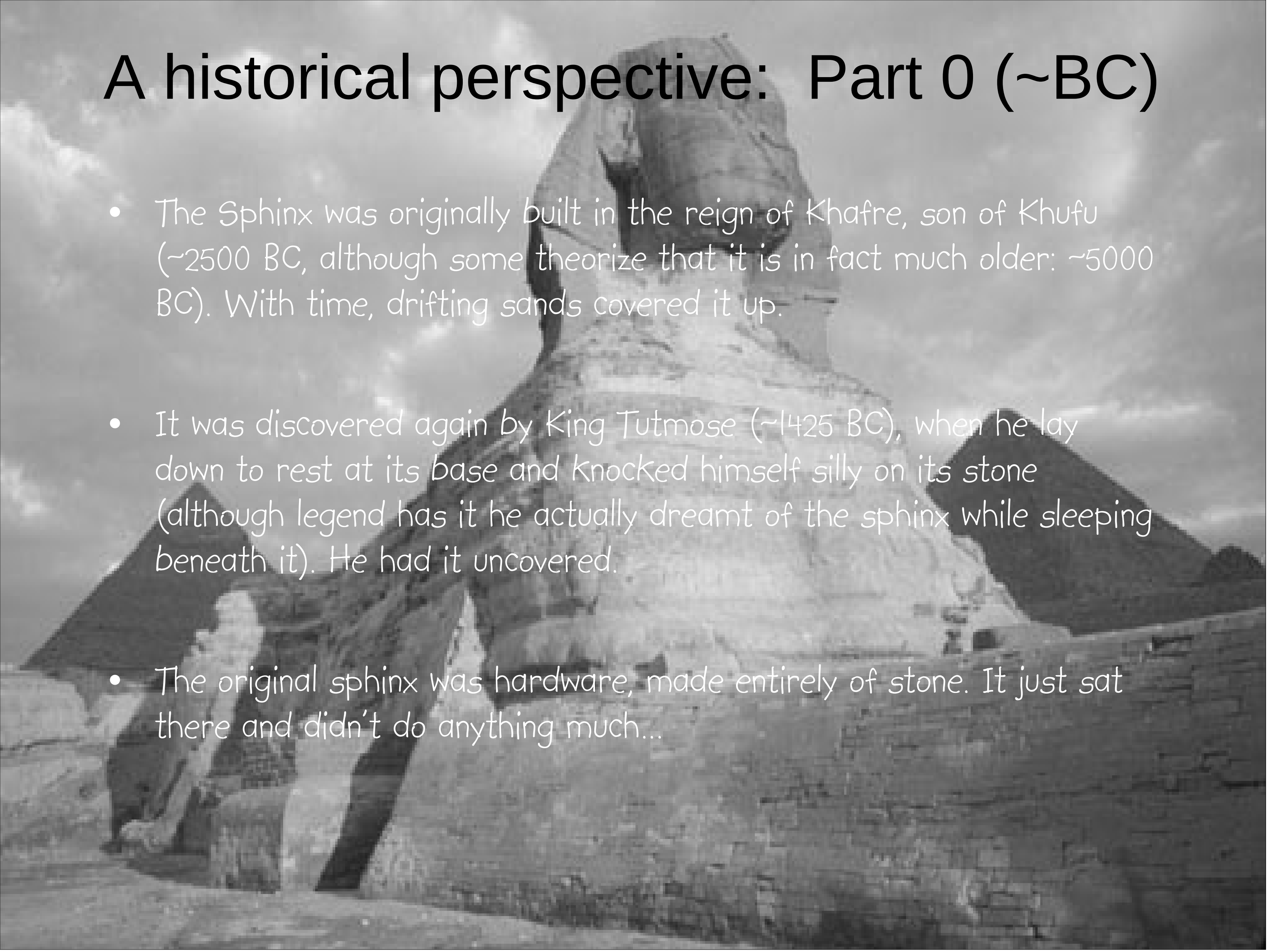
A Historical Perspective

Sphinx 4

MIT Lunch Discussion

December 18, 2002

Sphinx 4 Team



A historical perspective: Part 0 (~BC)

- The Sphinx was originally built in the reign of Khafre, son of Khufu (~2500 BC, although some theorize that it is in fact much older: ~5000 BC). With time, drifting sands covered it up.

- It was discovered again by King Tutmose (~1425 BC), when he lay down to rest at its base and knocked himself silly on its stone (although legend has it he actually dreamt of the sphinx while sleeping beneath it). He had it uncovered.

- The original sphinx was hardware, made entirely of stone. It just sat there and didn't do anything much...

A historical perspective: Part I (1987 AD)

- SPHINX-I
- Many upgrades over the original Sphinx
  - Software, written in C
  - Automatic Speech Recognizer built by Kai-Fu Lee
    - This was revolutionary in the Sphinx world.
  - Continuous speech recognizer
    - The first high-performance speaker-independent large-vocabulary continuous speech recognition system
  - Discrete HMMs
    - 3 Codebooks of size 256
  - Simple word-pair grammars
  - Generalized triphones

  - Accuracy of ~90% on Resource Management
    - Real time on Sun3 or DEC 3000 (top of the line in 1988)

A historical perspective: Part II (1992 AD)

- SPHINX-II
  - Built by Xuedong Huang
- Semi-continuous HMMs.
  - 4 feature streams, 4 codebooks of distributions
  - Basic cepstral feature assumed 13 dimensional
- State tying with senones
  - Using CART-based decision trees
- 5-state HMM topology
- N-gram language models
- Fast lextree decoder (fbs8) for live decoding
- Accuracy of ~90% on WSJ

A historical perspective: Part III (1996 AD)

- SPHINX-III
    - Built by Eric Thayer and Mosur Ravishankar
- Fully-continuous (and semi-continuous) HMMs.
    - Flexible feature vectors, single or 4-stream
    - Flexible HMM topology
- N-gram language models
- State tying with senones
    - Using CART-based decision trees
- Two decoders
    - Decoder1 : Flat search (slow)
        - Upto trigrams (with “pseudo trigram” search)
    - Decoder 2: Lextree search (fast)
        - Any Ngram (in principle). Trigrams implemented
        - Subvector quantization based Gaussian selection

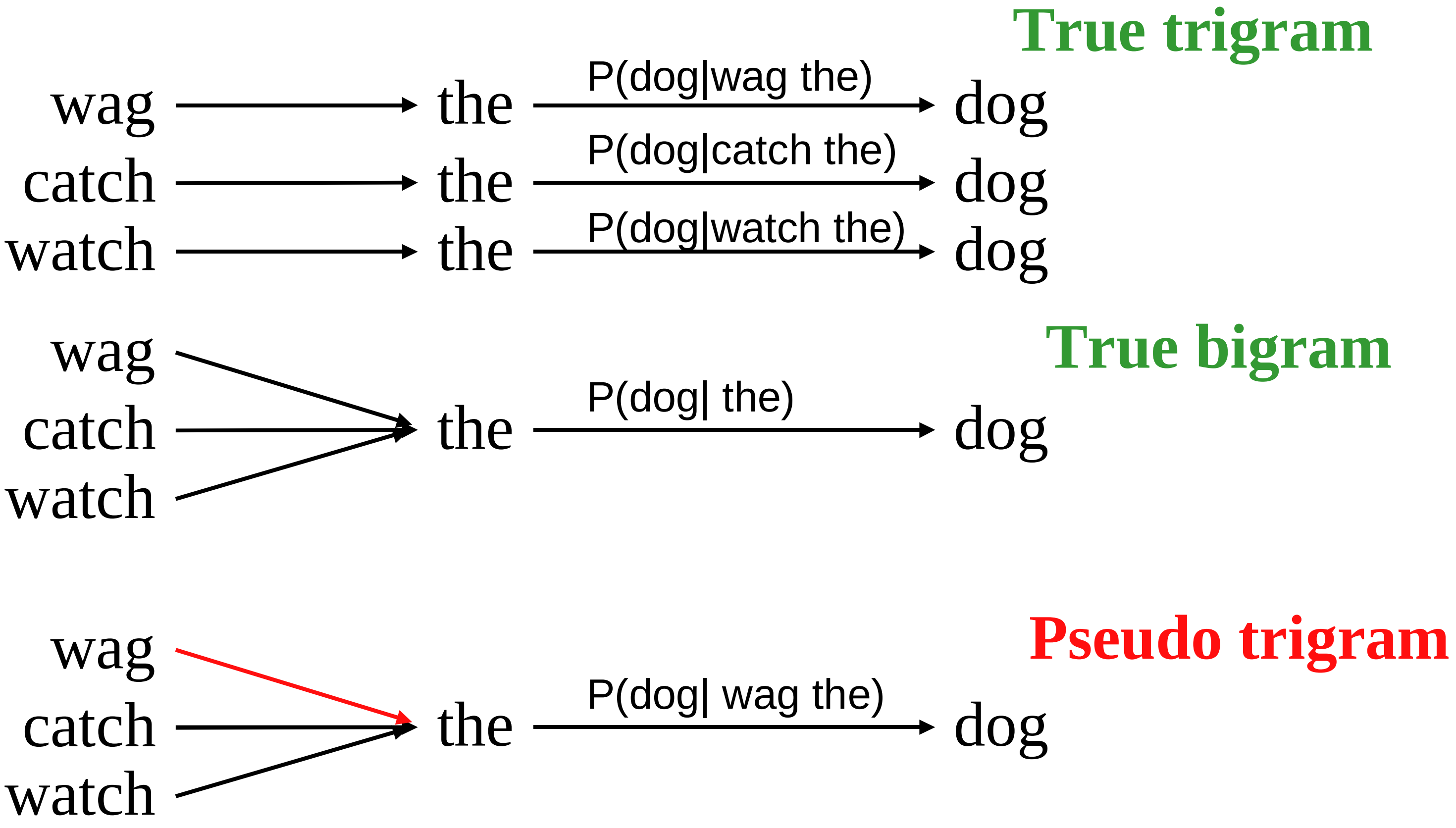
- WER of ~19% (first pass) on BN (1998 eval set)

A historical perspective: A new millennium

- SPHINX-III has limitations
- Only triphone contexts
- Only Ngram models
  - No CFG / FSA / SCFG models allowed
- Uniform HMM topology for all sound units
  - HMMs for all sound units to have the same no. of states
- Uniform acoustic model structure
  - All state output distributions to have same no. of Gaussians
- Features can be combined only at state level
  - State-feature synchrony enforced
- Decoders were suboptimal :::

Sphinx 3 decoders: Flat decoder

- Each word has its own HMM
  - Computation and memory intensive
- Only a “pseudo-trigram” search:

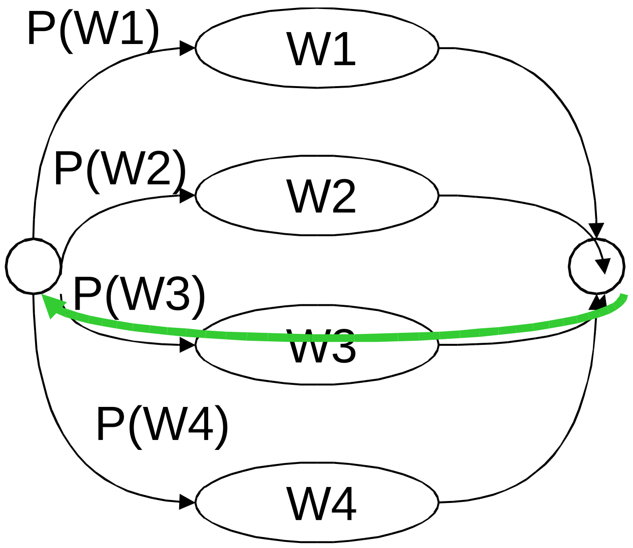


Flat decoder: Why use pseudo trigram?

- In a true bigram search HMMs are required for all D words in the dictionary
  - D is the vocabulary size
- In a true trigram search D^2 word HMMs are needed
  - D copies of the HMM for every word in the vocabulary
- In a pseudo-trigram the maximum number of HMMs remains D
  - The accuracy is better than that achieved with bigrams
  - Still handicapped with respect to a true trigram search
    - Although in most tasks the difference is negligible
  - Must construct DAGs from lattices and rescore with true trigrams to get better approximation to true trigram decoding

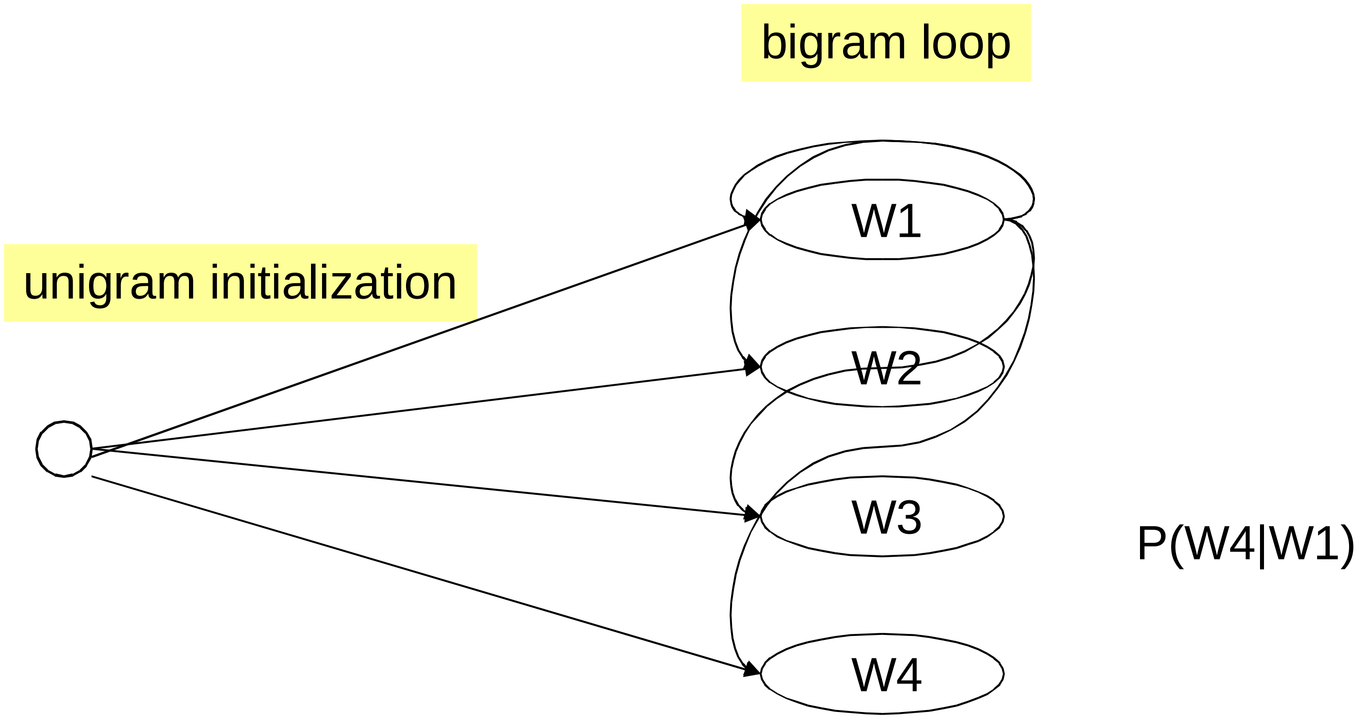
Simplified representation of unigram-based decoding

- The probability of a word is independent of preceding words



Bigram-based decoding with a simple four-word vocabulary (including initial and terminal stages)

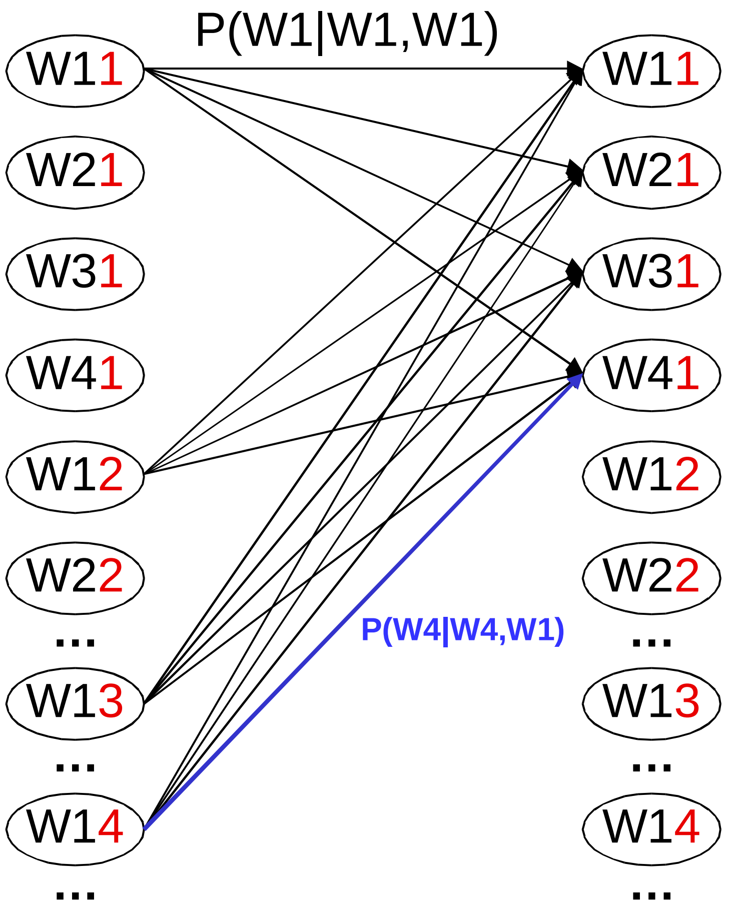
- The probability of a word is dependent on the preceding word



Trigram-based decoding with a simple four-word vocabulary (partial view)

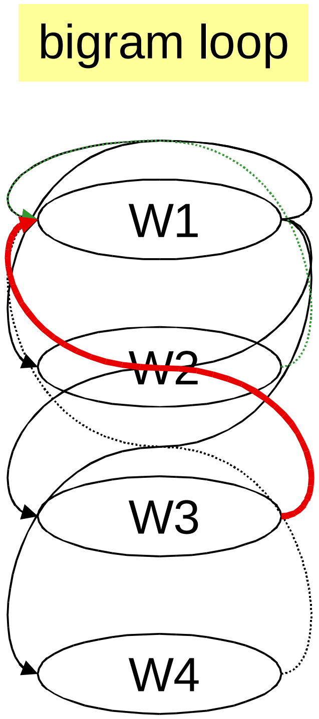
- N instances of each word created, where N is the size of the vocabulary

Wij is an HMM for word Wi that can only be accessed from HMMs for word Wj. E.g. W12 is the HMM for word W1 that can only be used when the previous word was W2



Psuedo-trigram-based decoding with a simple four-word vocabulary

- The probability of a word is dependent on the preceding word and its best predecessor



P(W4|W3,W1)

Probability for a link out of a word uses that word and its best predecessor (maintained in a backpointer table) as context

Sphinx 3 decoders: Fast lextree decoder

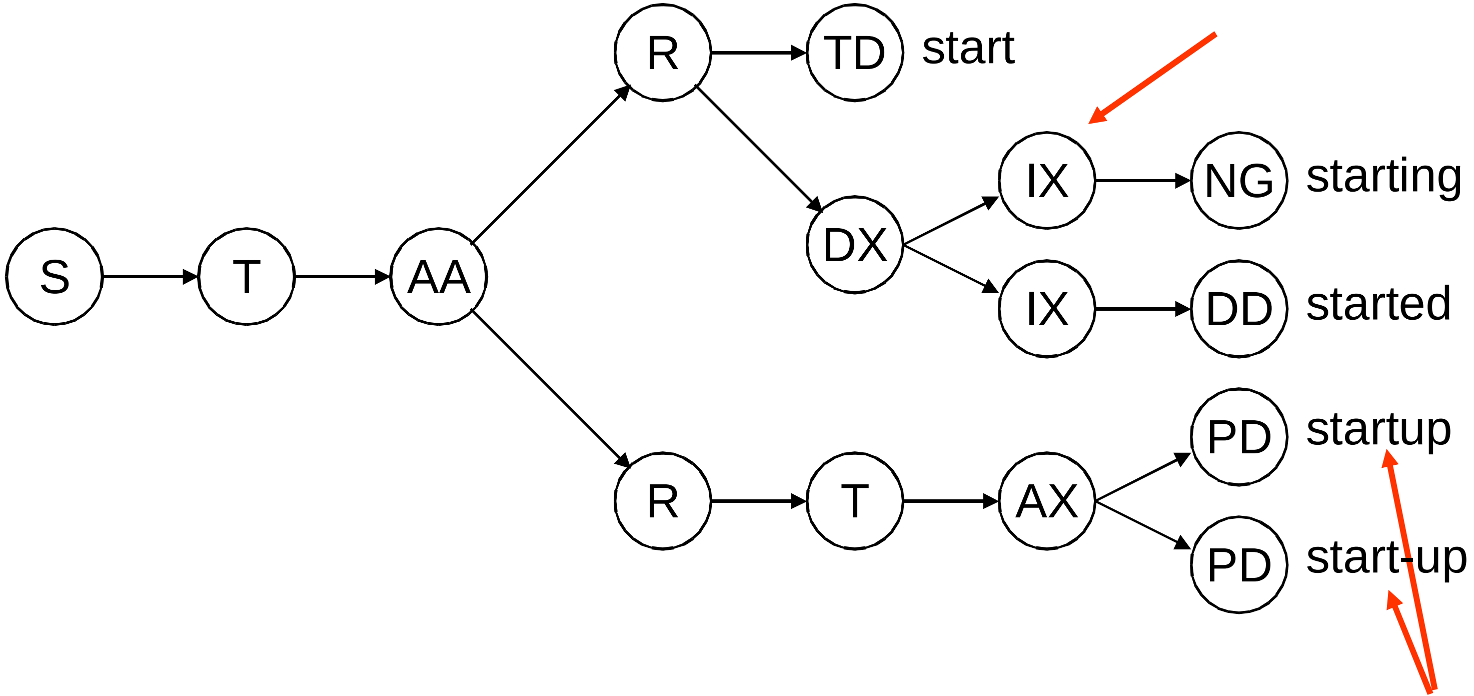
- Also makes compromises
  - Uses a fixed number of lextrees
  - The number is configurable.
  - 3 lextrees has been found to give the best results

- A true lextree decoder for trigram decoding needs D^2 lextrees
  - This represent D^3 word HMMs

- The sphinx decoder uses static lextrees to generate a backpointer table
  - Ngram contexts from backpointer tableSuboptimal with respect to true Ngram lextree decoding
  - Much more efficient in terms of resources

Lextree

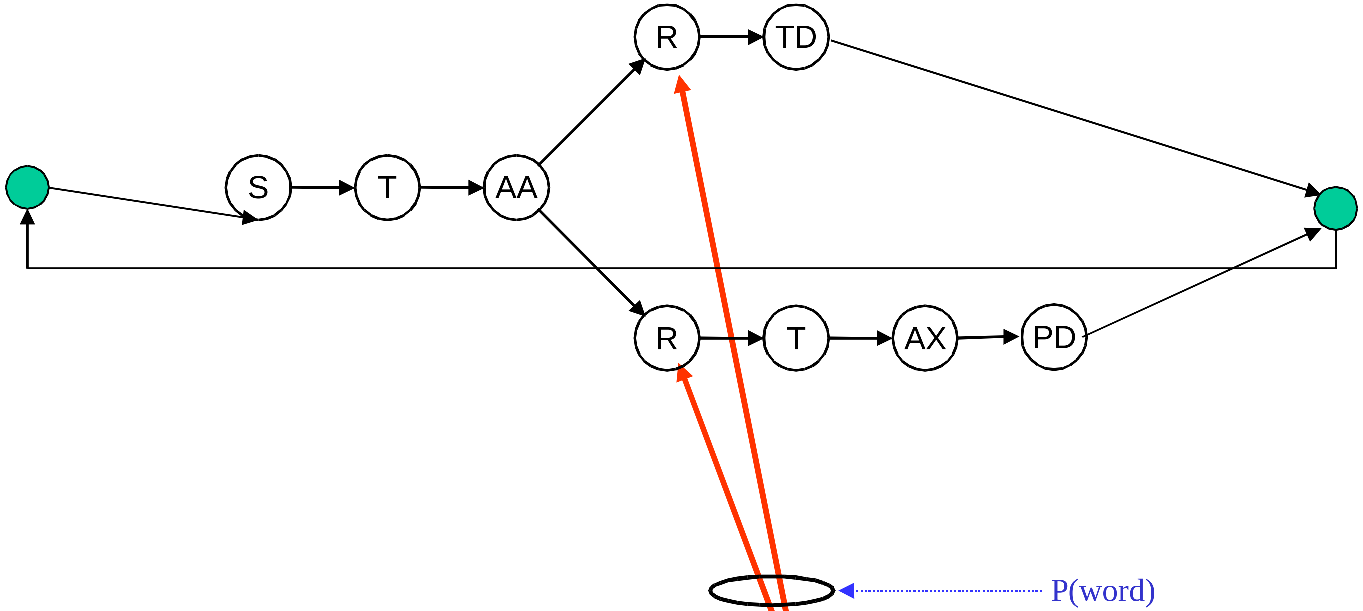
- The probability of a word is obtained deep in the tree
  - Example assumes triphone models



Word identity only known here

Different words with identical pronunciations must have different terminal nodes

Unigram Lextree Decoding

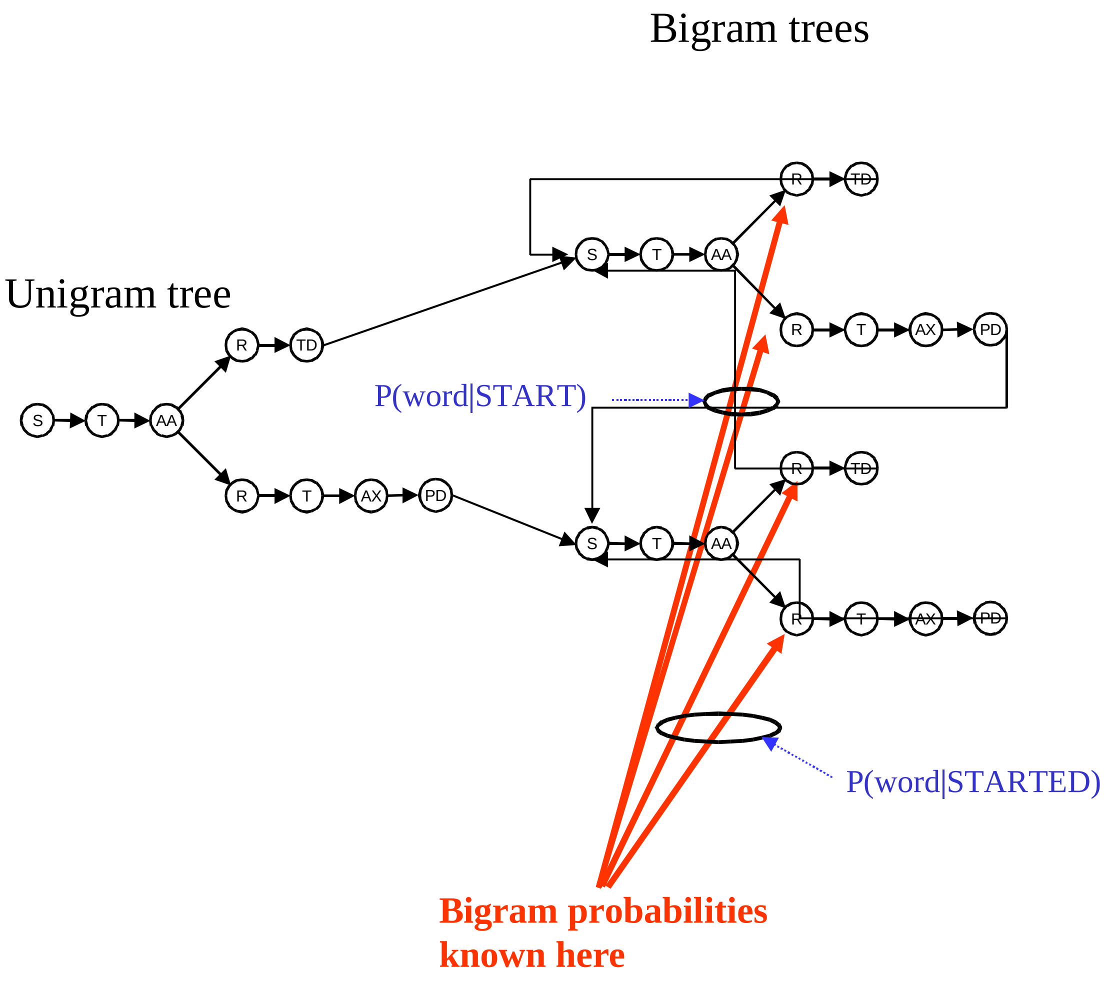


P(word)

Unigram probabilities known here

Rita Singh, July 28, 2002

Bigram Lextree Decoding



Bigram trees

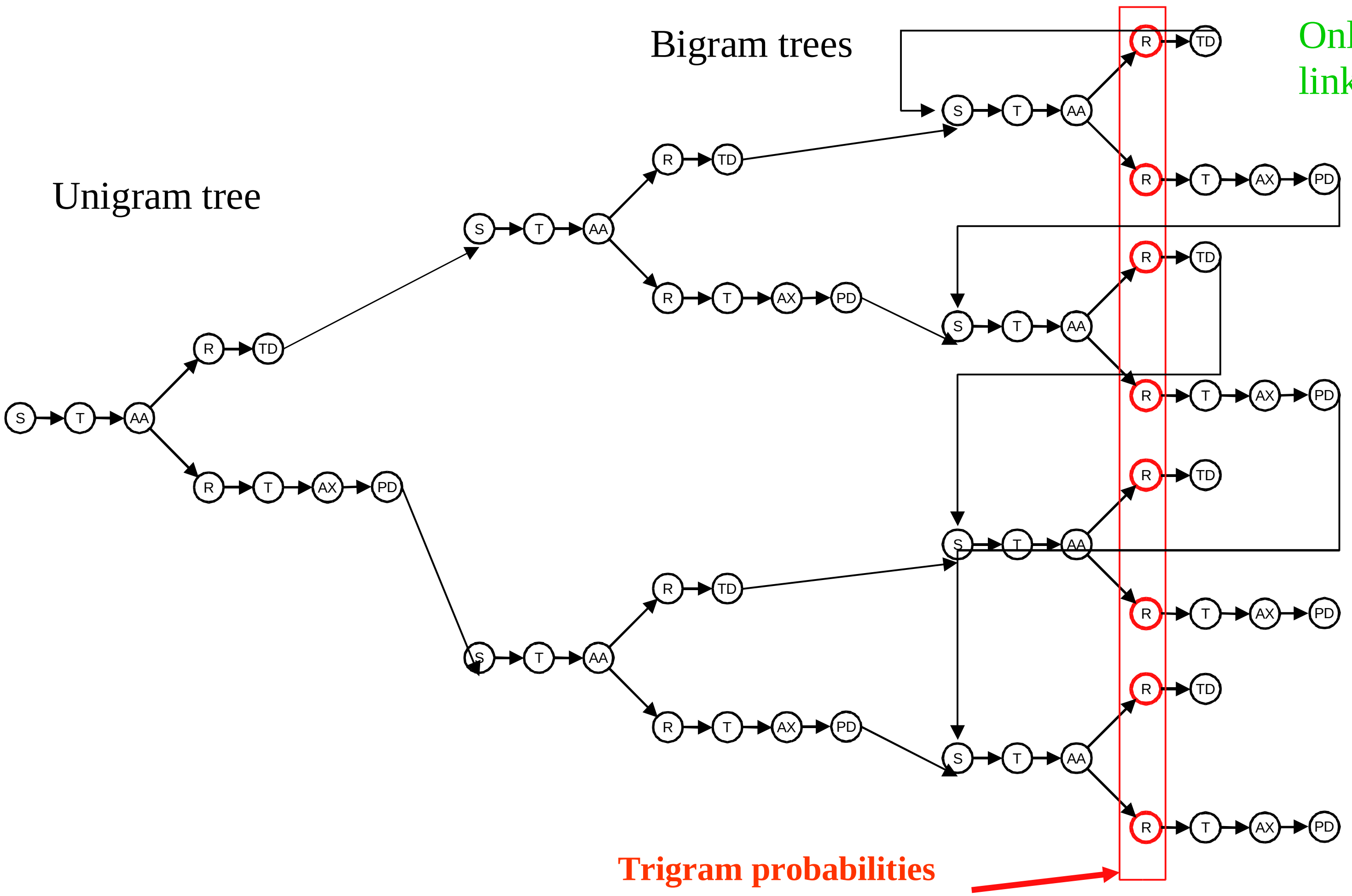
Unigram tree

P(word|START)

P(word|STARTED)

Bigram probabilities known here

Trigram Lextree Decoding



Trigram trees

Bigram trees

Only some links shown

Unigram tree

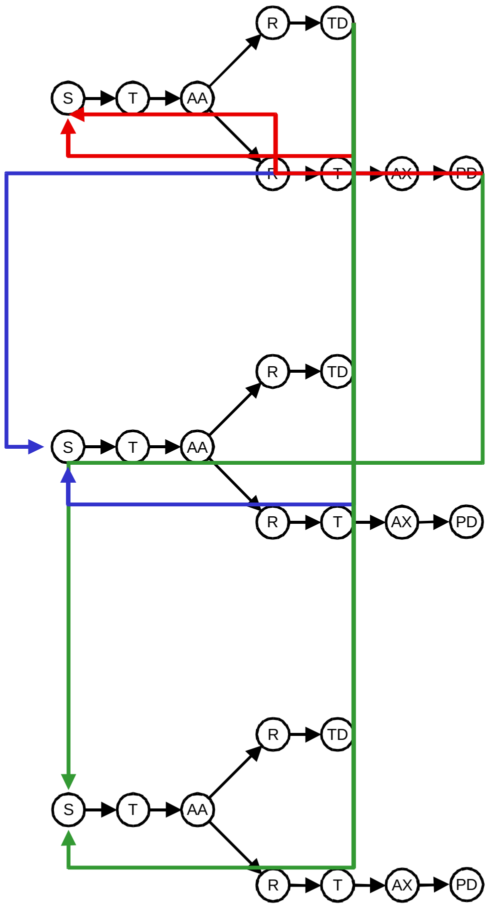
Trigram probabilities known here

Static 3-Lextree Decoding

All three lextrees similarly connected.

The color of a link indicates time constraints – different lextrees can be entered at different times

Trigram probability for any word uses the best bigram history for entire lextree (history obtained from backpointer table)



Sphinx 3 decoders: Fast lextree decoder

- Other speedup: Gaussian computation
  - Sub-vector quantized models used for Gaussian selection
  - Only selected Gaussians explicitly computed
- Highly accurate
  - Less than ~3% relative degradation of accuracy due to Gaussian selection
- Inflexible
  - Adaptation requires recomputation of sub-vector codebooks

The need for Sphinx 4

- Need to overcome Sphinx-3’s limitations
- Need for flexibility in acoustic modeling
- Require handling of multimodal inputs
  - With information fusion at various levels
- Need for more “correct” decoders
- Need for expansion of language model capabilities
- Facilitate the incorporation of several new online algorithms, that are currently difficult to incorporate into Sphinx-3
- Need for better application interfaces
- … .

The SPHINX of the new millennium

- Thanks to Re, the SUN god..
- An open source project by Carnegie Mellon University, SUN Microsystems Inc. and MERL
- Written entirely in Java™
  - the language of Re
- Highly modularized and flexible architecture
- Supports any acoustic model structure
- Supports most types of language models
  - CFGs, Ngrams, Combinations
- New algorithms for obtaining word level hypotheses
- Multimodal inputs
- Flexible APIs

Ngram to FST conversion: 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:**

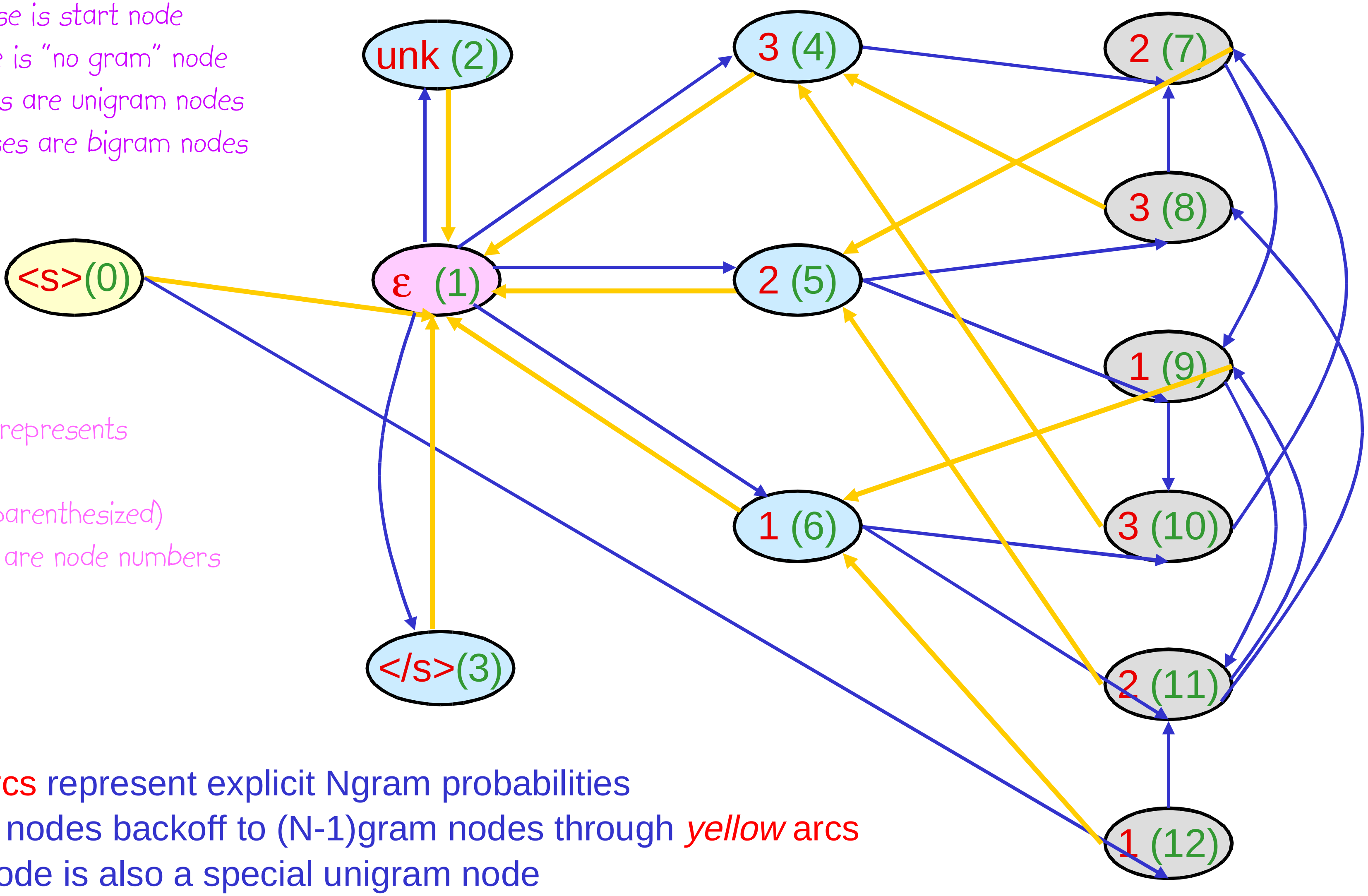
```
-0.1761 <s> one      0.0000
-0.4771 one three    0.1761
-0.3010 one two      0.3010
-0.1761 three two    0.0000
-0.3010 two one      0.3010
-0.4771 two three    0.1761
```

- **\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
```

Ngram to FST conversion: FST

- Yellow ellipse is start node
- Pink ellipse is "no gram" node
- Blue ellipses are unigram nodes
- Gray ellipses are bigram nodes

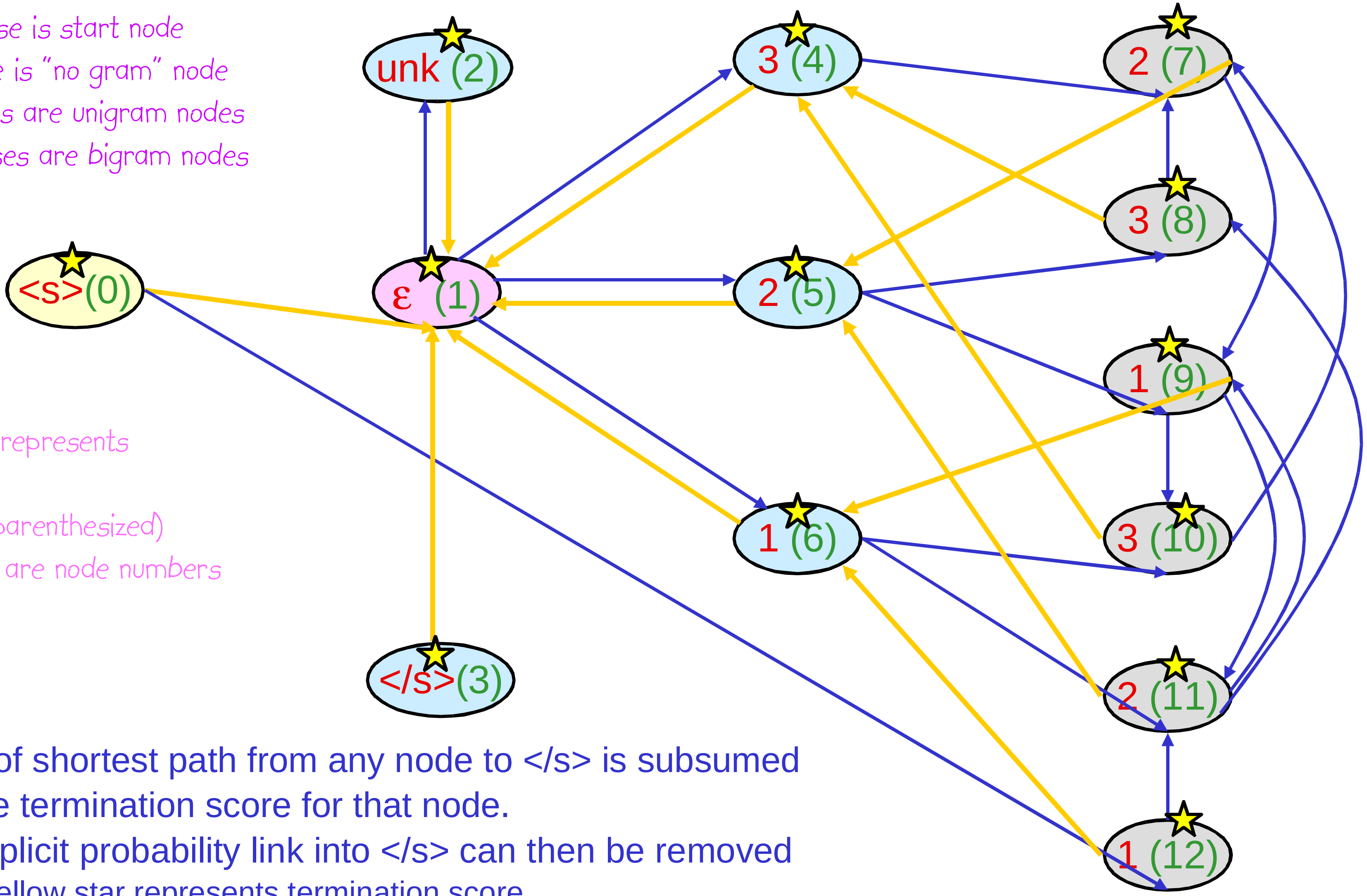


- red text represents words
- Green (parenthesized) numbers are node numbers

- *Blue* arcs represent explicit Ngram probabilities
- Ngram nodes backoff to (N-1)gram nodes through *yellow* arcs
- Start node is also a special unigram node

Ngram to FST conversion: FST

- Yellow ellipse is start node
- Pink ellipse is "no gram" node
- Blue ellipses are unigram nodes
- Gray ellipses are bigram nodes

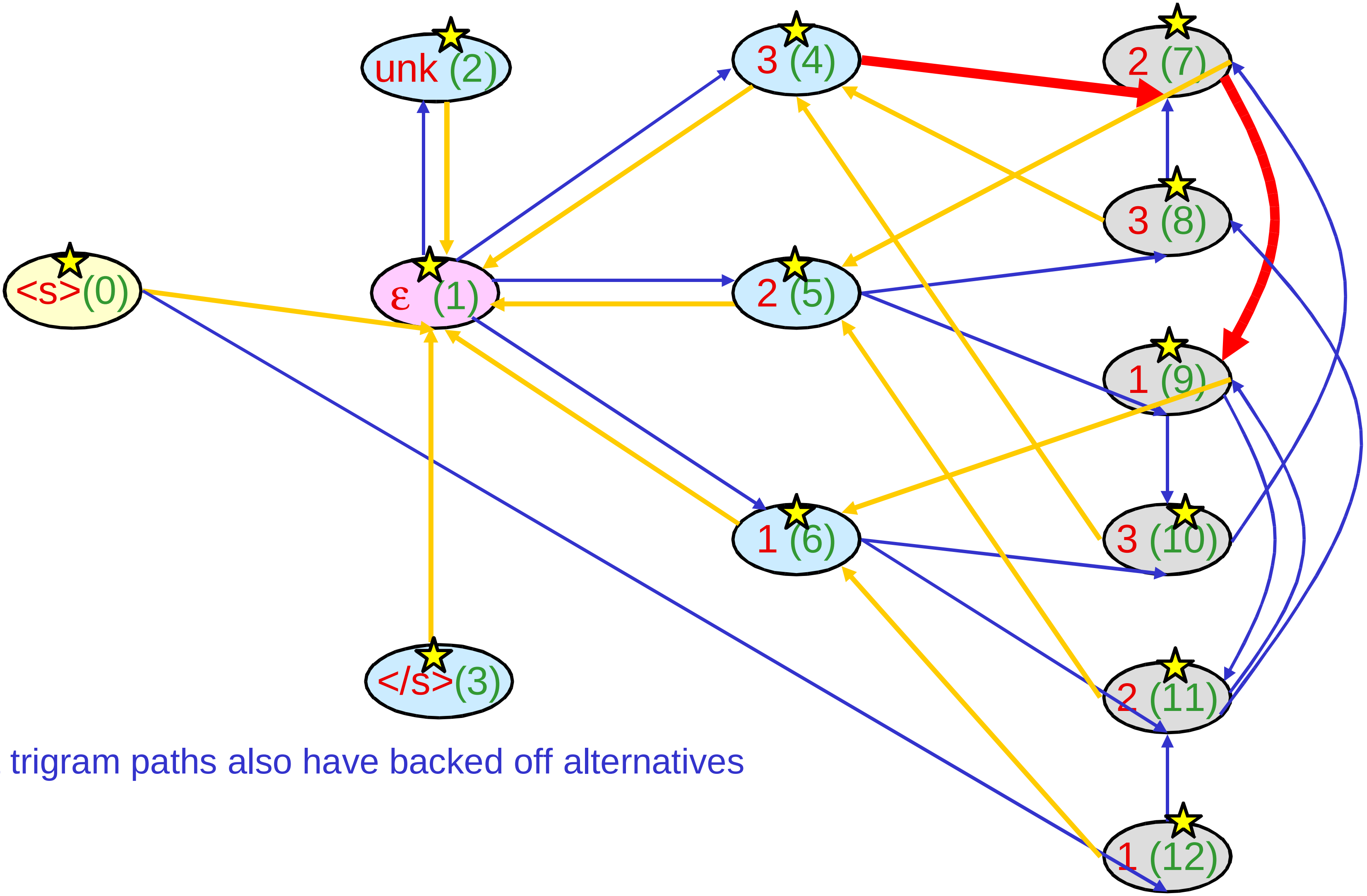


- red text represents words
- Green (parenthesized) numbers are node numbers

- Score of shortest path from any node to </s> is subsumed into the termination score for that node.
- The explicit probability link into </s> can then be removed
  - Yellow star represents termination score

Ngram to FST conversion: FST

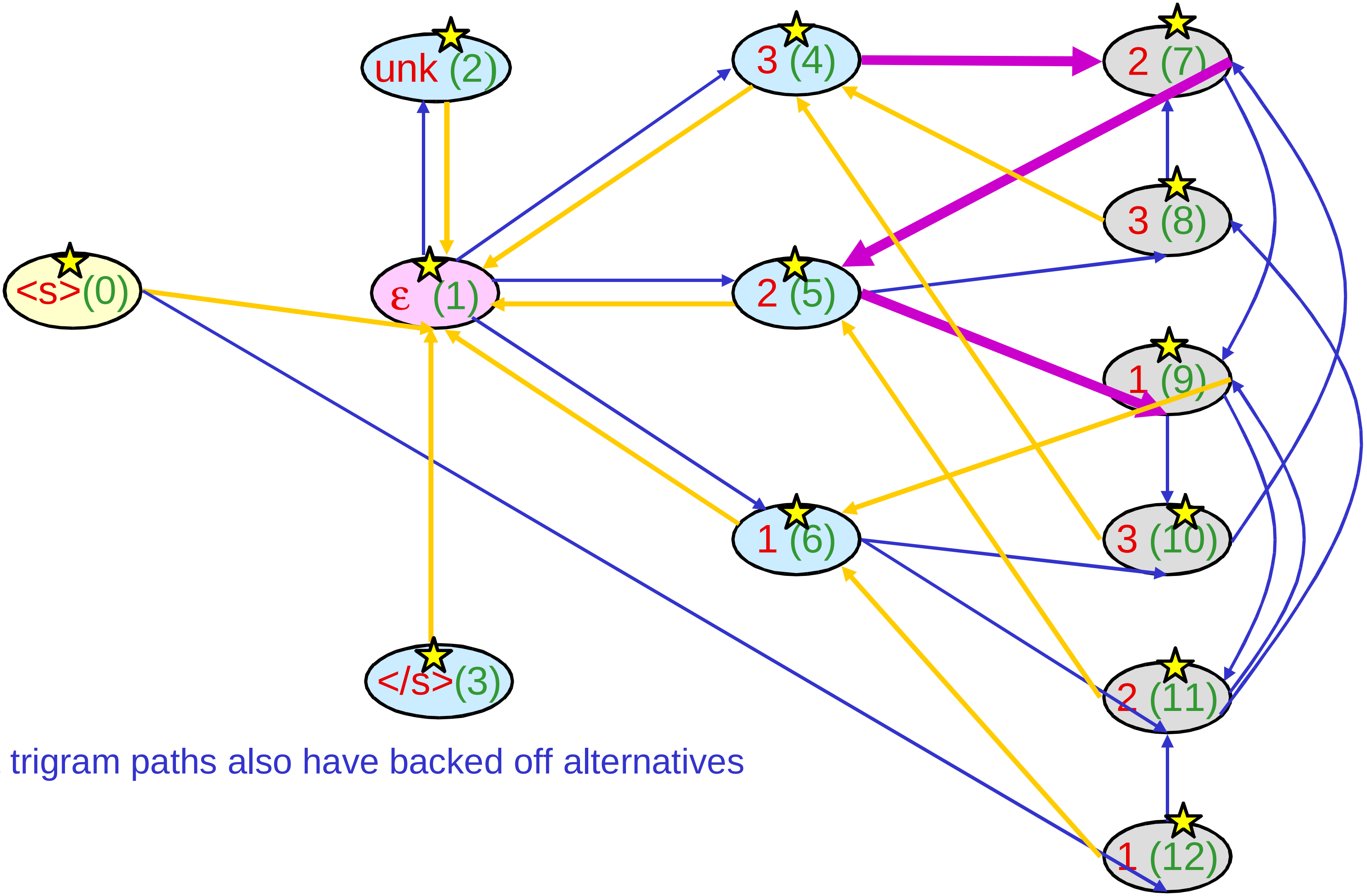
Explicit trigram path for trigram "three two one"



- Explicit trigram paths also have backed off alternatives

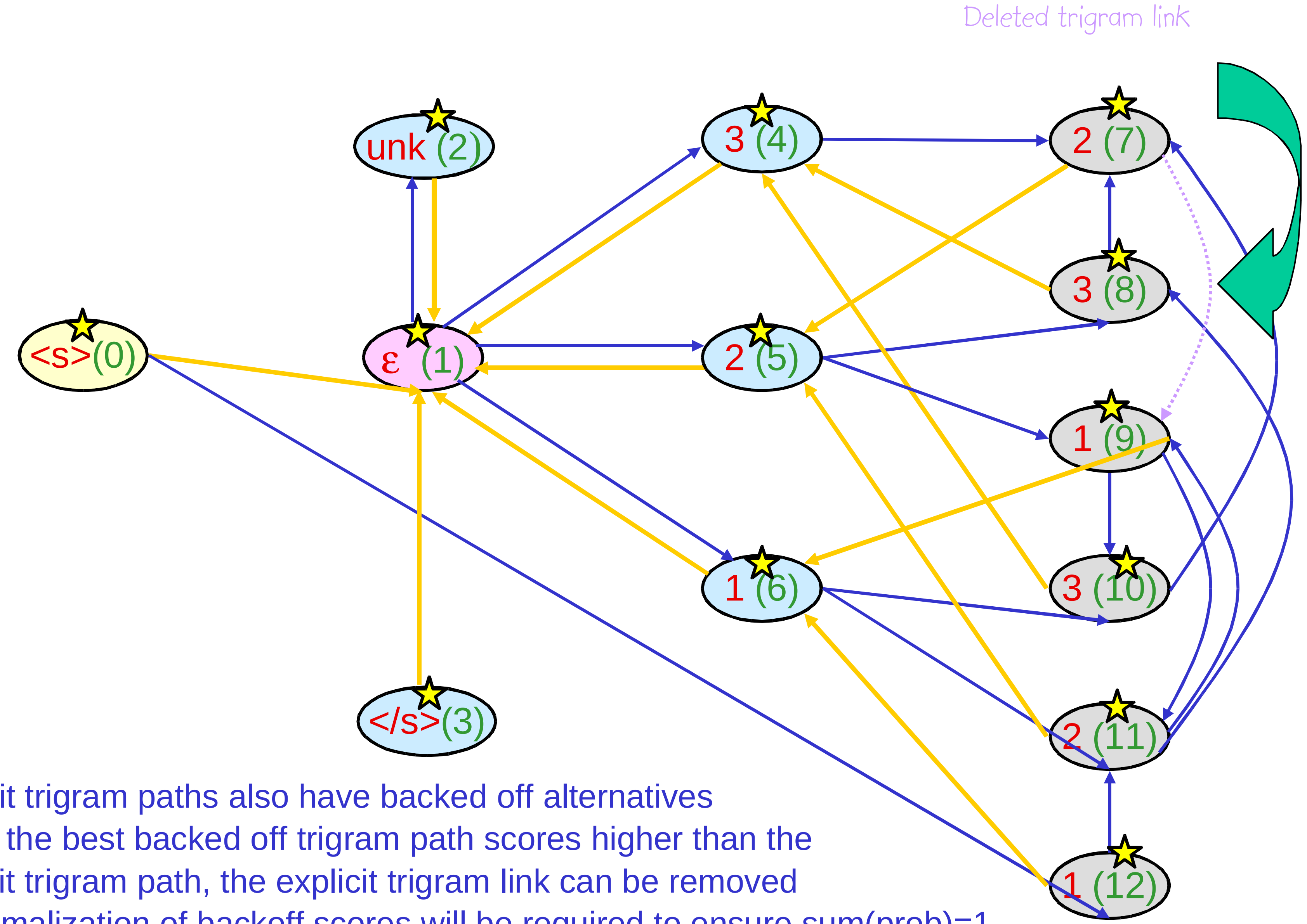
Ngram to FST conversion: FST

Backoff trigram path for trigram "three two one"



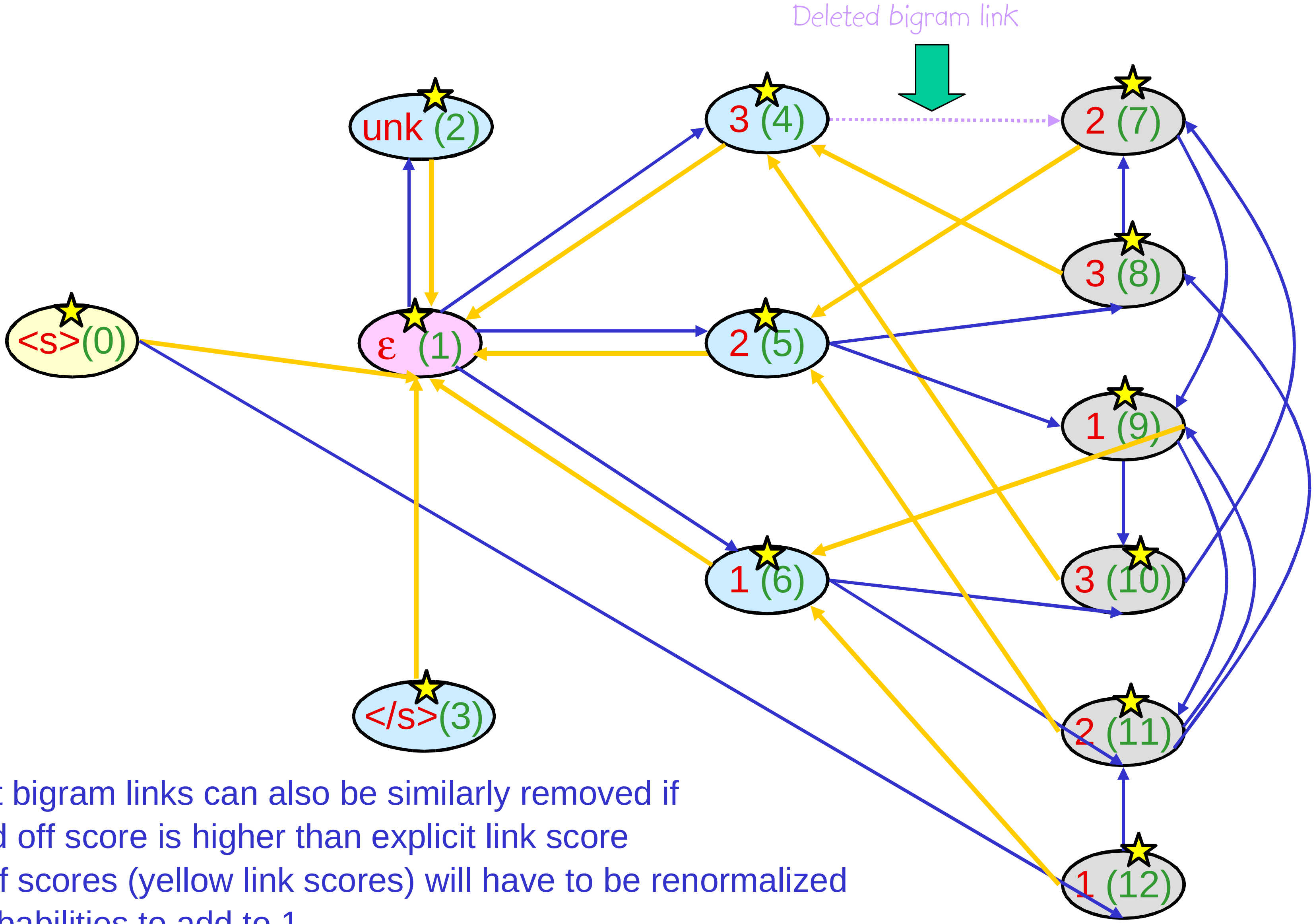
- Explicit trigram paths also have backed off alternatives

Ngram to FST conversion: FST



- Explicit trigram paths also have backed off alternatives
- When the best backed off trigram path scores higher than the explicit trigram path, the explicit trigram link can be removed
- Renormalization of backoff scores will be required to ensure sum(prob)=1

Ngram to FST conversion: FST



- Explicit bigram links can also be similarly removed if backed off score is higher than explicit link score
- Backoff scores (yellow link scores) will have to be renormalized for probabilities to add to 1.