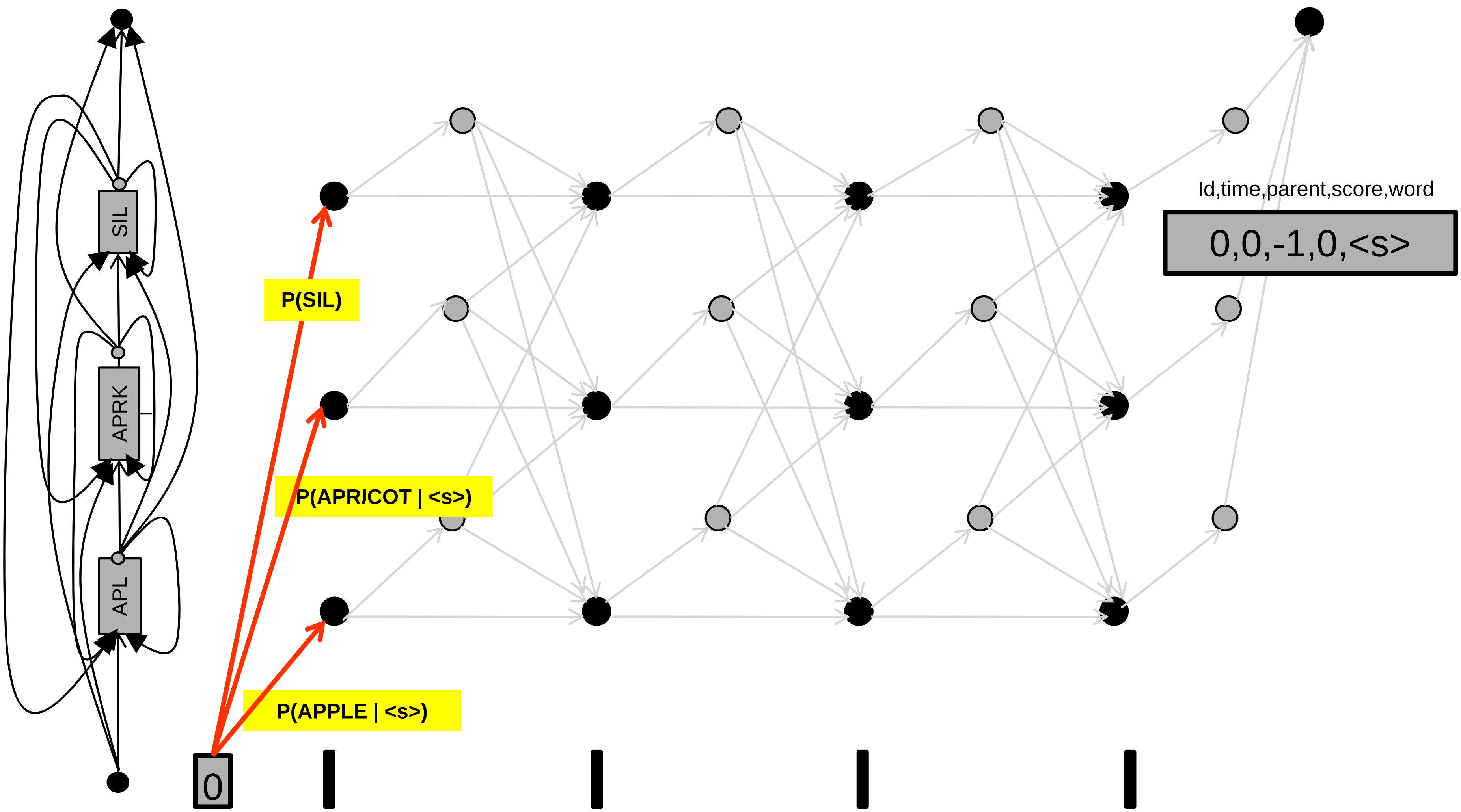
- In the following slides we will assume each word has only one state to simplify illustration

Recognition with flat bigram structure



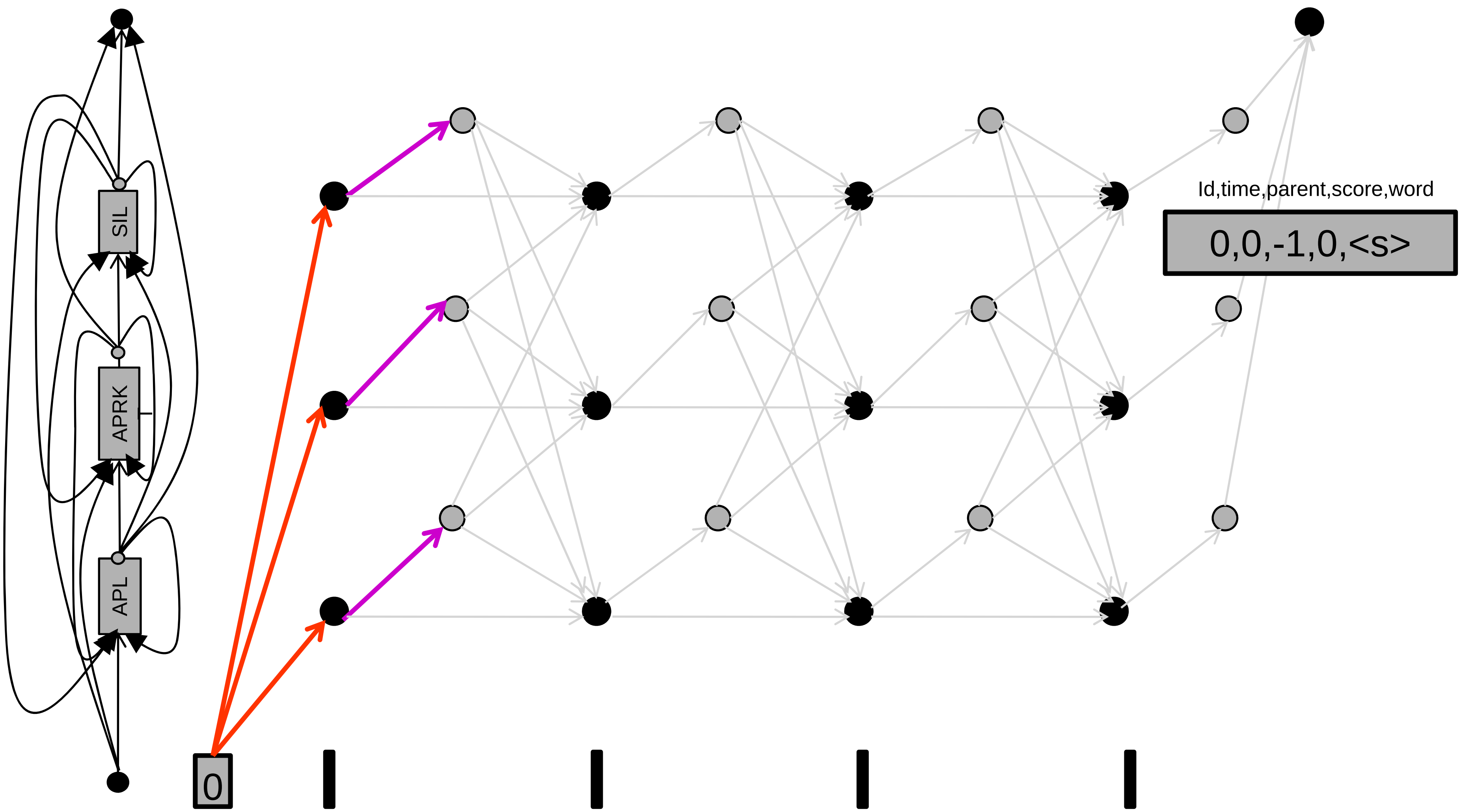
- The trellis is composed as usual
  - But no cross-word language-probabilities are introduced
    - Note: In this form of trellis the non-emitting state at word beginning may be superfluous

Recognition with flat bigram structure



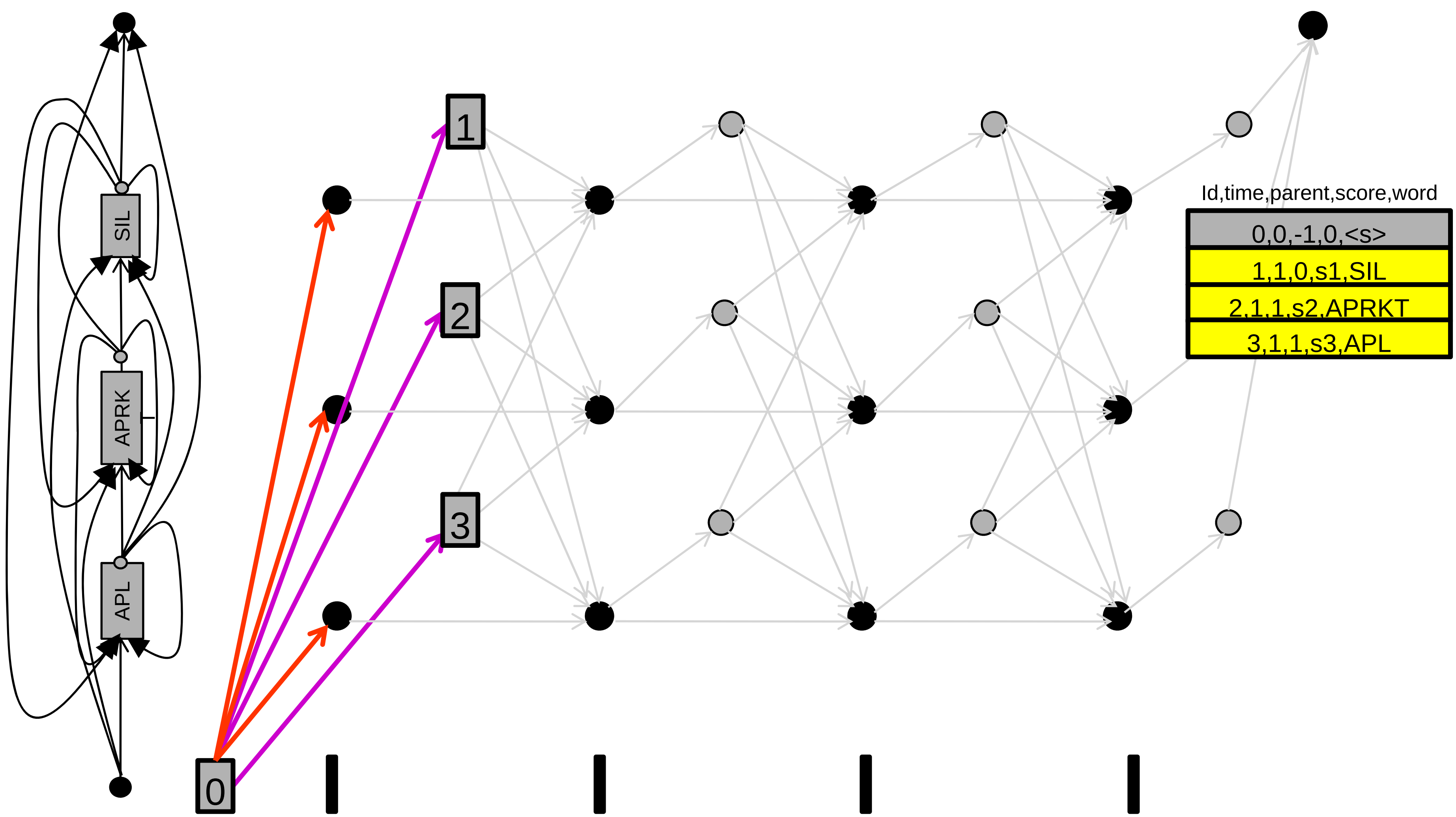
- Bigram probabilities conditioned on start of sentence are applied at the beginning
- Entries to silence carry silence penalty

Recognition with flat bigram structure



- Word ending states move into the backpointer table

Recognition with flat bigram structure

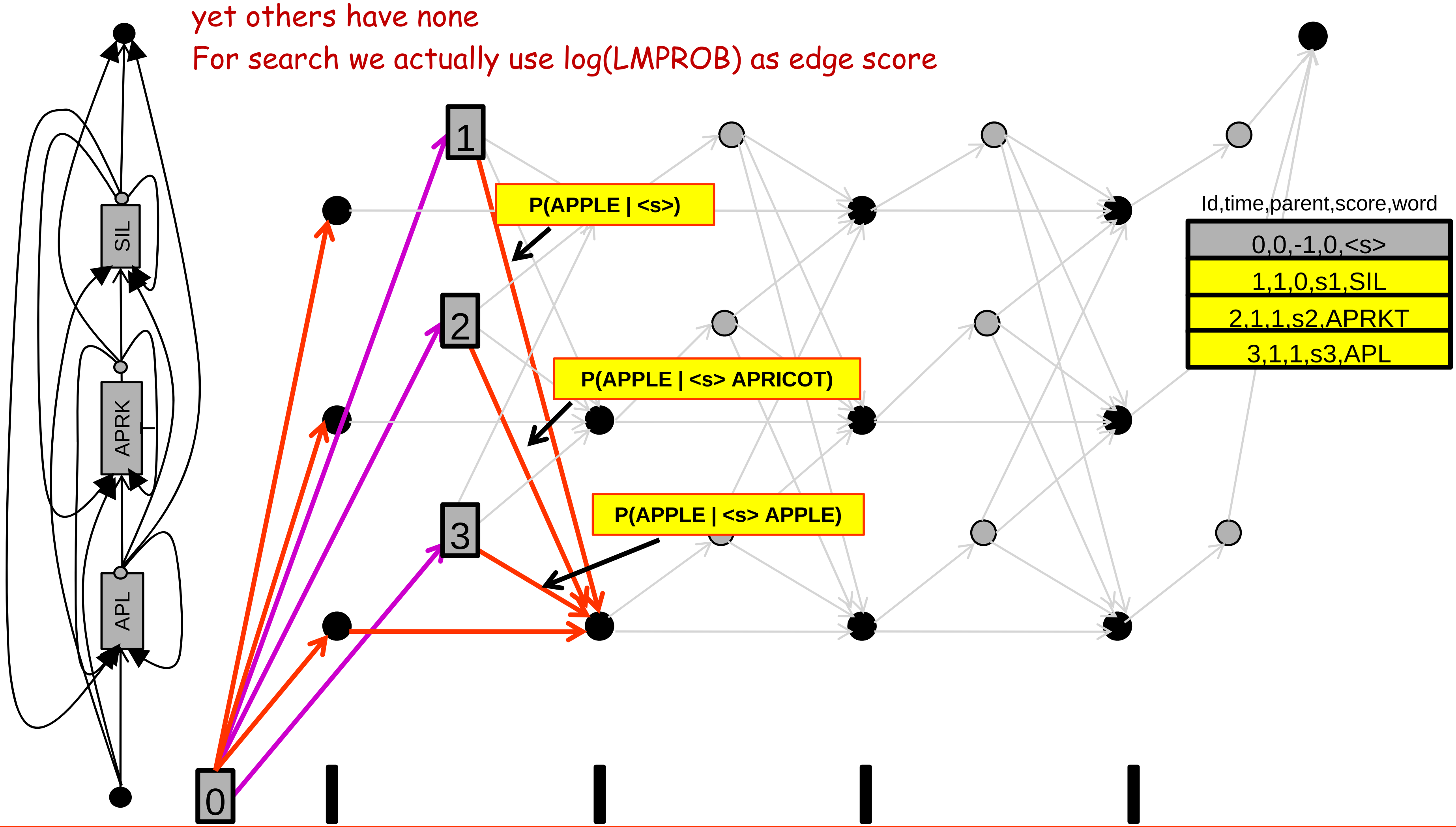


- Word ending states move into the backpointer table

Recognition with flat bigram structure

Some arcs have bigram probs, others have trigram probs, and yet others have none

For search we actually use log(LMPROB) as edge score

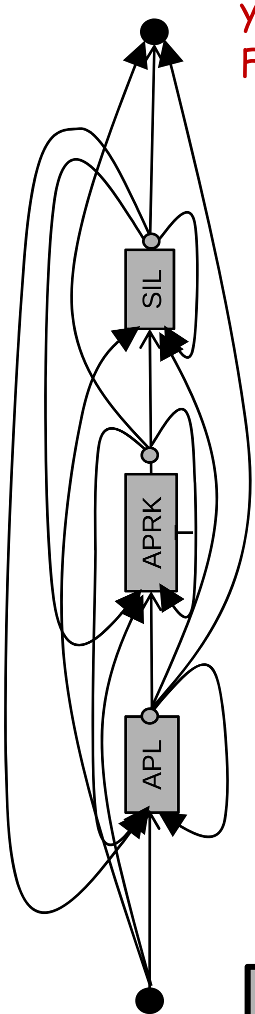


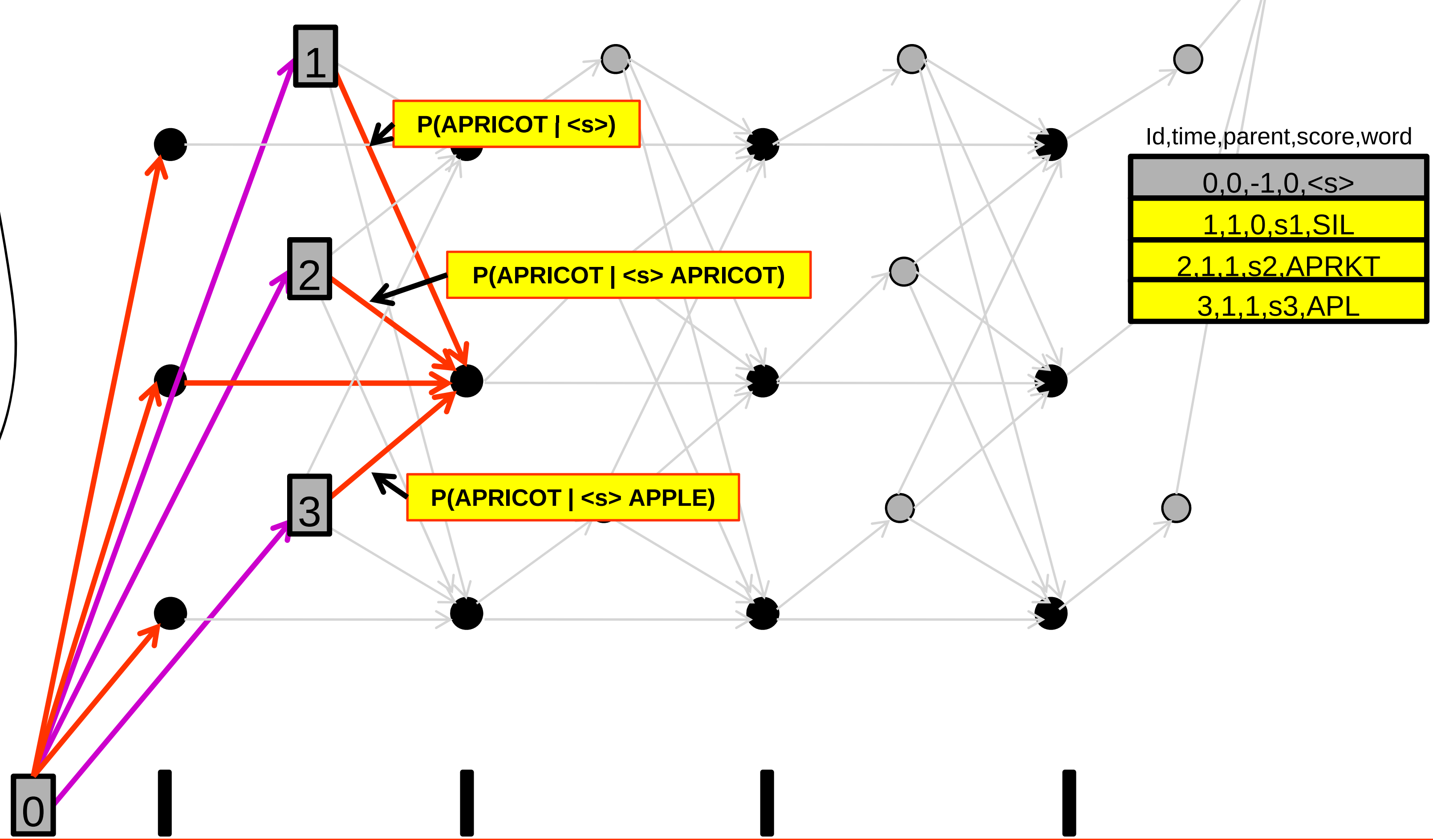
- ***Note the different LM probability terms applied to the arcs***
  - Assuming trigram LM
- The appropriate history to use for the LM probability is obtained from the BPtable

Recognition with flat bigram structure

Some arcs have bigram probs, others have trigram probs, and yet others have none

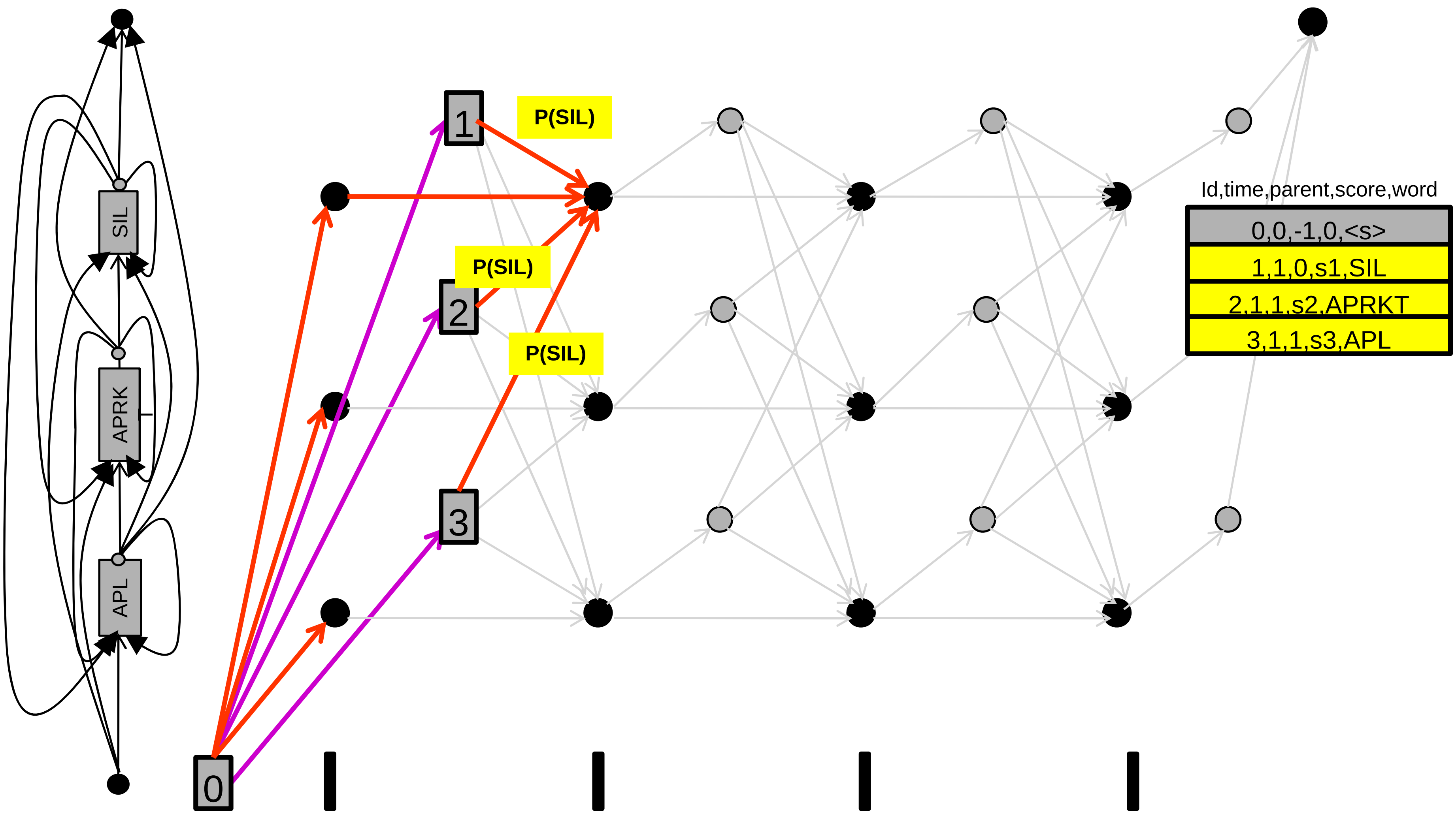
For search we actually use log(LMPROB) as edge score





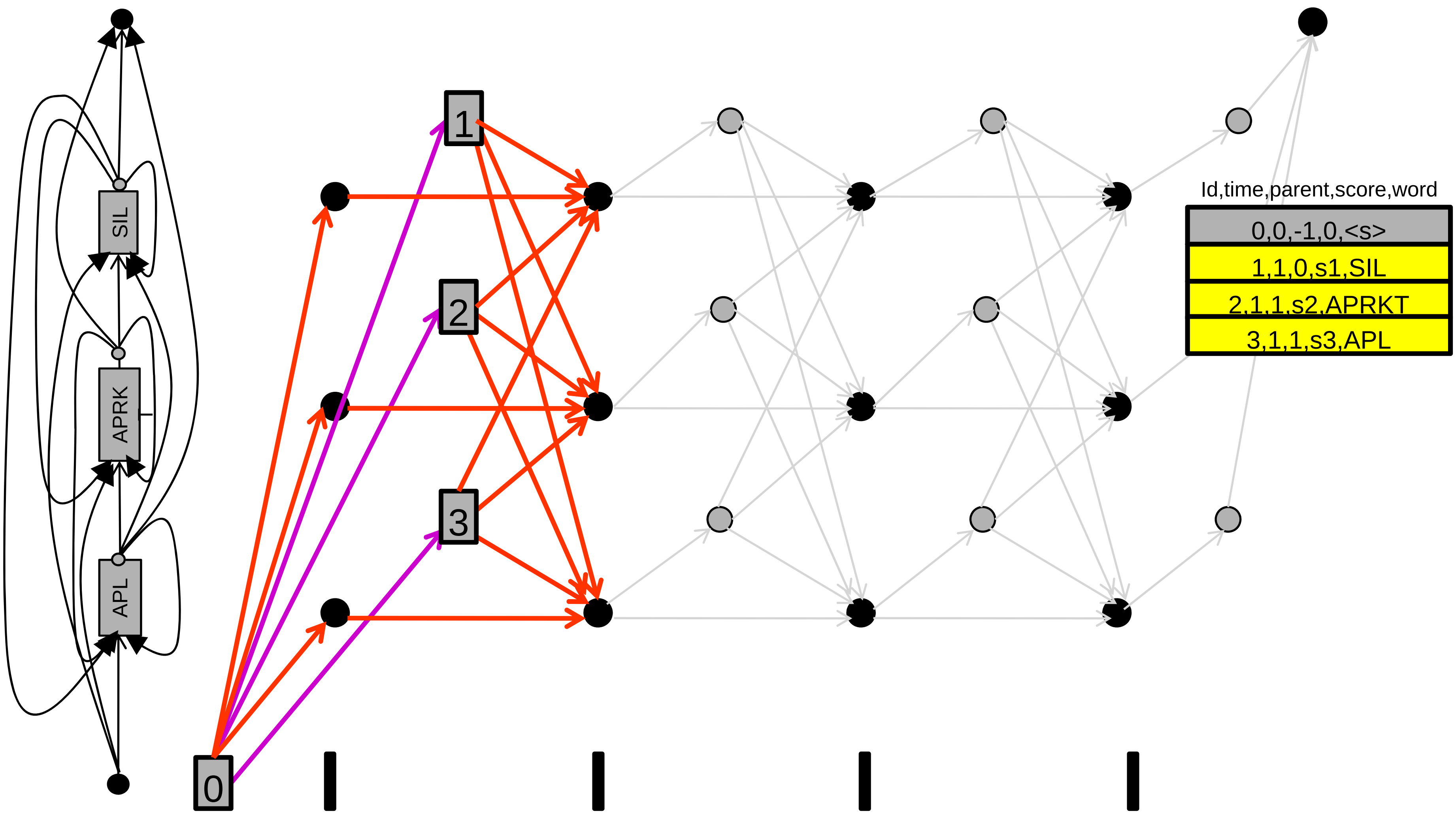
- ***Note the different LM probability terms applied to the arcs***
  - Assuming trigram LM
- The appropriate history to use for the LM probability is obtained from the BPtable

Recognition with flat bigram structure



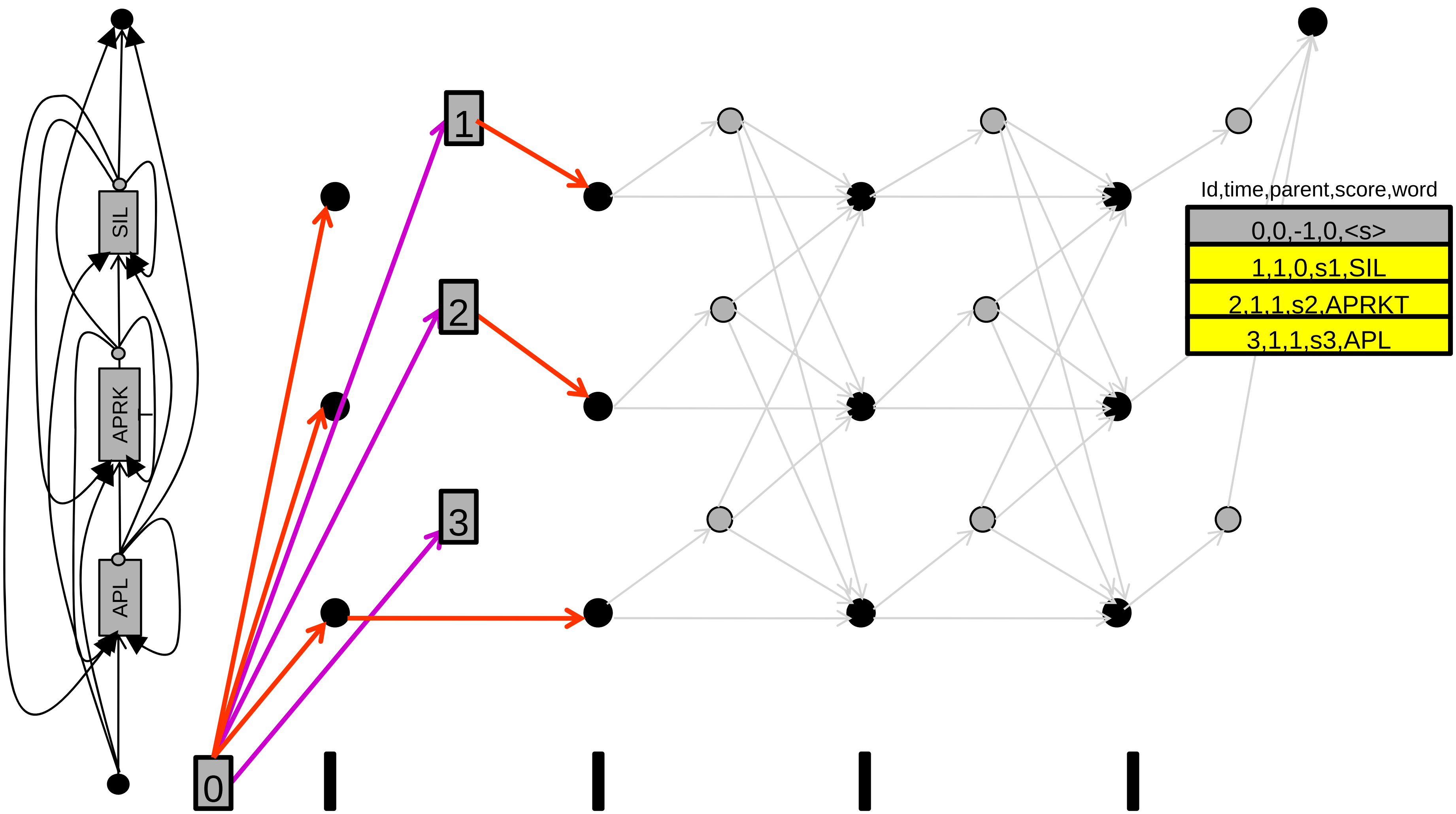
- All cross-word arcs into SILENCE carry the silence penalty
  - Self-transitions within the silence will only carry the self-transition probability for the states of the Silence model

Recognition with flat bigram structure



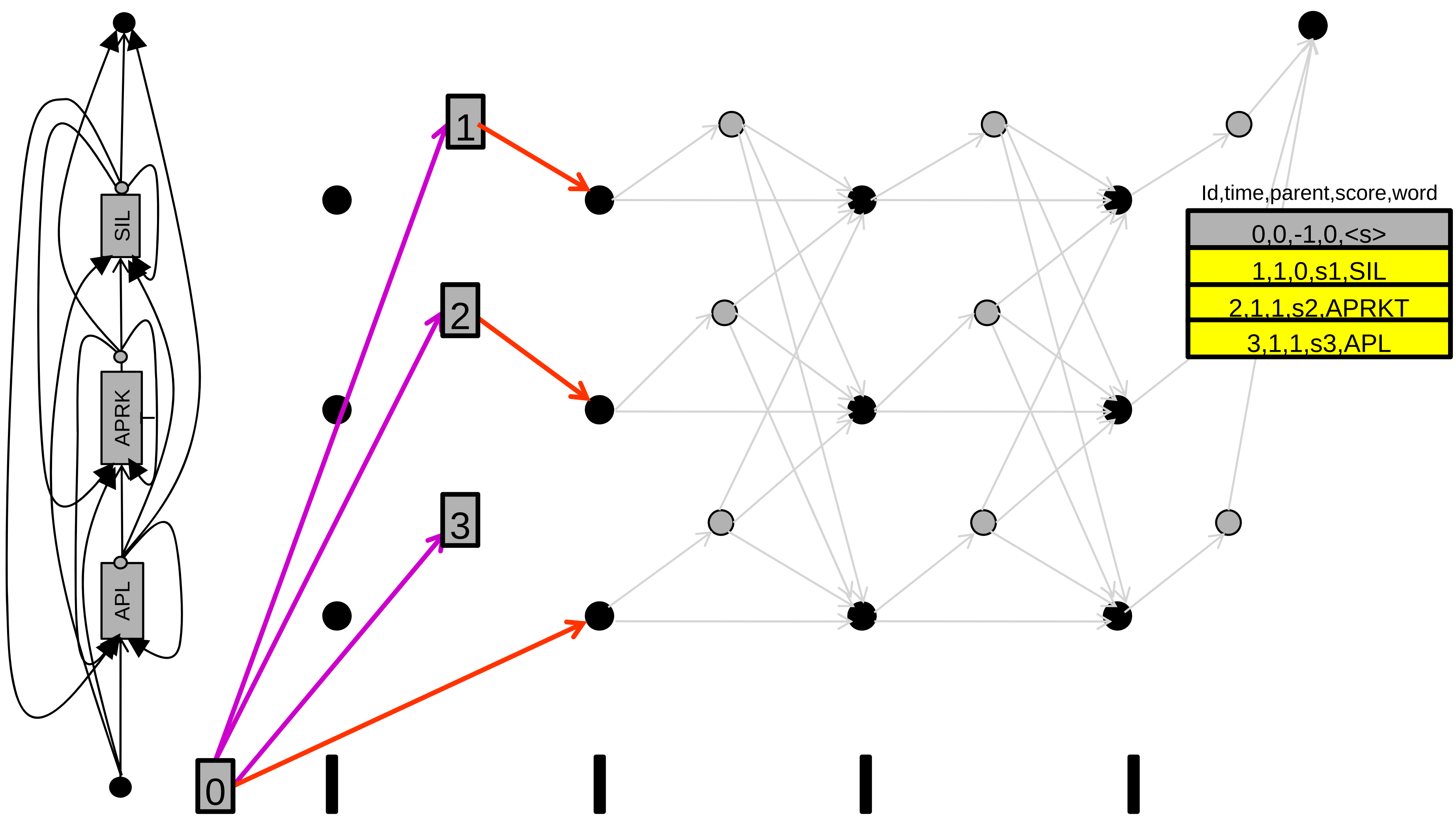
- The actual computation evaluates all of these states in the same timestep

Recognition with flat bigram structure



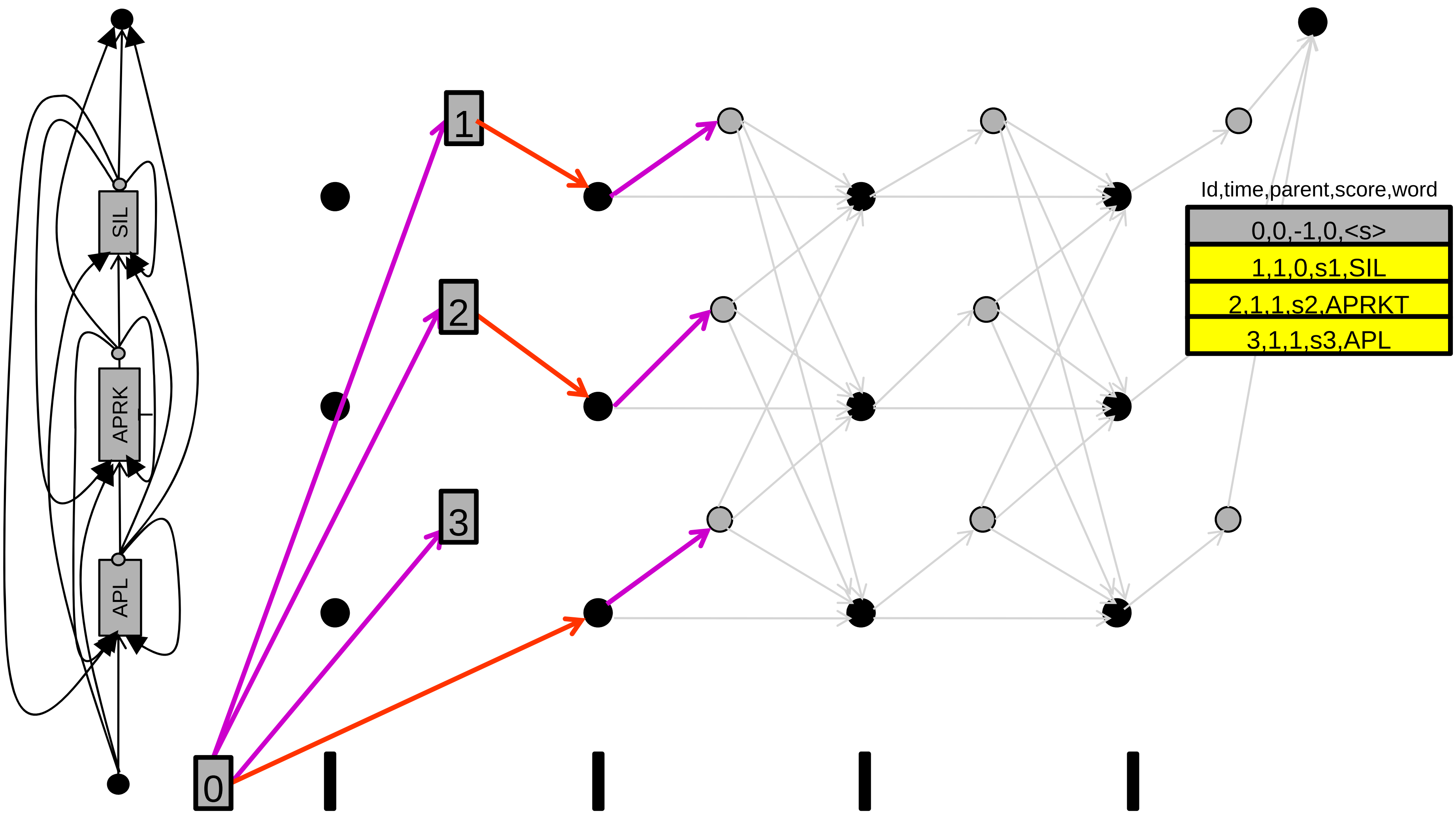
- The actual computation evaluates all of these states in the same timestep

Recognition with flat bigram structure



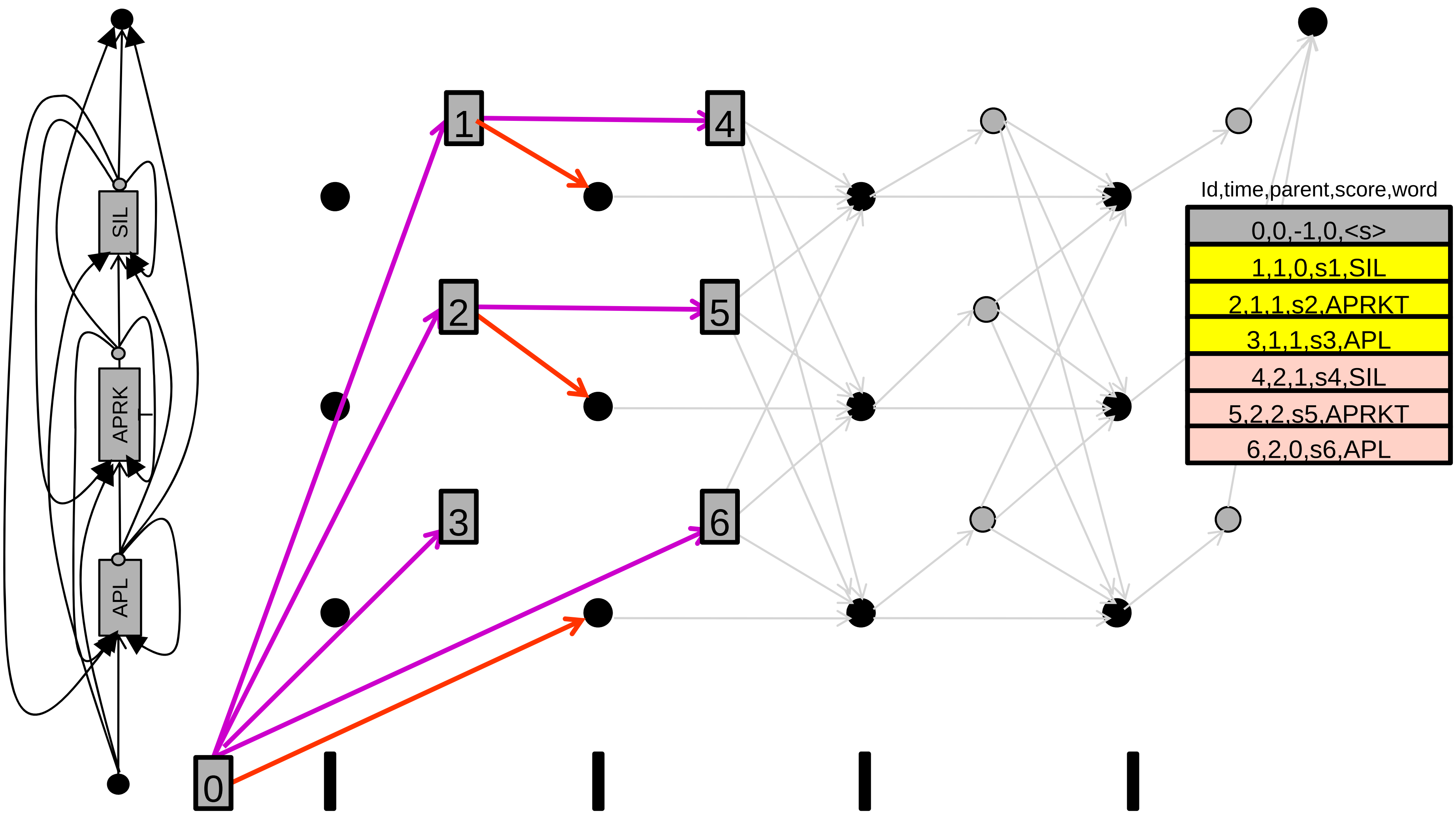
- The actual computation evaluates all of these states in the same timestep

Recognition with flat bigram structure



- Word ending states move into the BP table

Recognition with flat bigram structure



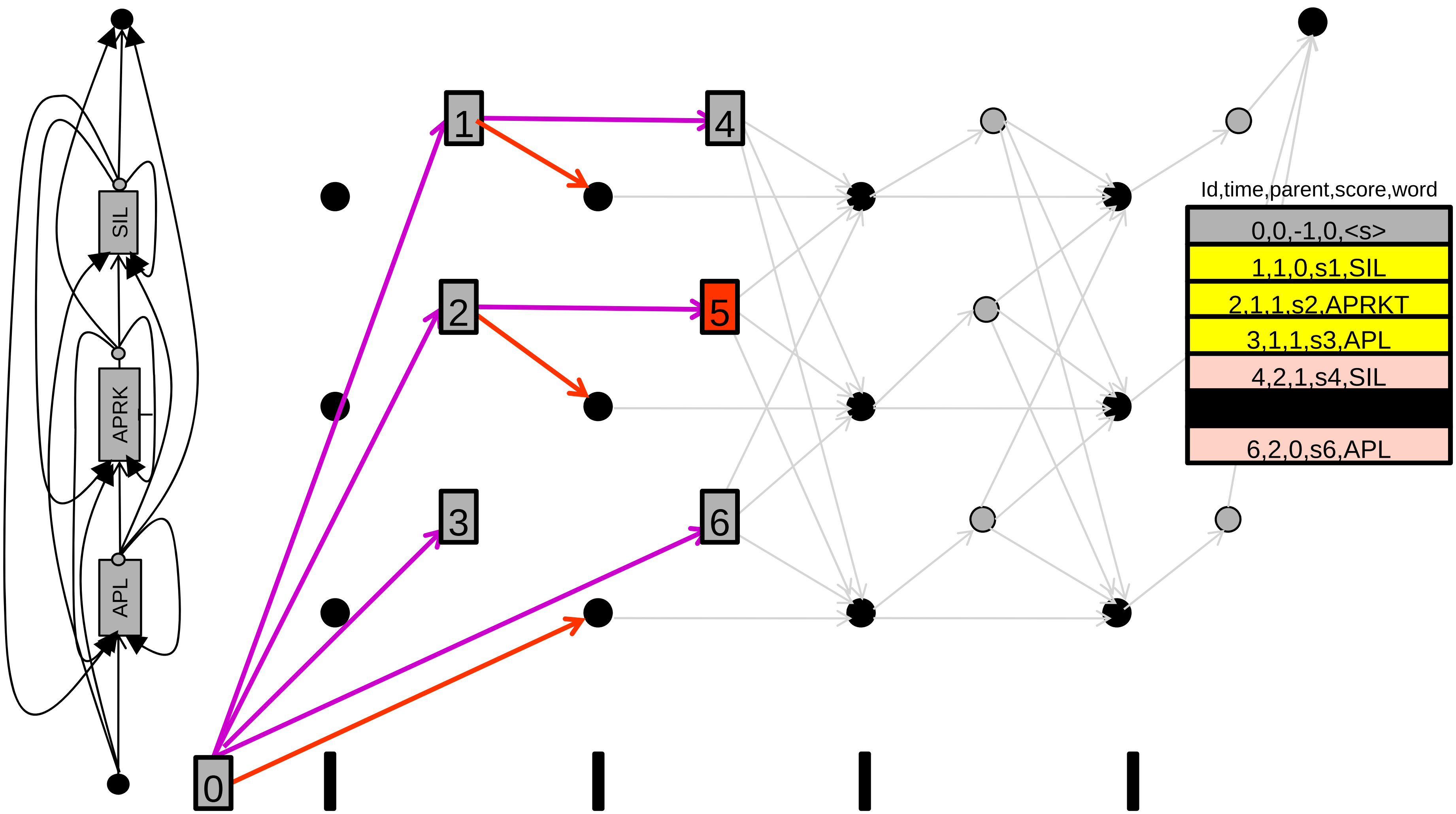
Id,time,parent,score,word
0,0,-1,0,<s>
1,1,0,s1,SIL
2,1,1,s2,APRKT
3,1,1,s3,APL
4,2,1,s4,SIL
5,2,2,s5,APRKT
6,2,0,s6,APL

- Word ending states move into the BP table

Cross-word Pruning

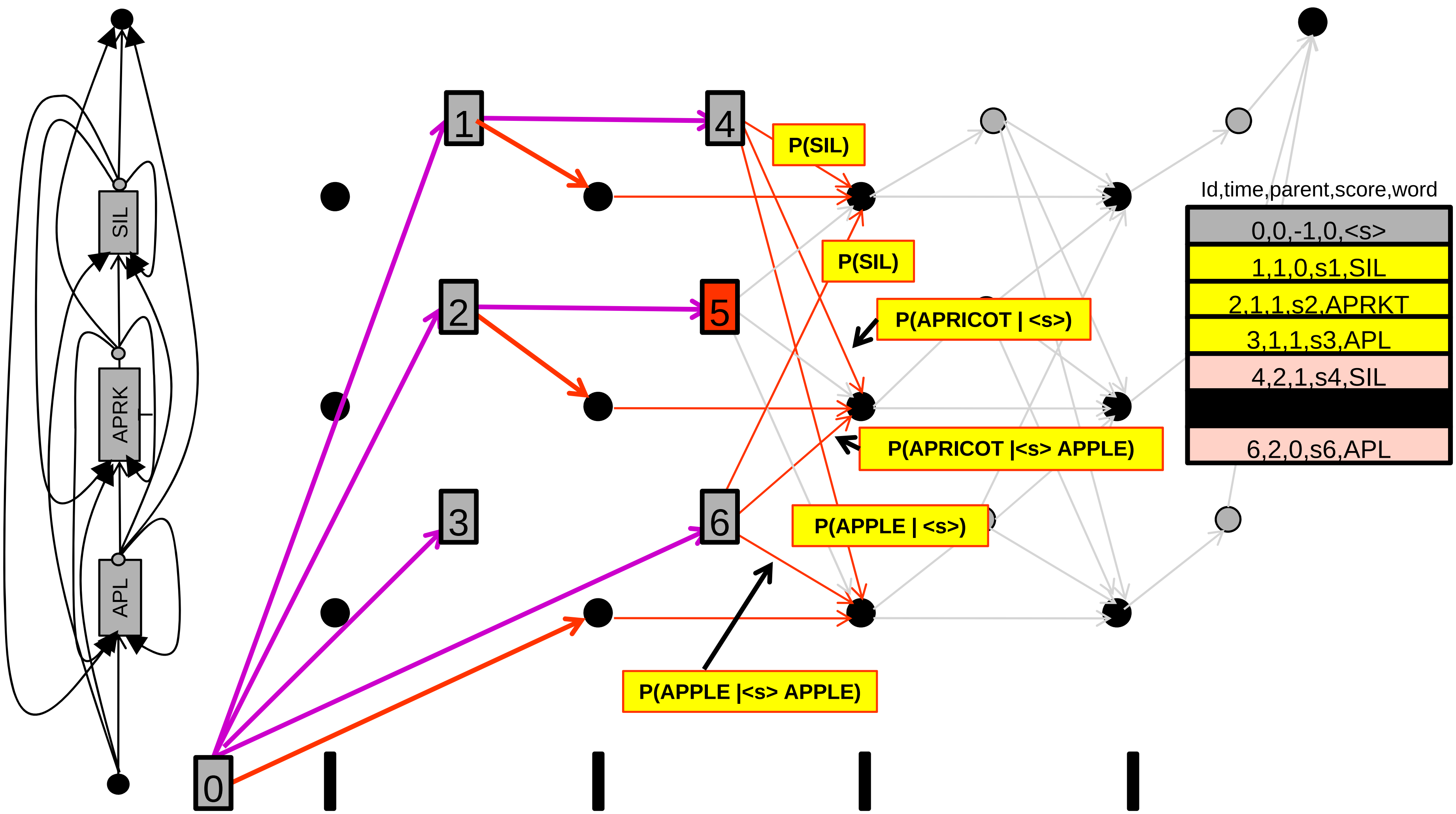
- We can apply a second pruning threshold locally to all entries added to the BP table at a given time
- This is the "***new-word beam***"
  - This is different from the ***state-level beam*** applied across all active states at a given time
  - This is only applied to new word terminations
    - A similar new-word beam may also be applied to the approximate lextree and to correct flat and lex-tree graphs
- In other words, there are TWO different beams we will apply
  - A state-level beam to prune poorly-scoring states
  - A word-level beam to prune poorly-scoring words
- Word beams are typically narrower than state beams

Recognition with flat bigram structure



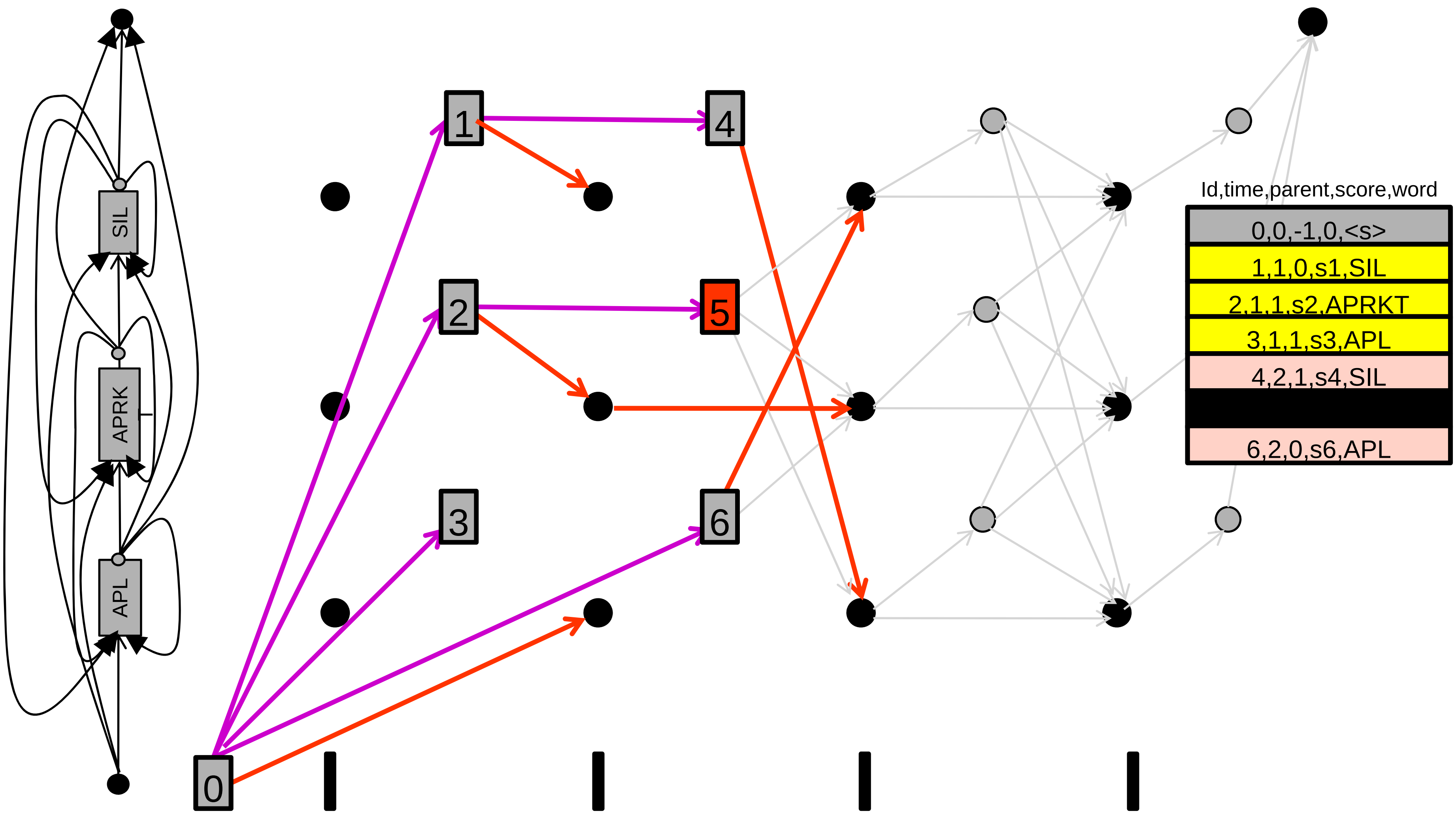
- Pruning the word exits

Recognition with flat bigram structure



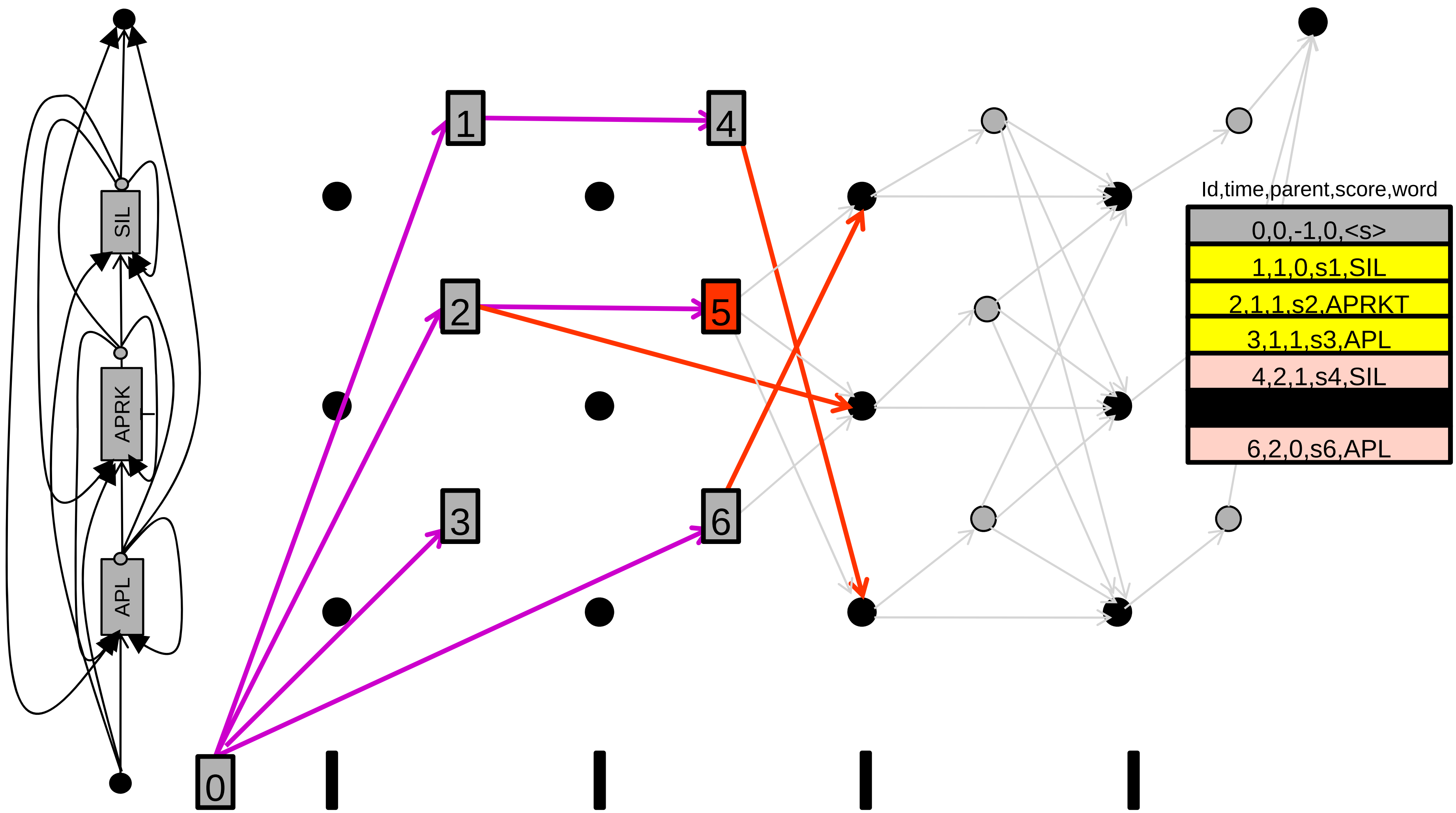
- Note the different LM probabilities applied

Recognition with flat bigram structure



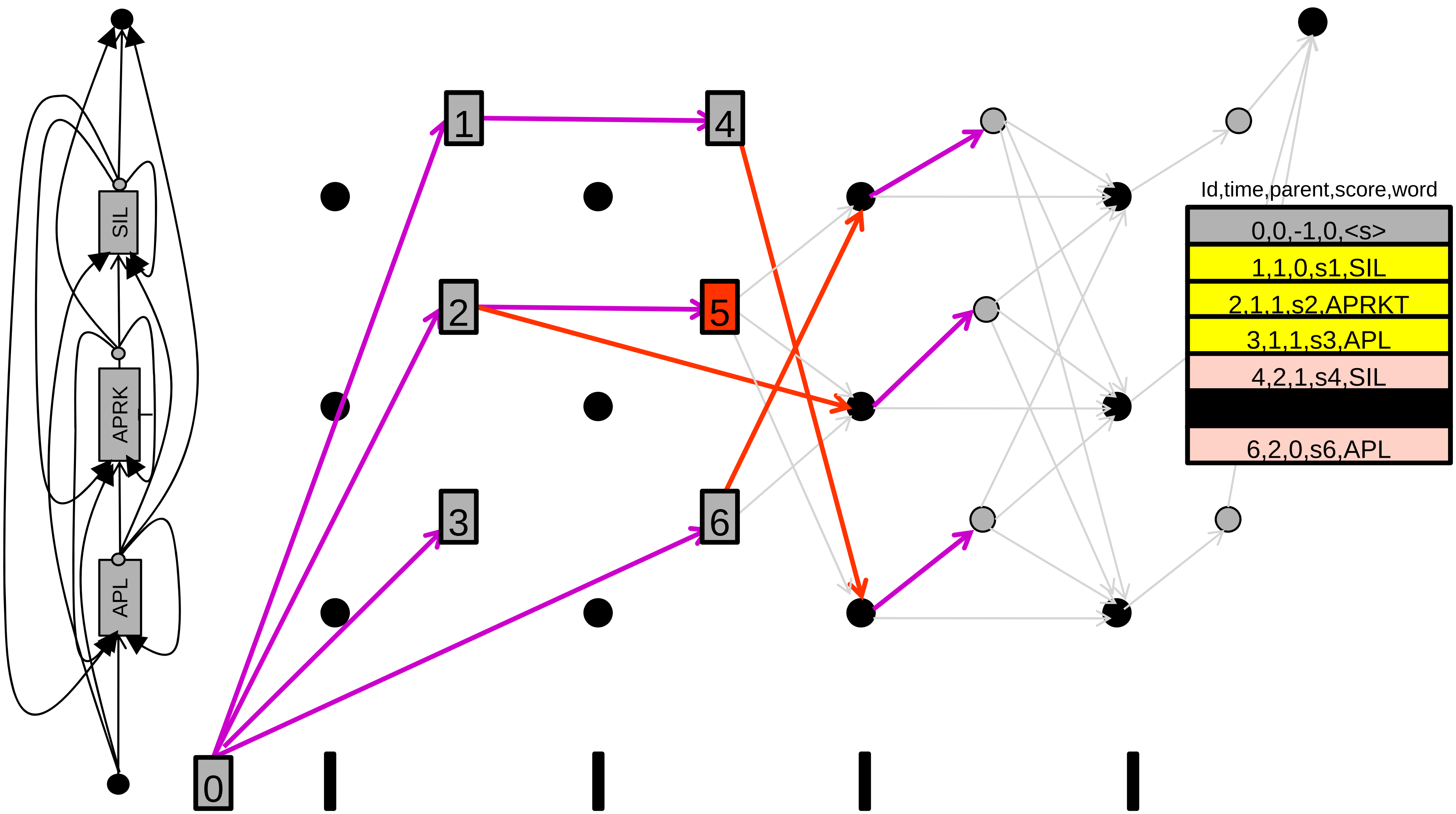
- Select the "winner"

Recognition with flat bigram structure



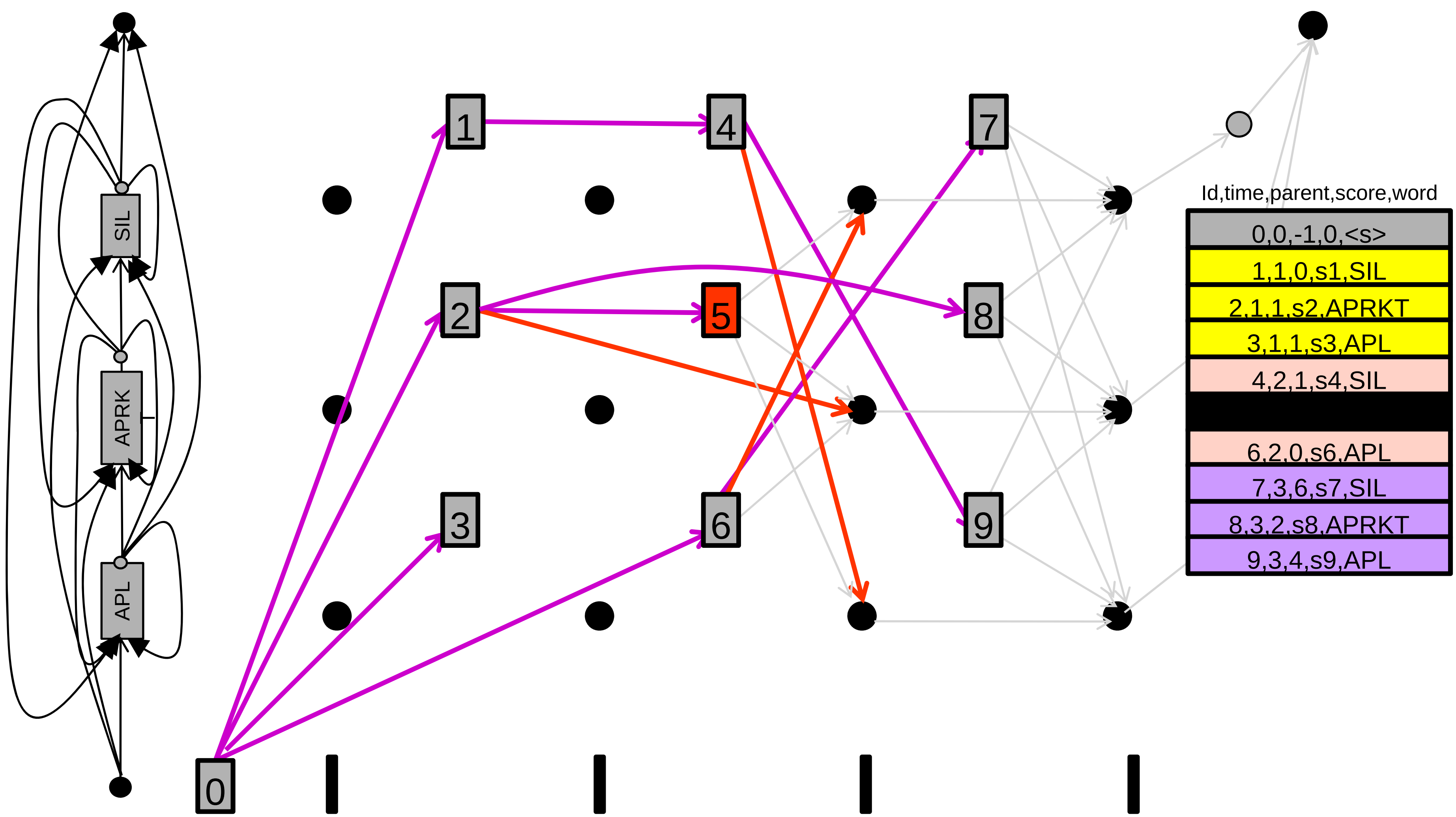
- Note the different LM probabilities applied

Recognition with flat bigram structure



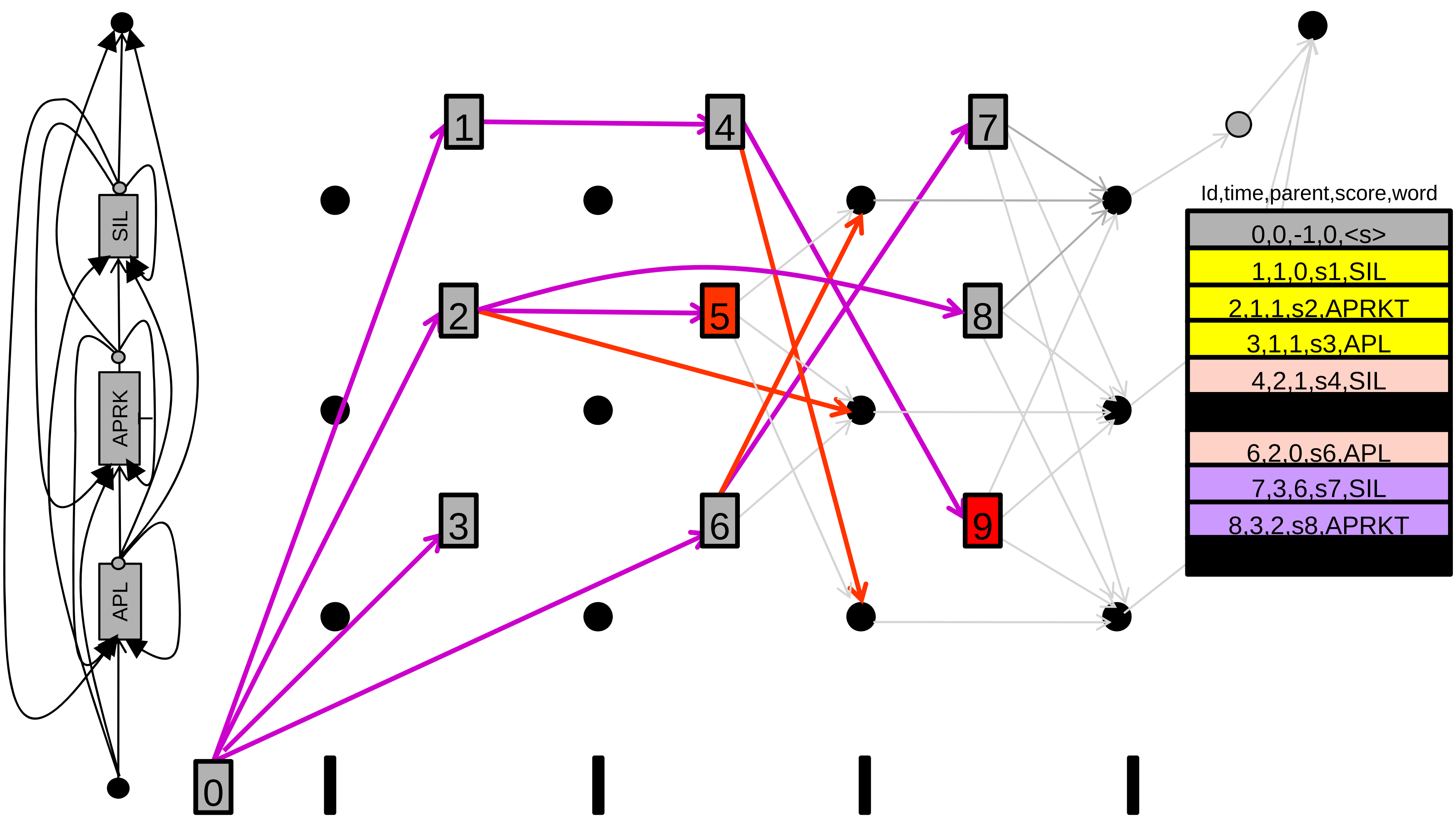
- As before, word ending states move into the BP table

Recognition with flat bigram structure



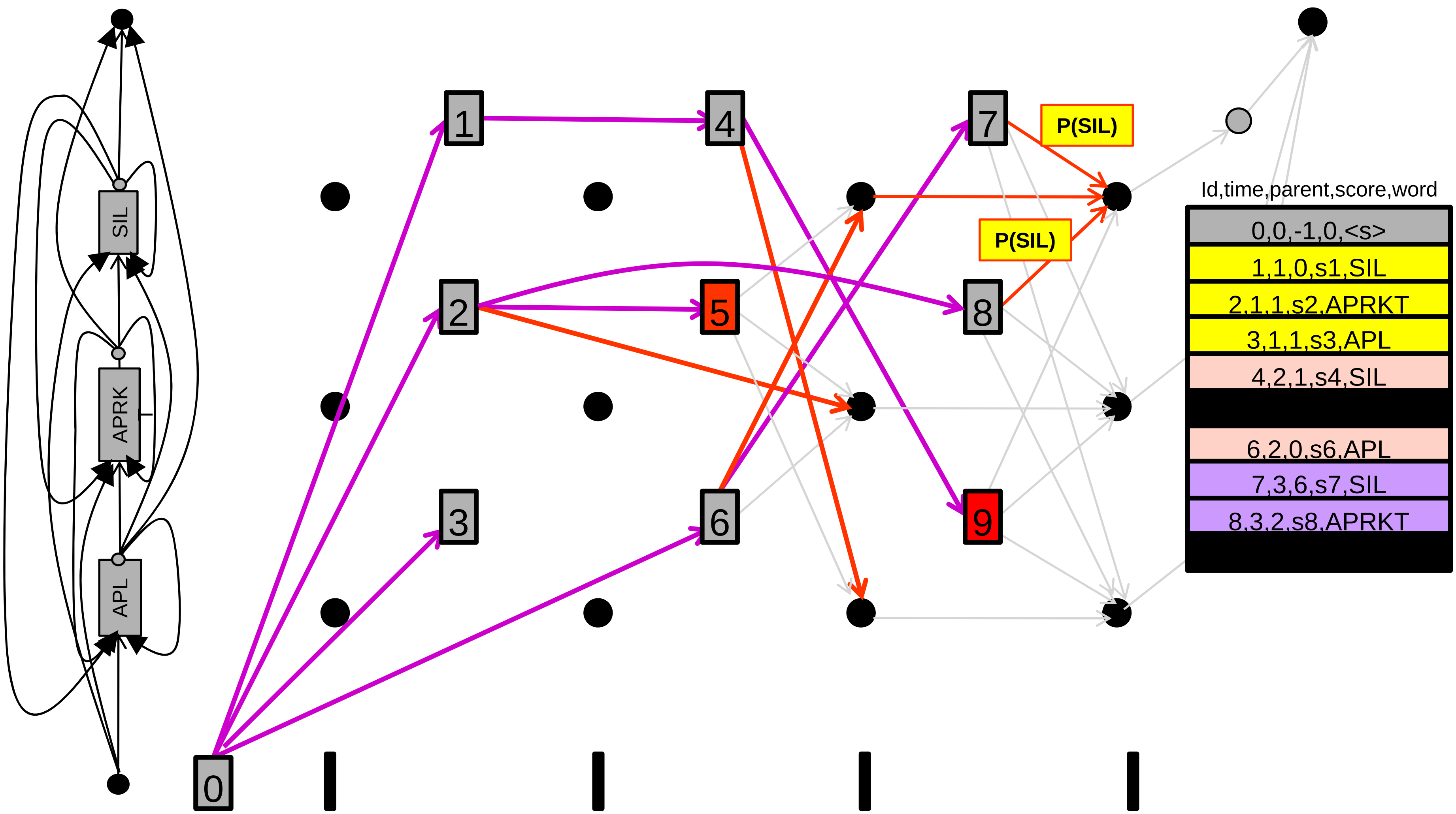
- As before, word ending states move into the BP table

Recognition with flat bigram structure



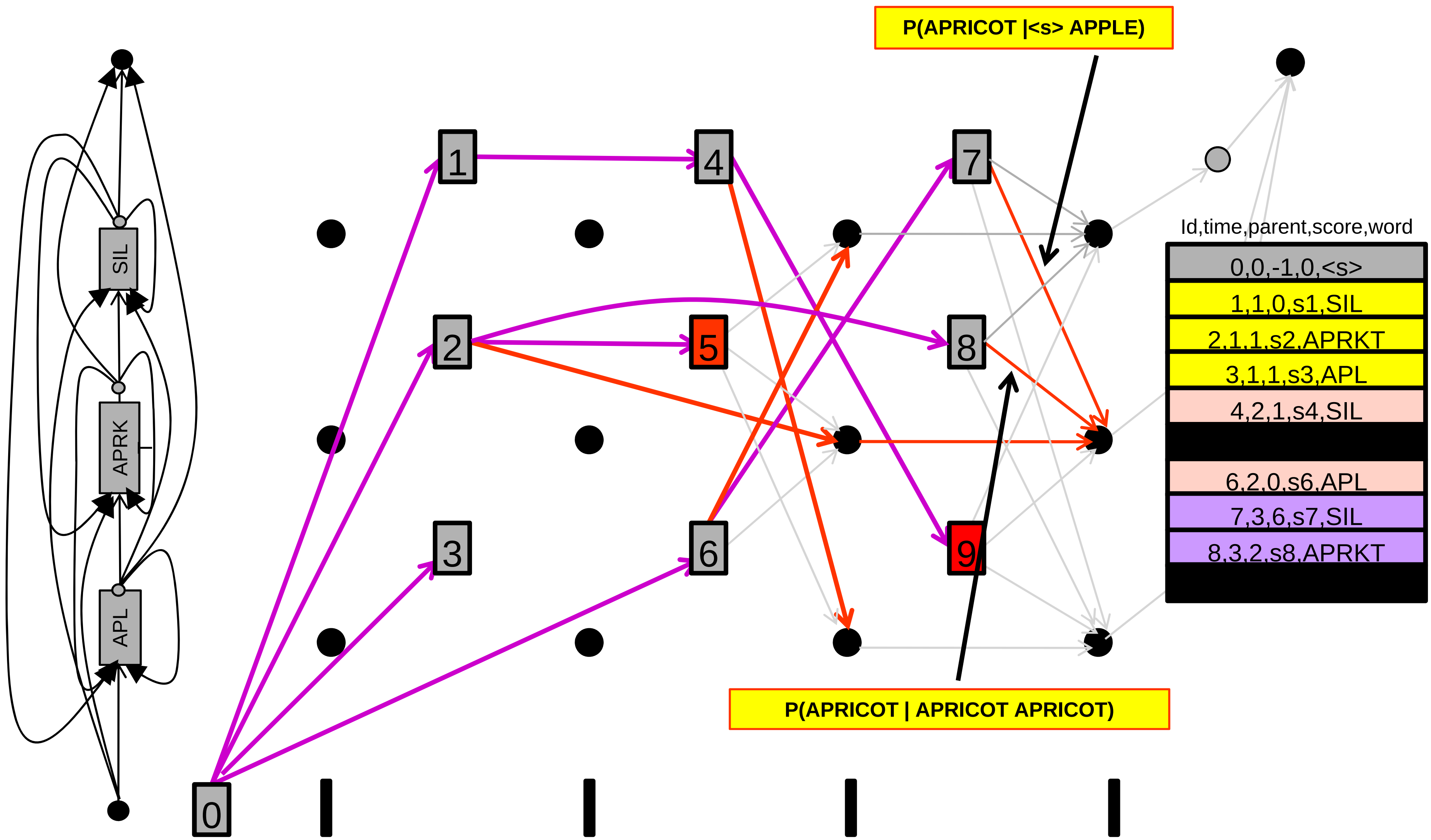
- As before, word ending states move into the BP table
  - **And pruned**

Recognition with flat bigram structure



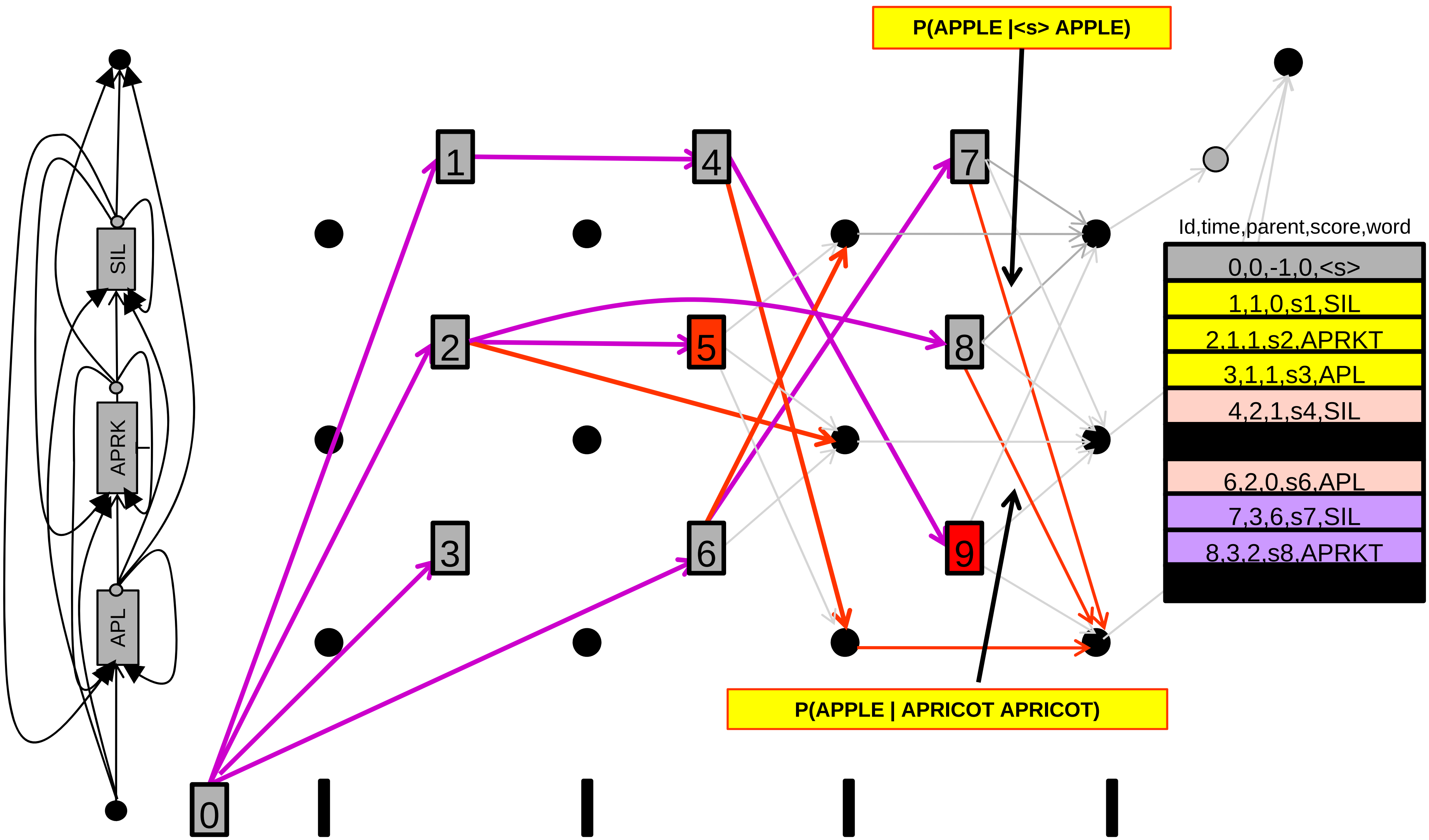
- Note LM probabilities now

Recognition with flat bigram structure



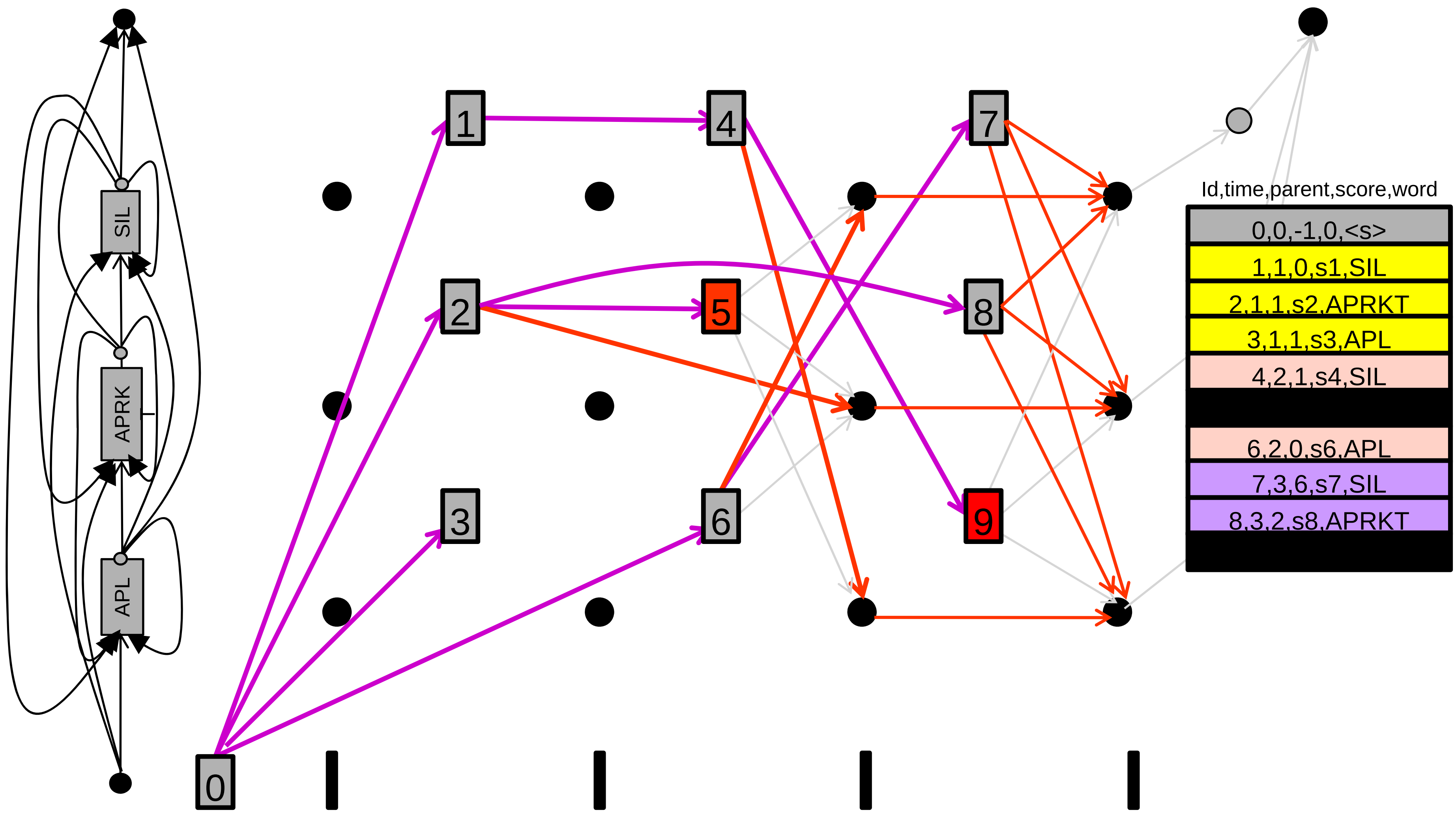
- Note LM probabilities now

Recognition with flat bigram structure



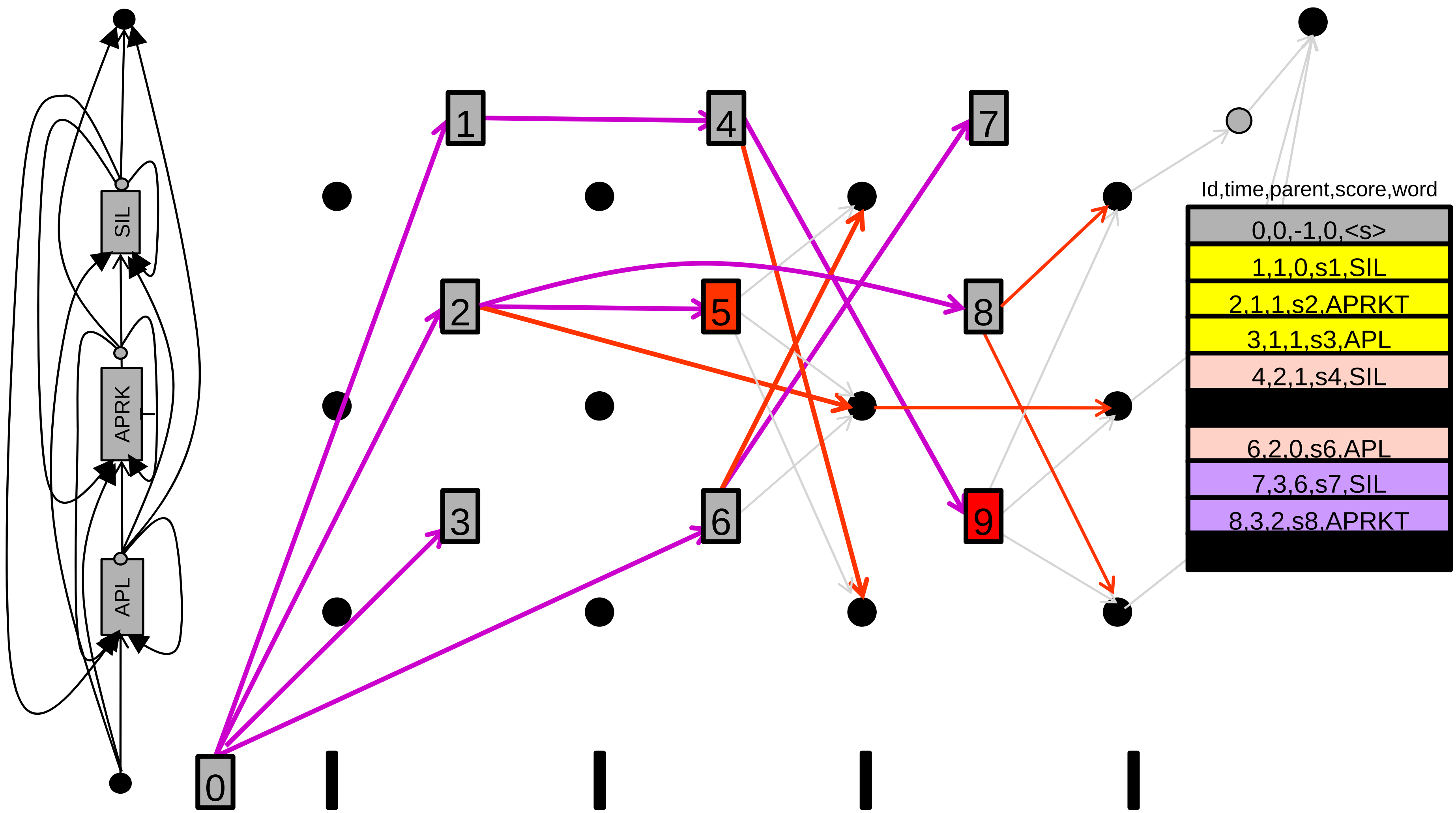
- Note LM probabilities now

Recognition with flat bigram structure



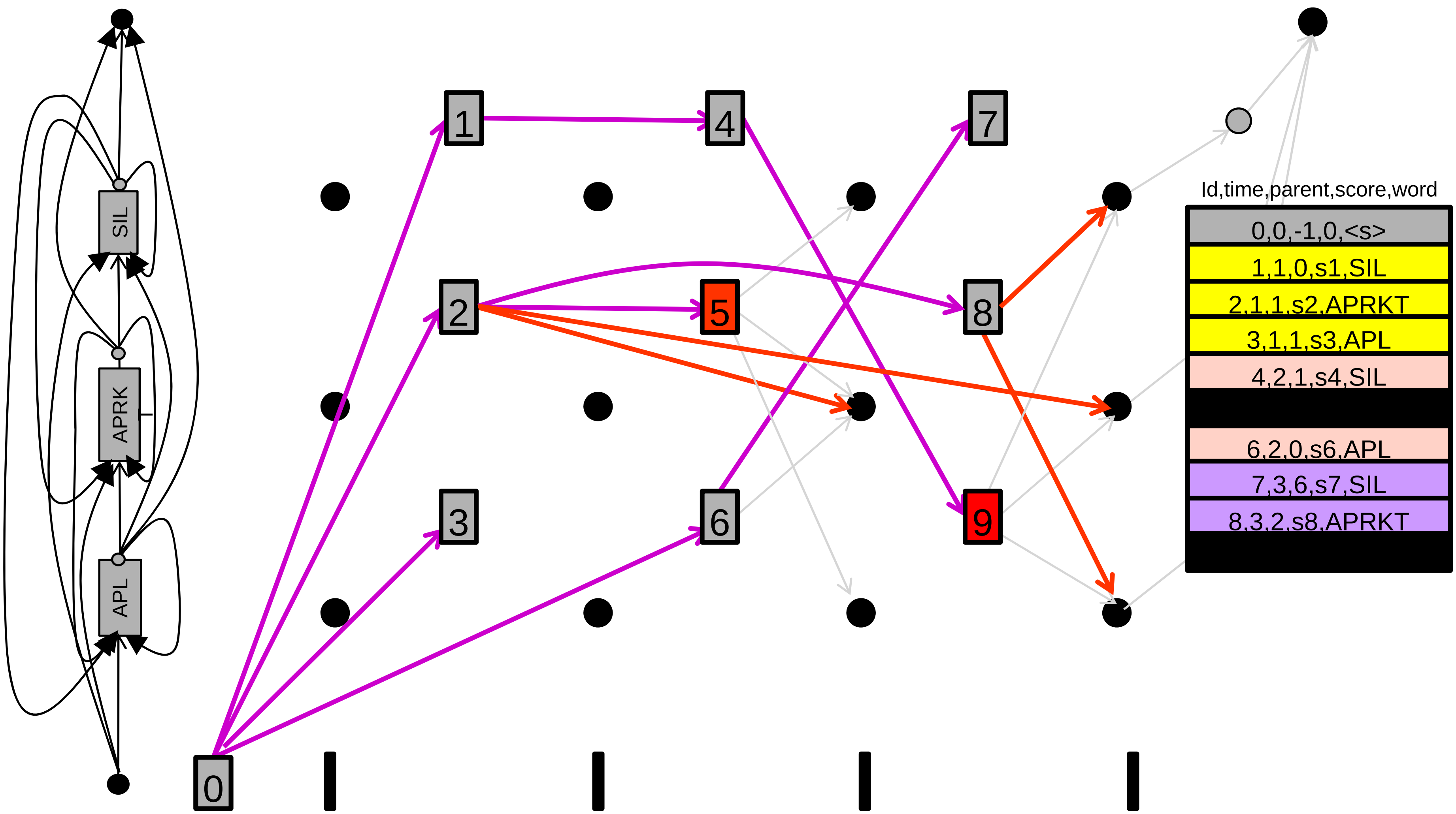
- Note LM probabilities now

Recognition with flat bigram structure



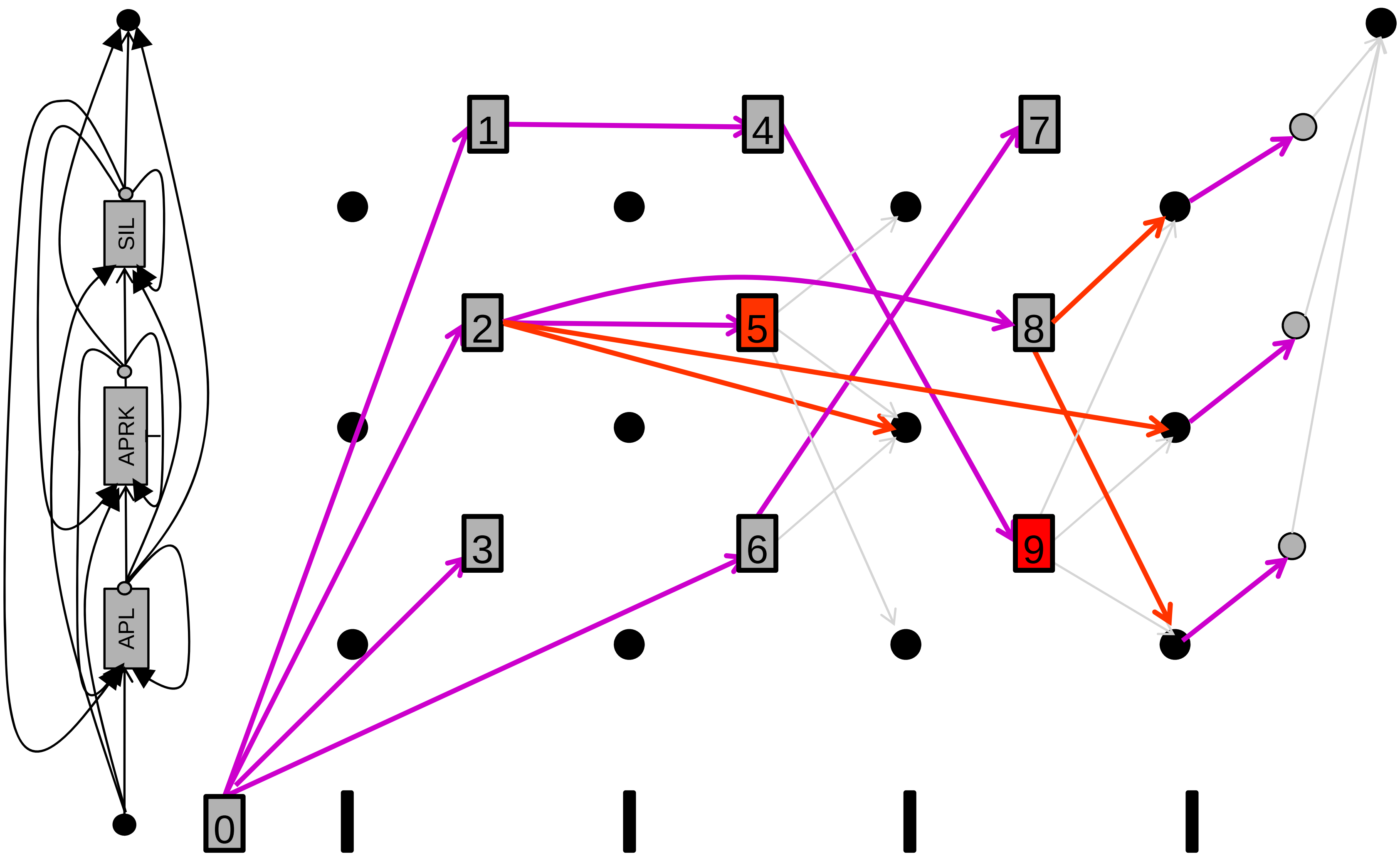
- Note LM probabilities now

Recognition with flat bigram structure



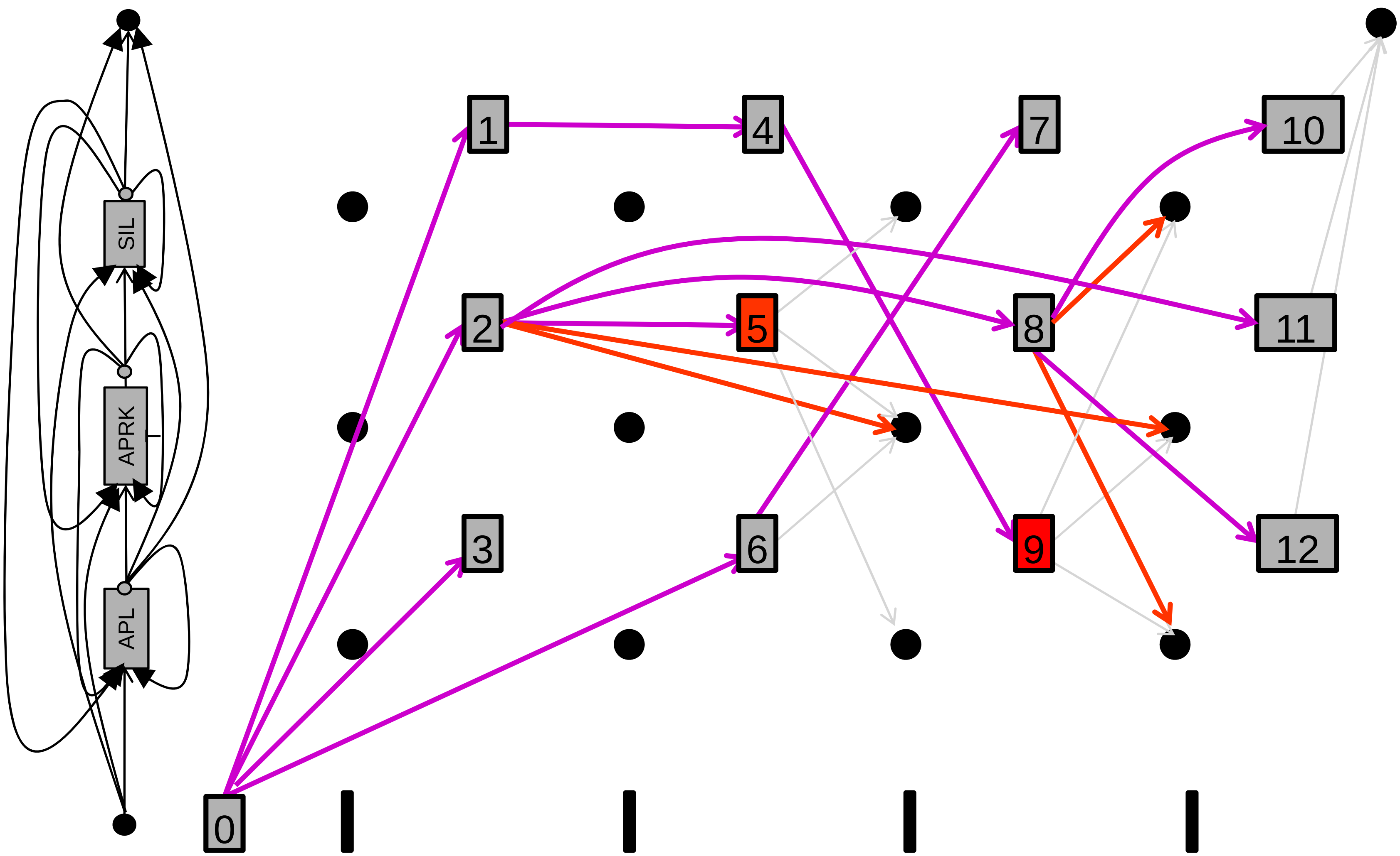
- Note LM probabilities now

Recognition with flat bigram structure



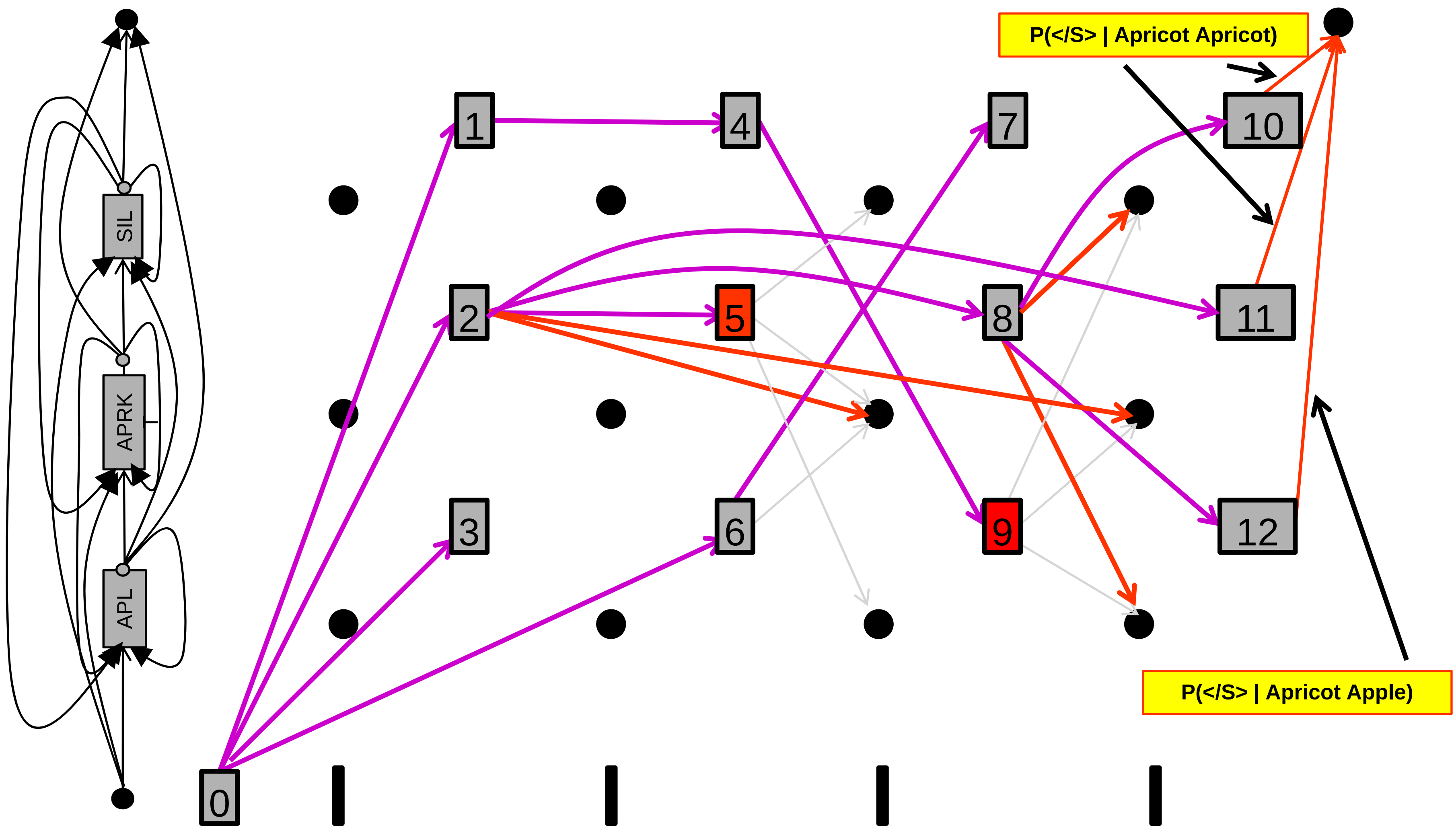
◆ These word exits will end up in the BP table (not shown)

Recognition with flat bigram structure



- These word exits will end up in the BP table (not shown)

Recognition with flat bigram structure



- Note Sentence Ending LM Probabilities Used
  - Note also that multiple hypotheses represent the same word sequence
    - Varying only in the location of silences and word boundaries

Additional Issues

- Several topics left uncovered
  - We lost 3 weeks
- Multi-pass search strategy:
  - The BP table is actually a “lattice”
    - A graph of words
  - A common strategy is to compute a lattice using a bigram LM and to use that as a grammar/graph for recognition using higher-order N-gram LMs
- N-best hypotheses generation
  - How to search the word graph to generate more than one hypotheses
- Confidence: How to assign a “confidence” score to a hypothesis
  - How much we believe the recognizer’s output

Final Assignment

- N-gram based recognition using an approximate decoding strategy
  - Choose between lextree and flat bigram structure
- We are still the equivalent of 4-5 assignments from a nearly-state-of-art system
  - Triphones
  - Lattices
  - Rescoring
  - Nbest
  - Confidence
- In reality, each step would have been very incremental..