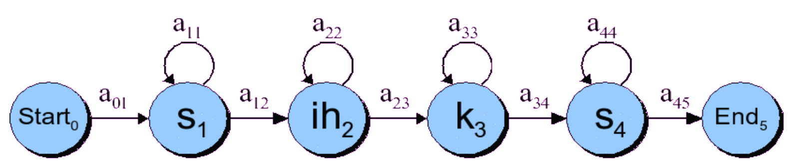
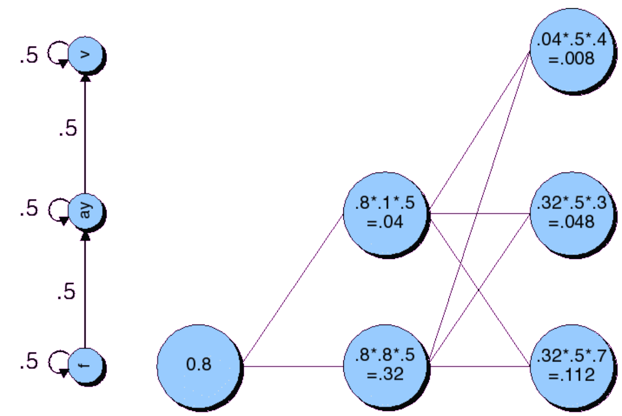
WFST: Weighted Finite State Transducer

CS136 Speech Recognition
January 24, 2020
Prof. Marie Meteer

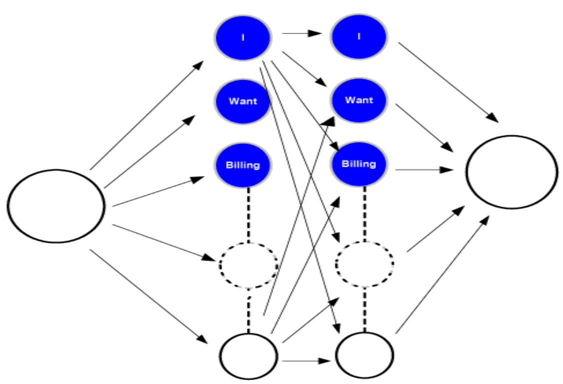
+ FSAs: A recurring structure in speech



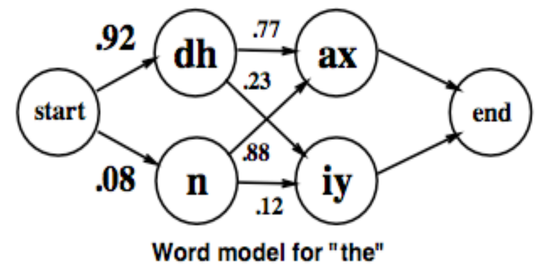
Phonetic HMM



Viterbi trellis

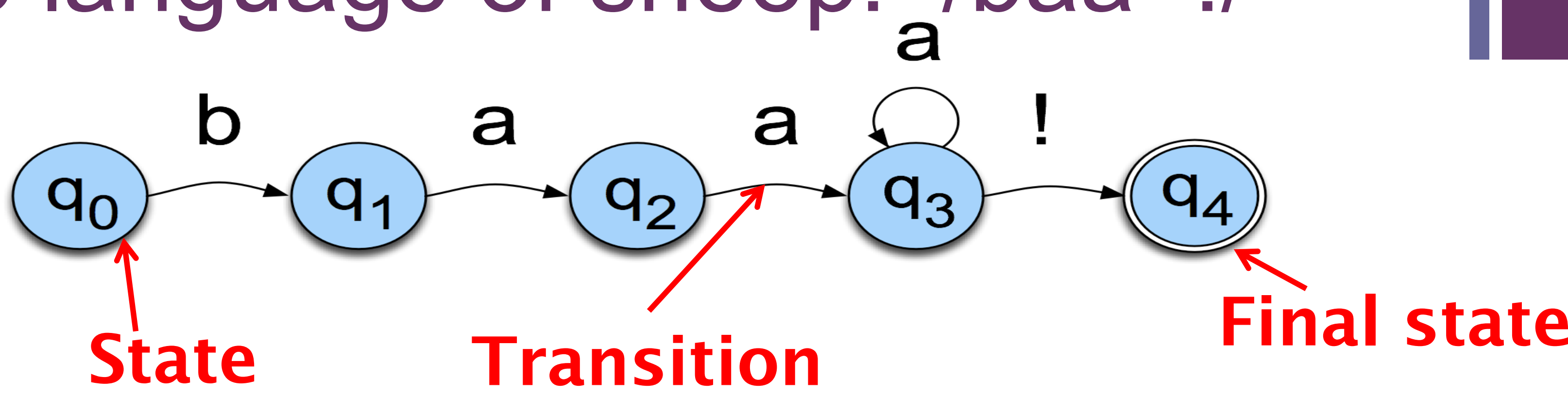


Language model



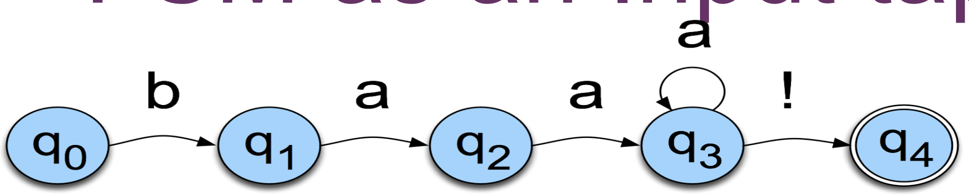
Pronunciation modeling

+ The language of sheep: /baa+!/



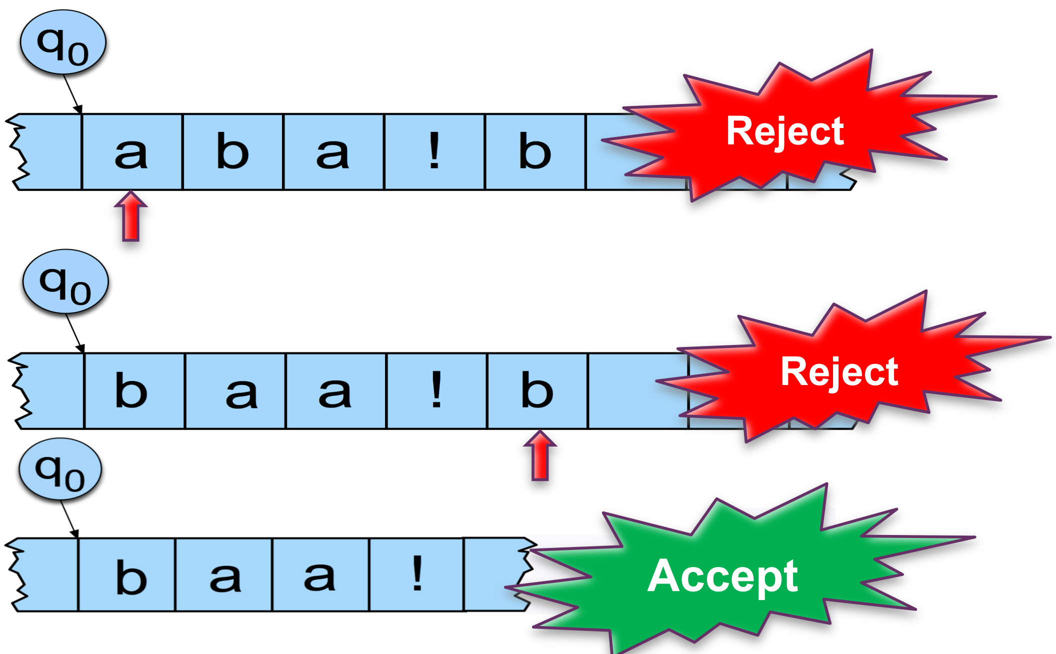
- We can say the following things about this machine
  - It has 5 states
  - b, a, and ! are in its alphabet
  - q_0 is the start state
  - q_4 is an accept state
  - It has 5 transitions

+ FSM as an Input tape acceptor



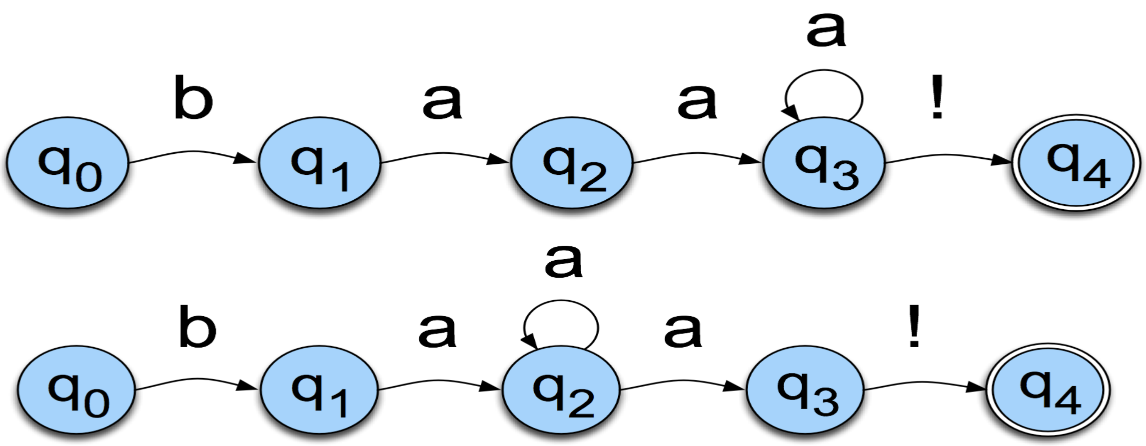
Transition Table

Given an input "tape", does my machine accept or reject that input?



	b	a	!	e
0	1			
1		2		
2		3		
3		3	4	
4				

+ Non-Determinism



Both define /baa+/

	b	a	!	e
0	1			
1		2		
2		2,3		
3			4	
4				

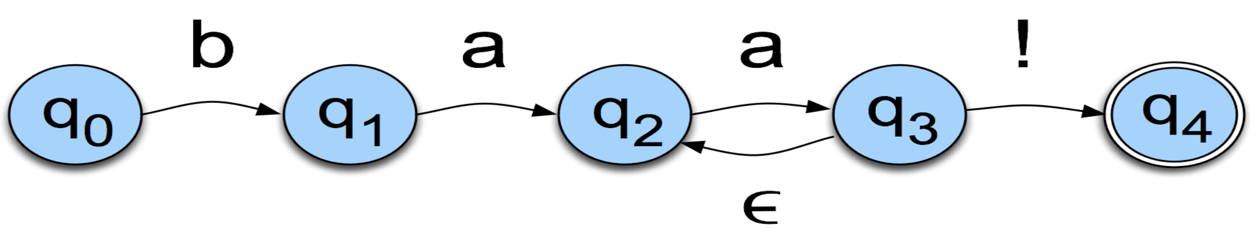
Input:

b	a	a

Stay in q2 or go to q3?

+ Non-Determinism cont.

- Yet another technique
  - Epsilon transitions
  - Key point: these transitions do not examine or advance the tape during recognition



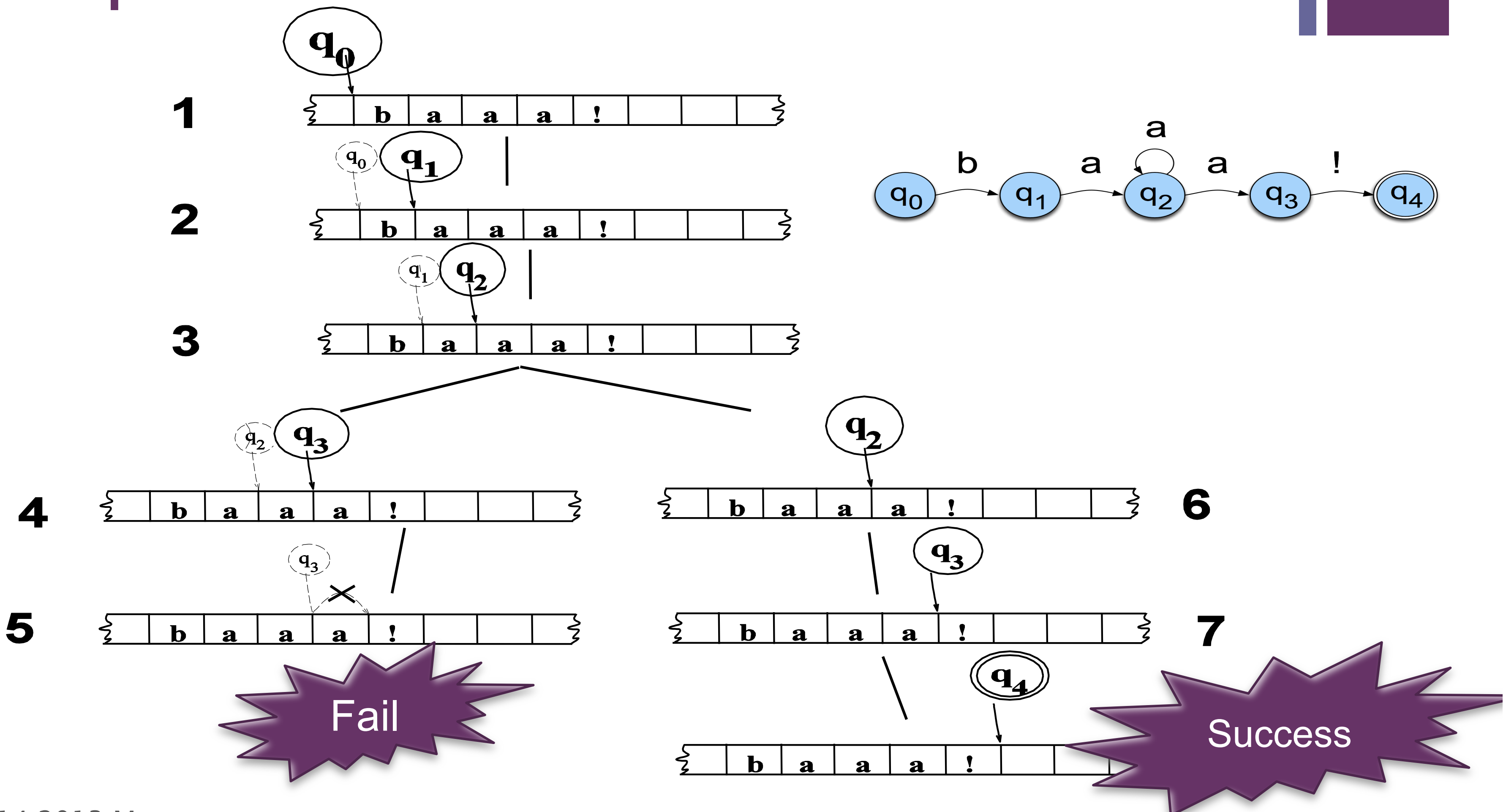
+ Equivalence

- Non-deterministic machines can be converted to deterministic ones with a fairly simple construction
  - That means that they have the same power:
  - non-deterministic machines are not more powerful than deterministic ones in terms of the languages they can accept
- Two basic approaches to ND recognition (used in all major implementations of regular expressions)
  - Either take a ND machine and convert it to a D machine and then do recognition with that.
  - Or explicitly manage the process of recognition as a state-space search (leaving the machine as is).

+ Non-Deterministic Recognition: Search

- In a ND FSA *there exists at least one path* through the machine for a string that is in the language defined by the machine.
- *But not all paths* directed through the machine for an accept string lead to an accept state.
- *No paths* through the machine lead to an accept state for a string not in the language.
- Non-determinism doesn't get us more formal power and it causes headaches so why bother?
  - More natural (understandable) solutions

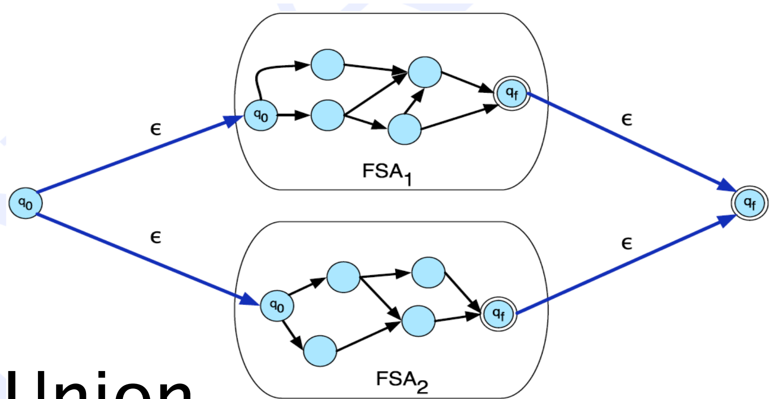
+ Example



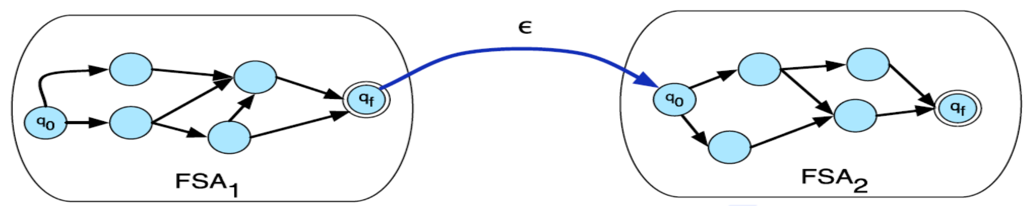
q0
1
b a a a !
q0 q1
2
b a a a !
q1 q2
3
b a a a !
q2 q3
4
b a a a !
q3
5
b a a a !
Fail
q2
6
b a a a !
q3
7
b a a a !
q4
b a a a !
Success
b a a a !
q0 q1 q2 q3 q4

+ Compositional Machines

- Formal languages are just sets of strings
- Therefore, we can talk about various set operations (intersection, union, concatenation)
- This turns out to be a useful exercise

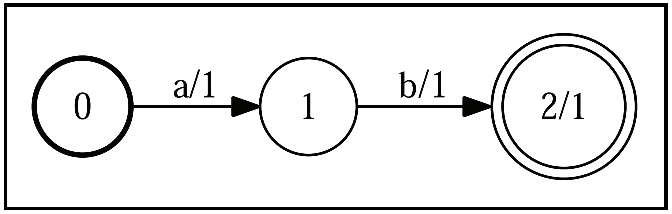


Union



Concatenation

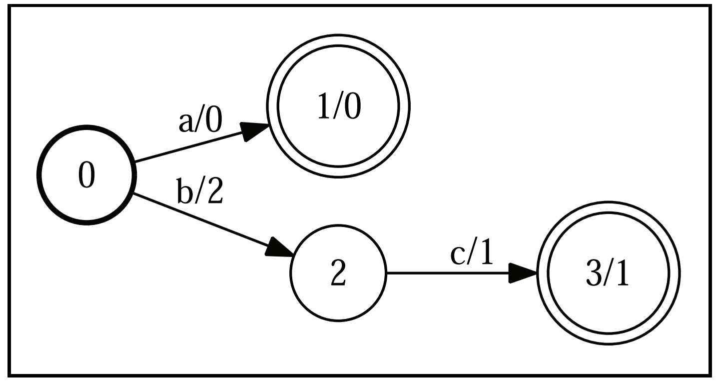
+ Weighted finite state acceptors



- Like a normal FSA but with costs on the arcs and final-states
  - Note: cost comes after “/”. For final-state, “2/1” means final-cost 1 on state 2.
- View WFSA as a function from a string to a cost.
- In this view, unweighted FSA is f : string → {0, ∞}.
- If multiple paths have the same string, take the one with the lowest cost.
- This example maps ab to (3 = 1+1+1), all else to ∞.

Thanks for Mirko Hannemann for this slide

+ Weights vs. costs

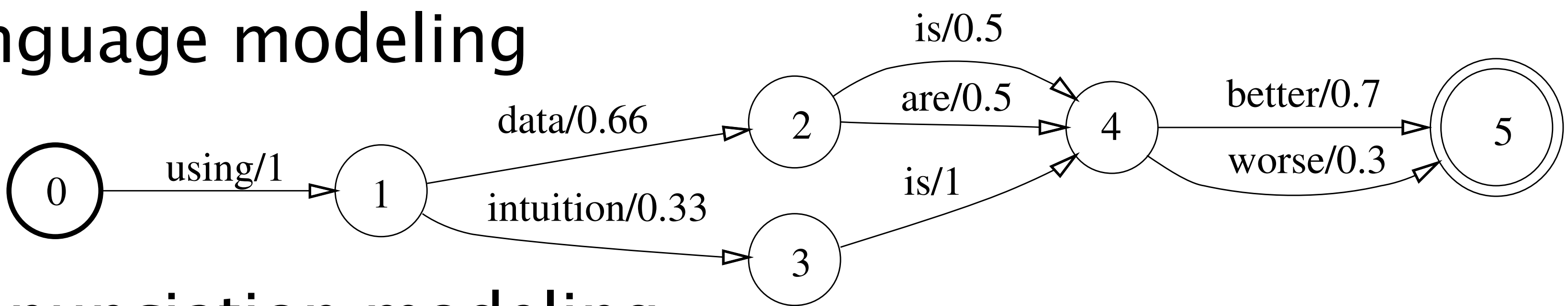


- Use “cost” to refer to the numeric value, and “weight” when speaking abstractly, e.g.:
  - The acceptor above accepts **a** with unit weight.
  - It accepts **a** with zero cost.
  - It accepts **bc** with cost 4=2+1+1
  - State 1 is final with unit weight.
  - The acceptor assigns zero weight to **xyz**.
  - It assigns infinite cost to **xyz** .

Thanks for Mirko Hannemann for this slide

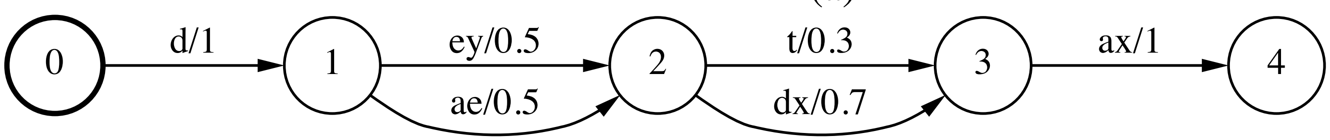
+ WSFAs in speech

Language modeling



Pronunciation modeling

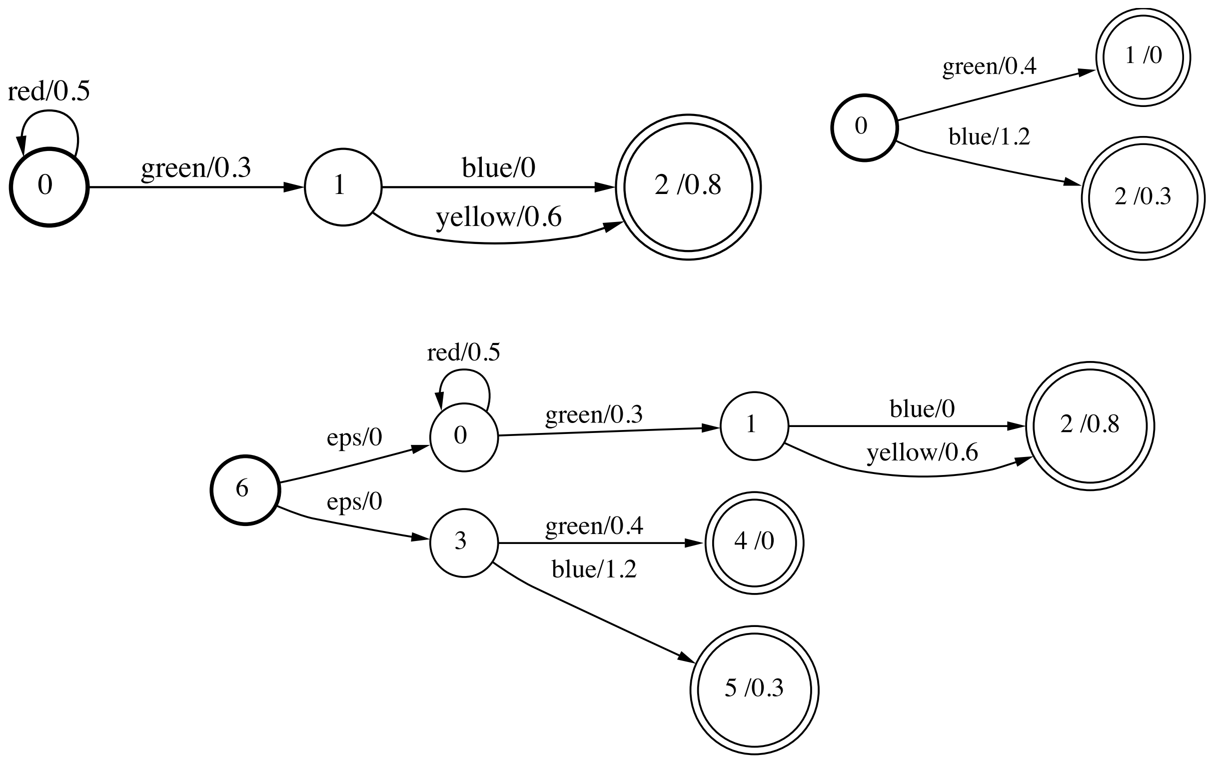
(a)



+ Operations: Union

Sum (Union) – Illustration

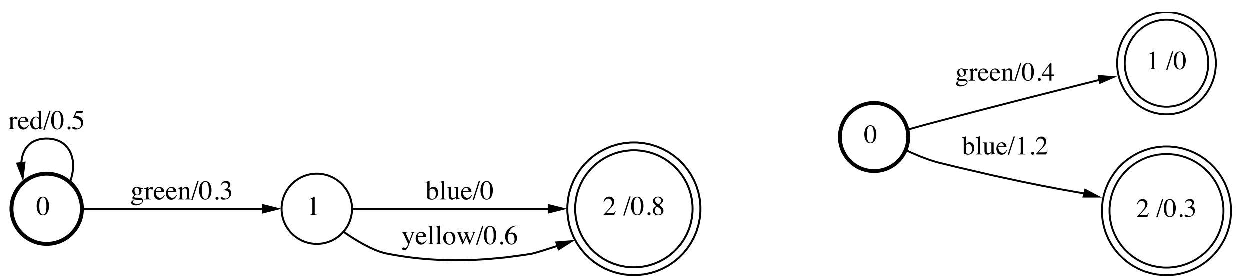
- Definition: $[\![T_1 \oplus T_2]\!](x, y) = [\![T_1]\!](x, y) \oplus [\![T_2]\!](x, y)$
- Example:

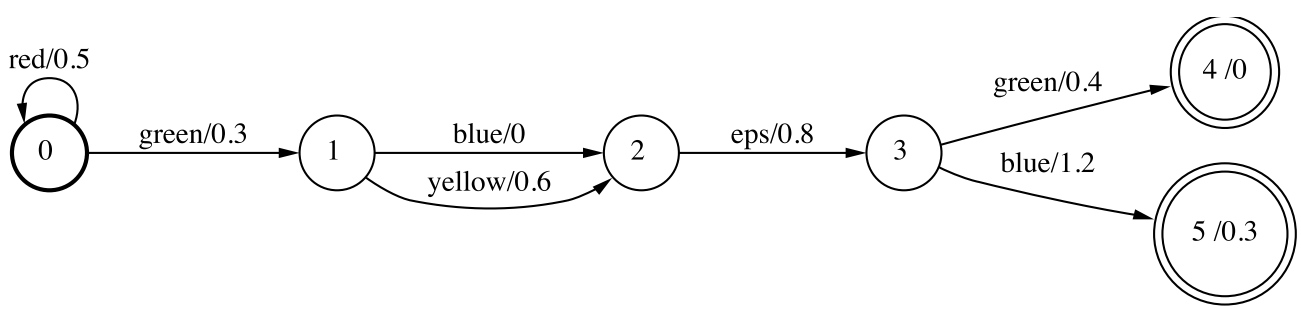


+ Unary Operation: Concatenation

Product (Concatenation) – Illustration

- Definition: $[\![T_1 \otimes T_2]\!](x, y) = \bigoplus_{x=x_1 x_2, y=y_1 y_2} [\![T_1]\!](x_1, y_1) \otimes [\![T_2]\!](x_2, y_2)$
- Example:

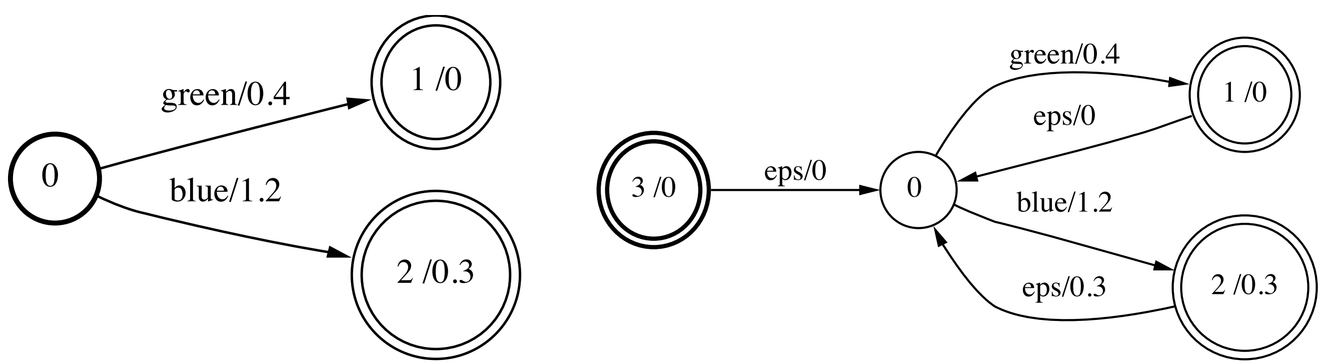




+ Unary Operation: Closure

Closure – Illustration

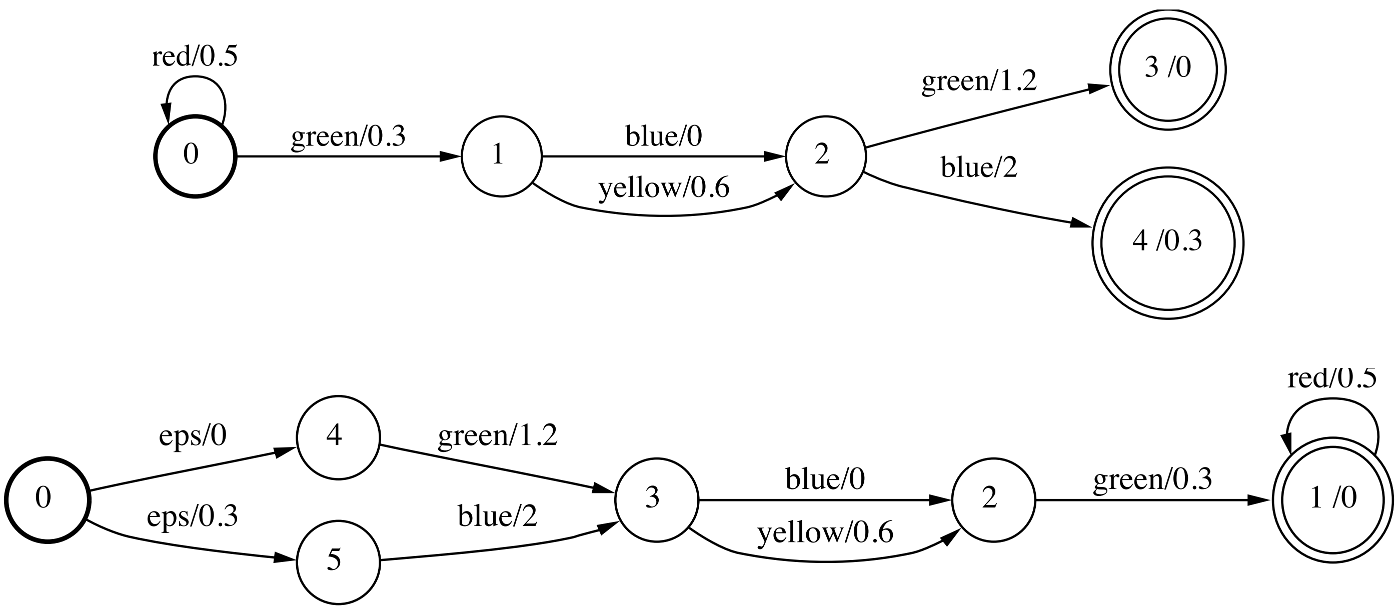
- Definition: $[\![T^*]\!](x,y) = \bigoplus_{n=0}^{\infty} [\![T^n]\!](x,y)$
- Example:



+ Unary Operation: Reversal

Reversal – Illustration

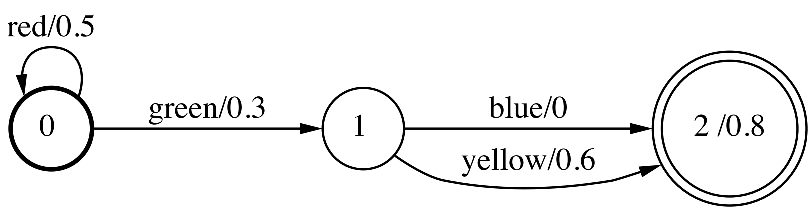
- Definition: $[\![\widetilde{T}]\!](x, y) = [\![T]\!](\widetilde{x}, \widetilde{y})$
- Example:

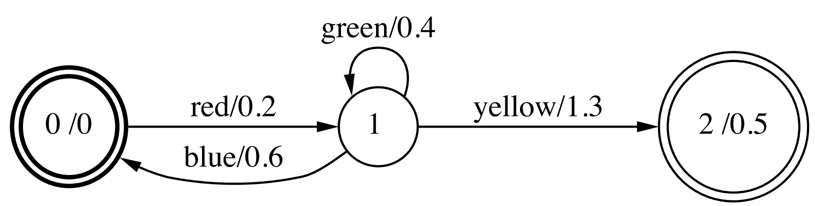


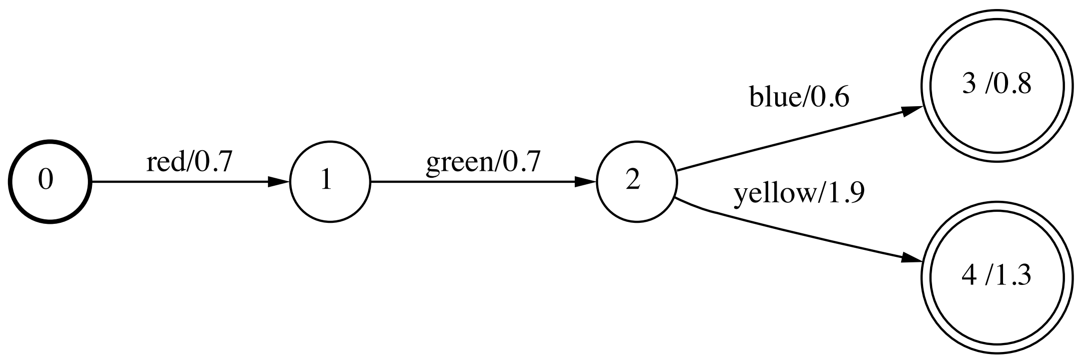
+ Binary operation: Intersection

Intersection – Illustration

- Definition: $[\![A_1 \cap A_2]\!](x) = [\![A_1]\!](x) \otimes [\![A_2]\!](x)$
- Example:



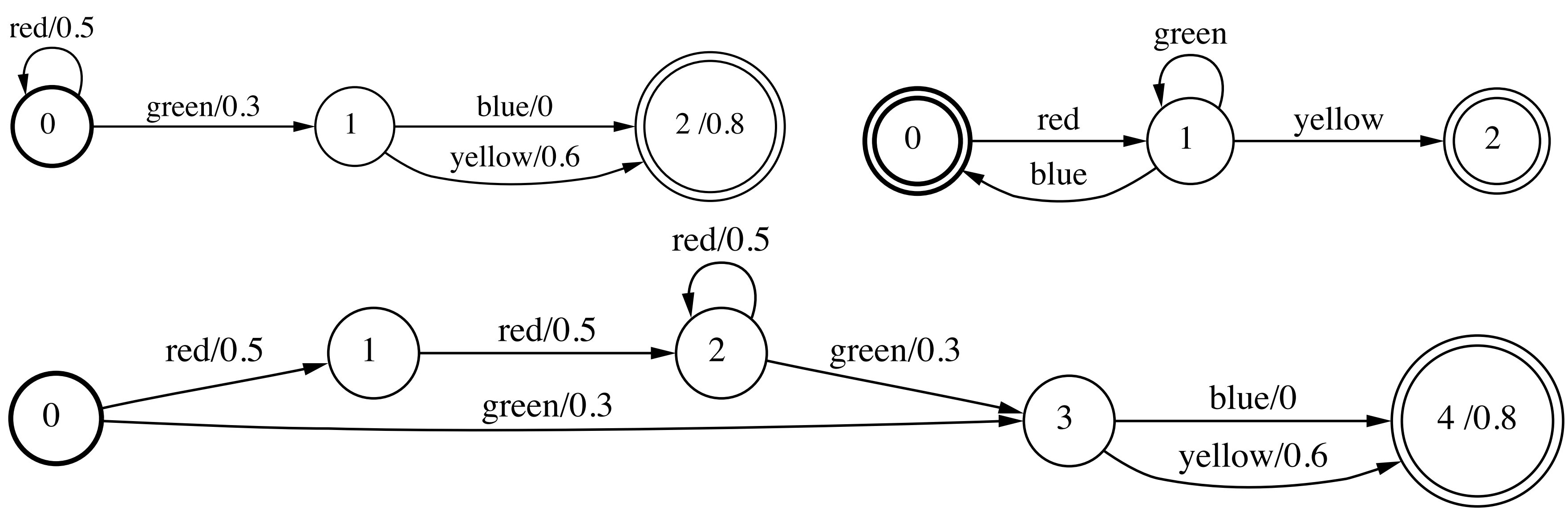




+ Binary operation: difference

Difference – Illustration

- Definition: $[\![A_1 - A_2]\!](x) = [\![A_1 \cap \overline{A_2}]\!](x)$
- Example:

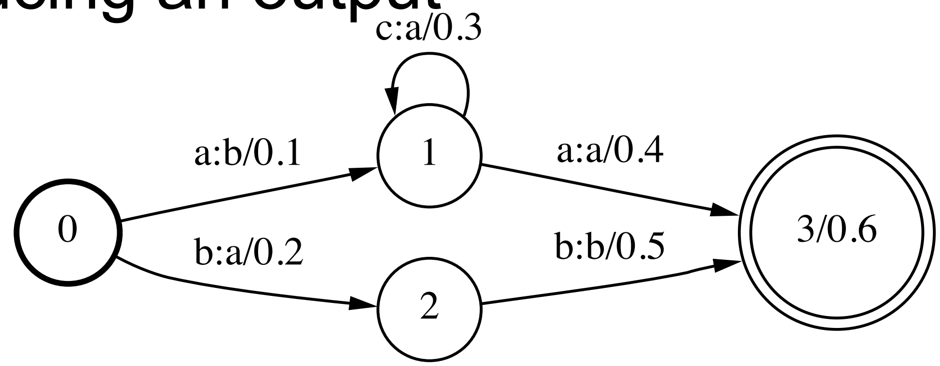


+ WFS Transducer

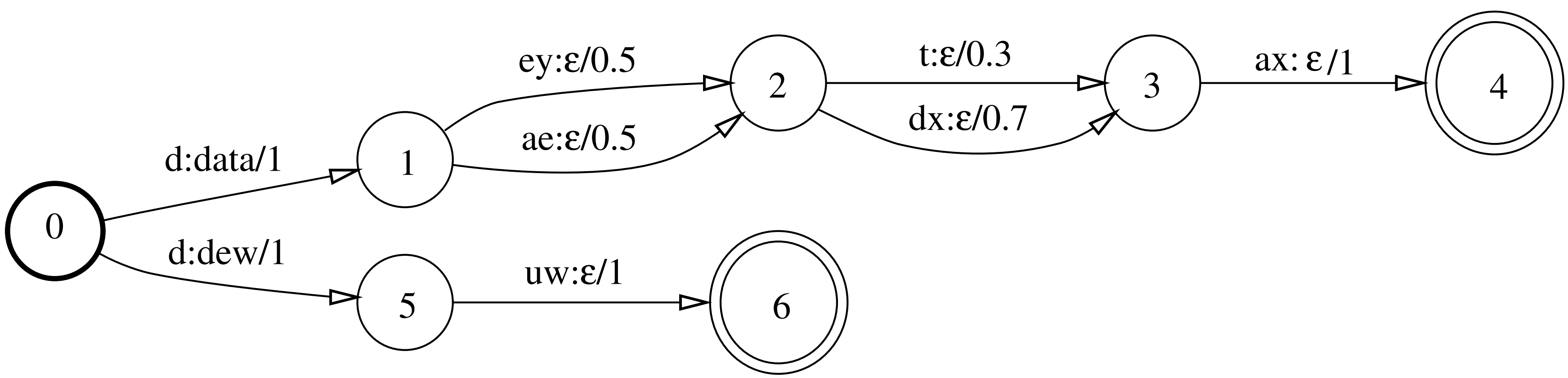
- Accept an input while producing an output

input output weight

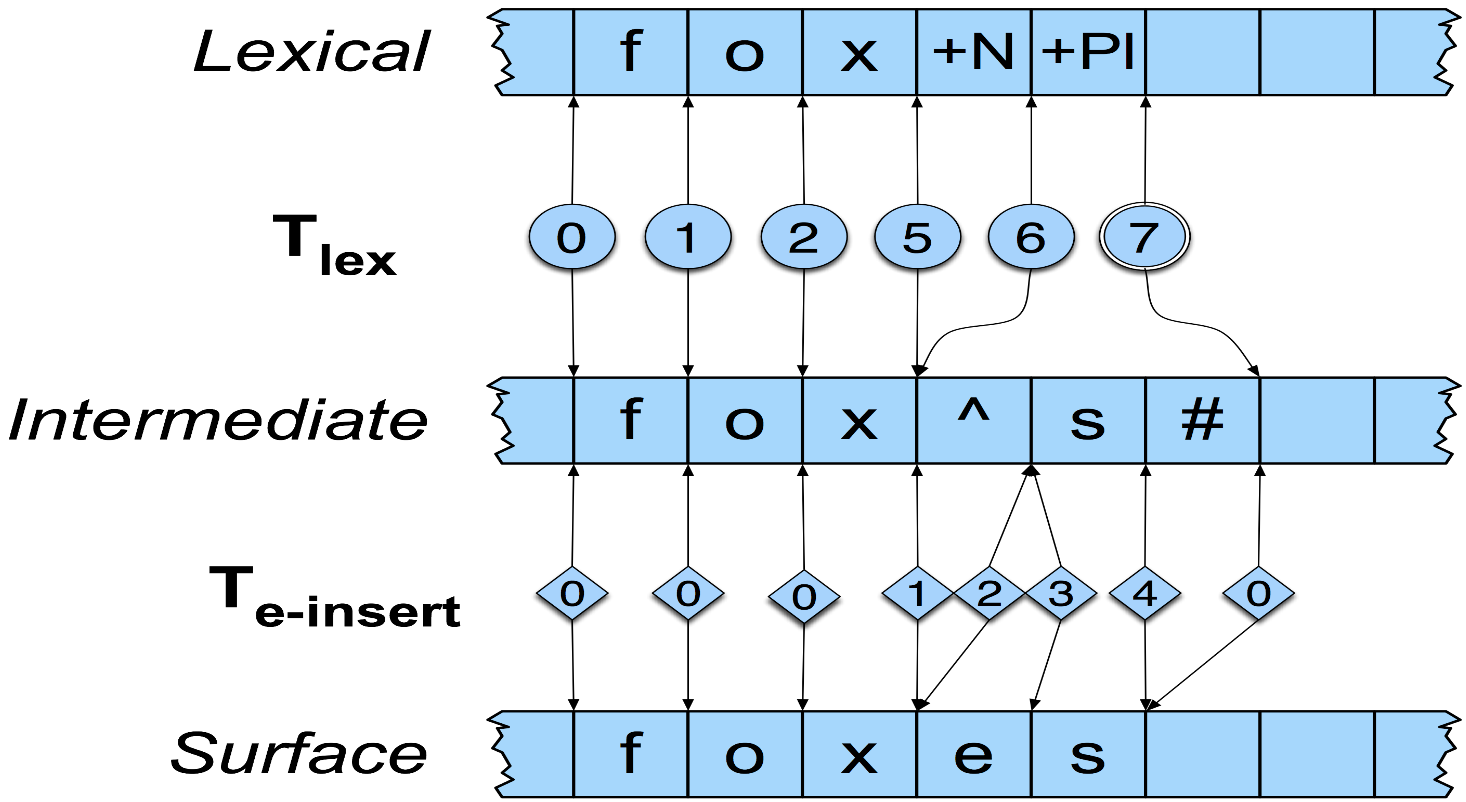
a:b/0.3



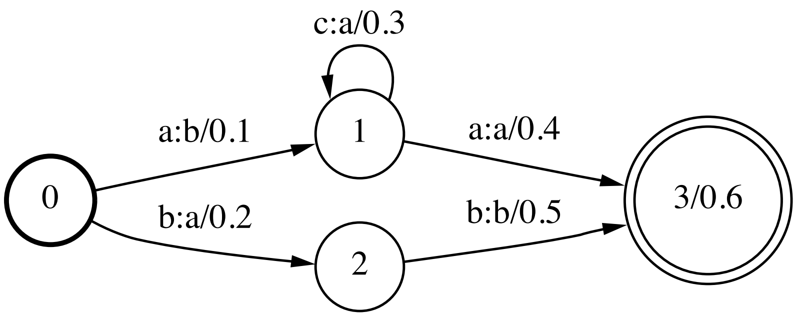
- Input phonemes: output words

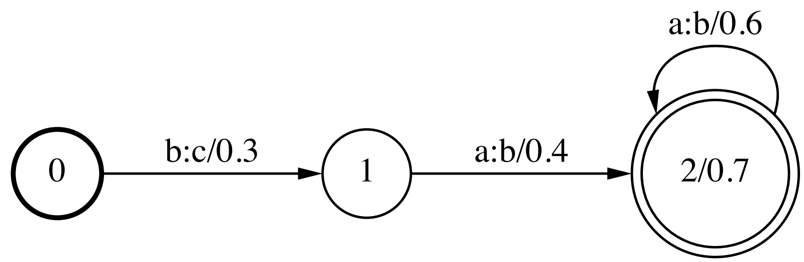


+ FST for morphology: Foxes and Cats

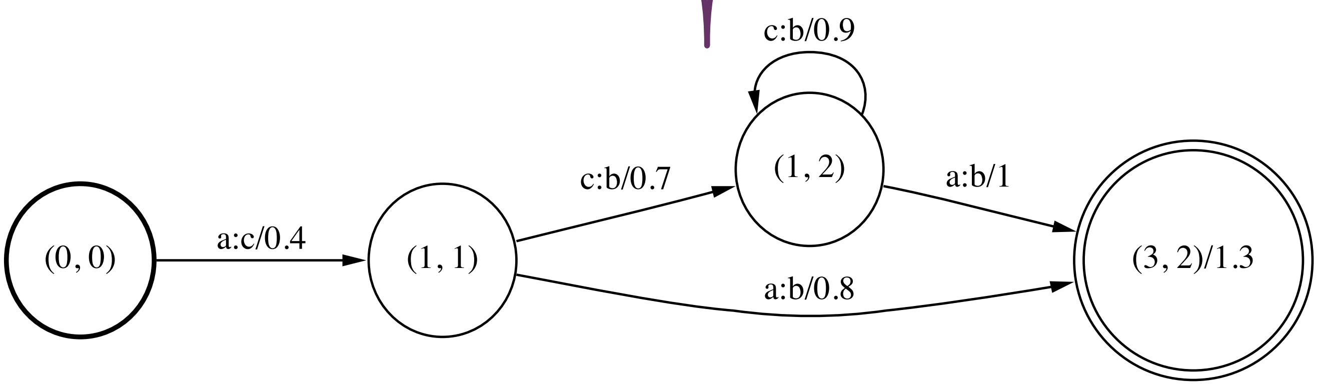


+ Composition



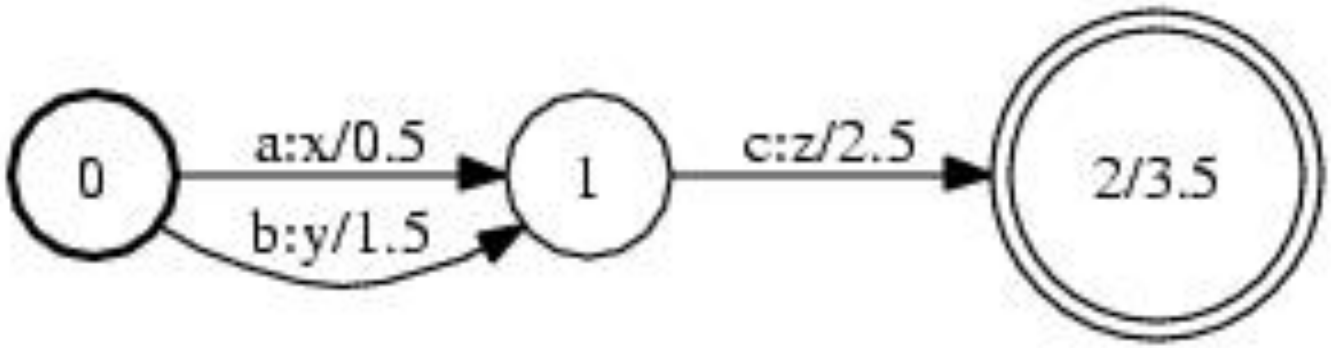


a c a → b a a → c b b



a c a → c b b

+ WFSTs in Action with OpenFST



- Create text file
- Compile
- Print
- Show info
- Union
- Concatenate
- Compose
- Invert

+ Math behind WFSTs: Semirings

Weight Sets: Semirings

A *semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ = a ring that may lack negation.

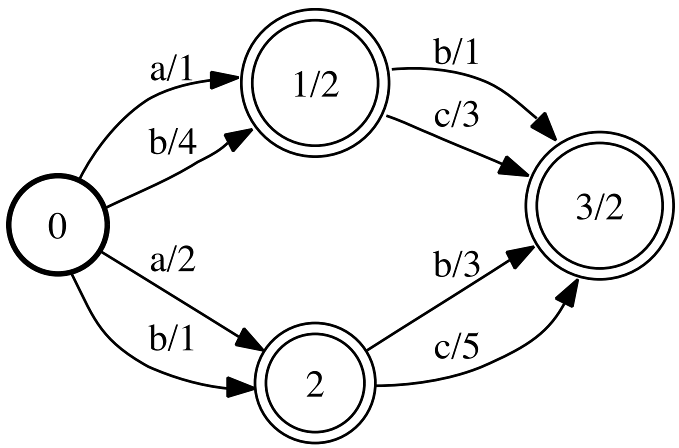
- Sum: to compute the weight of a sequence (sum of the weights of the paths labeled with that sequence).
- Product: to compute the weight of a path (product of the weights of constituent transitions).

SEMIRING	SET	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
Boolean	$\{0, 1\}$	$\vee$	$\wedge$	0	1
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	min	$+$	$+\infty$	0
String	$\Sigma^* \cup \{\infty\}$	$\wedge$	$\cdot$	∞	ϵ

$\oplus_{\log}$ is defined by: $x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$ and $\wedge$ is longest common prefix. The string semiring is a *left semiring*.

+ Same FSA, multiple purposes

Weighted Automaton/Acceptor



Forward Algorithm

Viterbi Algorithm

Probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$	Tropical semiring $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$
$[\![A]\!](ab) = 14$	$[\![A]\!](ab) = 4$
$(1 \times 1 \times 2 + 2 \times 3 \times 2 = 14)$	$(\min(1 + 1 + 2, 3 + 2 + 2) = 4)$

+ Optimization Algorithms

- Definitions

Operation	Description
Connection	Removes non-accessible/non-coaccessible states
ϵ-Removal	Removes ϵ-transitions
Determinization	Creates equivalent deterministic machine
Pushing	Creates equivalent pushed/stochastic machine
Minimization	Creates equivalent minimal deterministic machine

- Conditions: There are specific semiring conditions for the use of these algorithms. Not all weighted automata or transducers can be determinized using that algorithm.

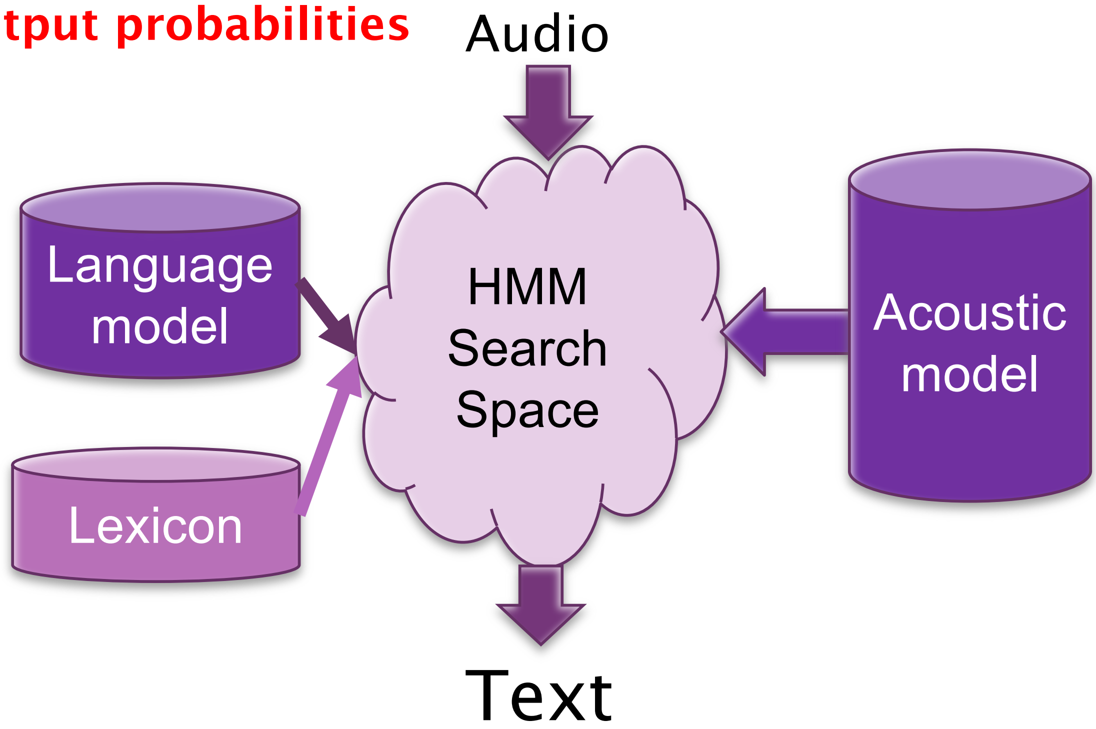
+ The Speech Problem

Search through space of all possible sentences

Defined by the HMM

Pick the one that is most probable given the waveform.

Based on the transition and output probabilities in the HMM



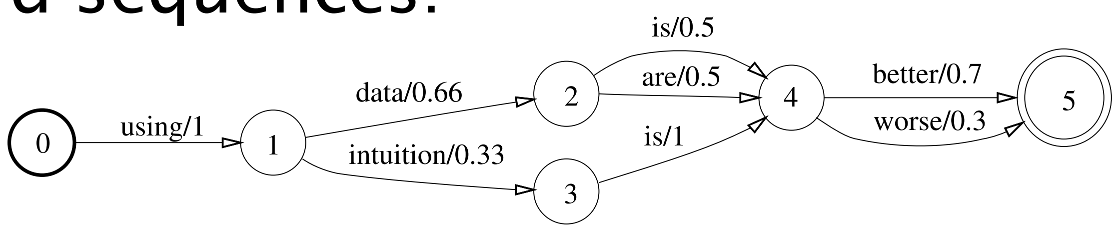
+ Weighted Finite State Transducers

- Used by Kaldi
- Weighted finite state automaton that transduces an input sequence to an output sequence (Mohri 2008)
  - States connected by transitions.
  - Each transition has an input label and output label weight

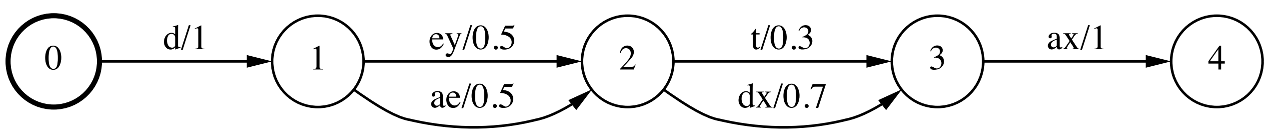
Thanks to Steve Renals for these slides.

+ Weighted Finite State Acceptors

Word sequences:



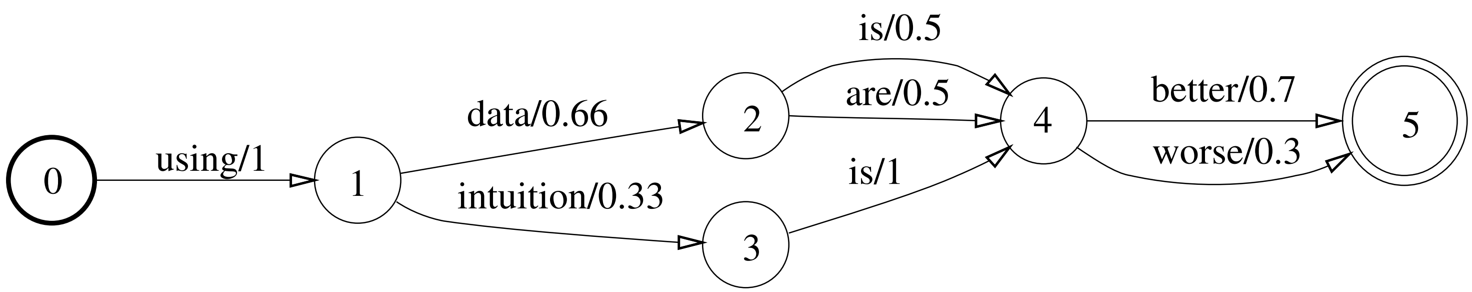
Phoneme sequences:



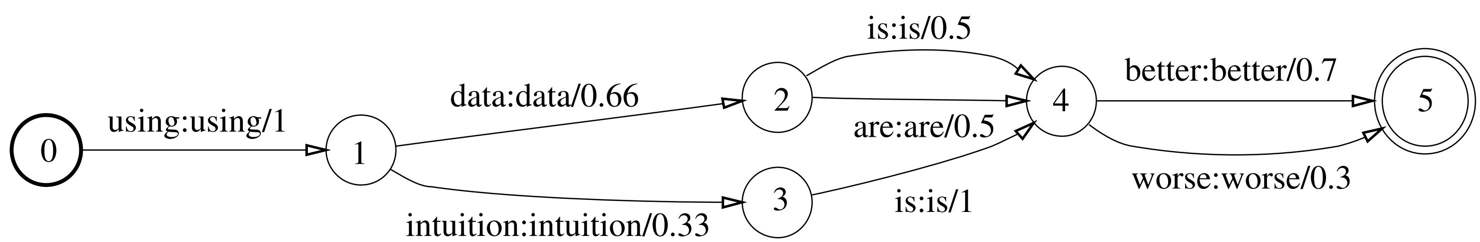
Thanks to Steve Renals for these slides.

+ Weighted Finite State Transducers

Acceptor



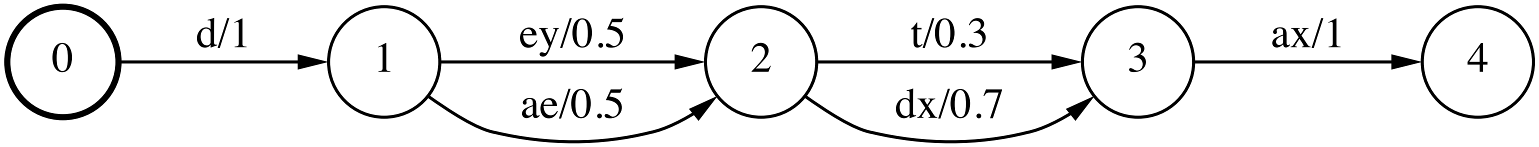
Transducer



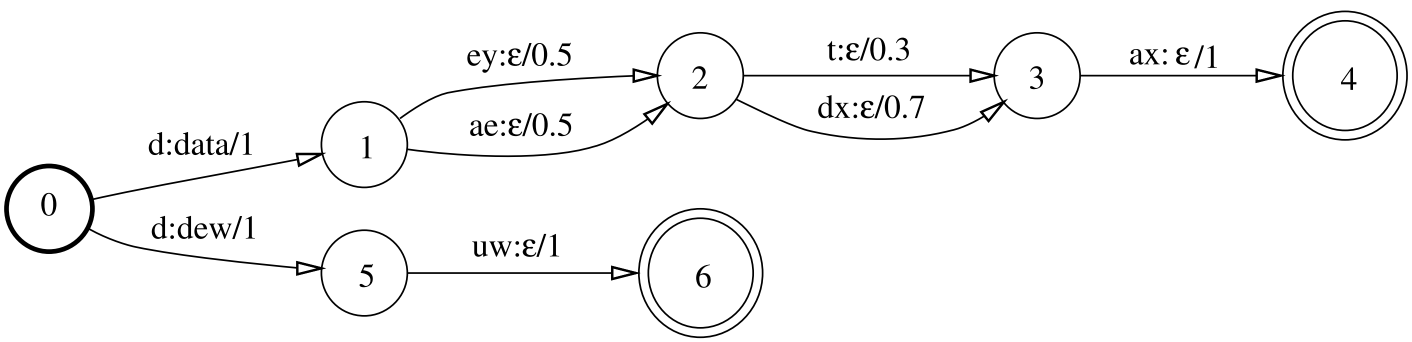
Thanks to Steve Renals for these slides.

+ Weighted Finite State Transducers

Acceptor



Transducer



Thanks to Steve Renals for these slides.

+ WFTS Algorithms

- Composition
  - Combine transducers at different levels.
  - For example if G is a finite state grammar and L is a pronunciation dictionary then $L \circ G$ transduces a phone string to word strings allowed by the grammar
- Determinization
  - Ensure that each state has no more than a single output transition for a given input label
- Minimization
  - transforms a transducer to an equivalent transducer with the fewest possible states and transitions

Thanks to Steve Renals for these slides.

Applying WFSTs to speech recognition

- Represent the following components as WFSTs

	transducer	input sequence	output sequence
G	word-level grammar	words	words
L	pronunciation lexicon	phones	words
C	context-dependency	CD phones	phones
H	HMM	HMM states	CD phones

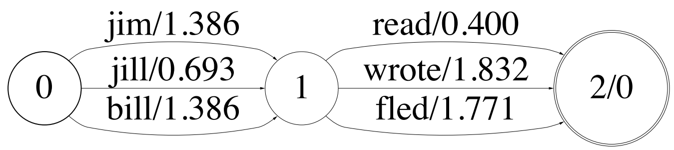
- Composing L and G results in a transducer $L \circ G$ that maps a phone sequence to a word sequence

- $H \circ C \circ L \circ G$ results in a transducer that maps from HMM states to a word sequence

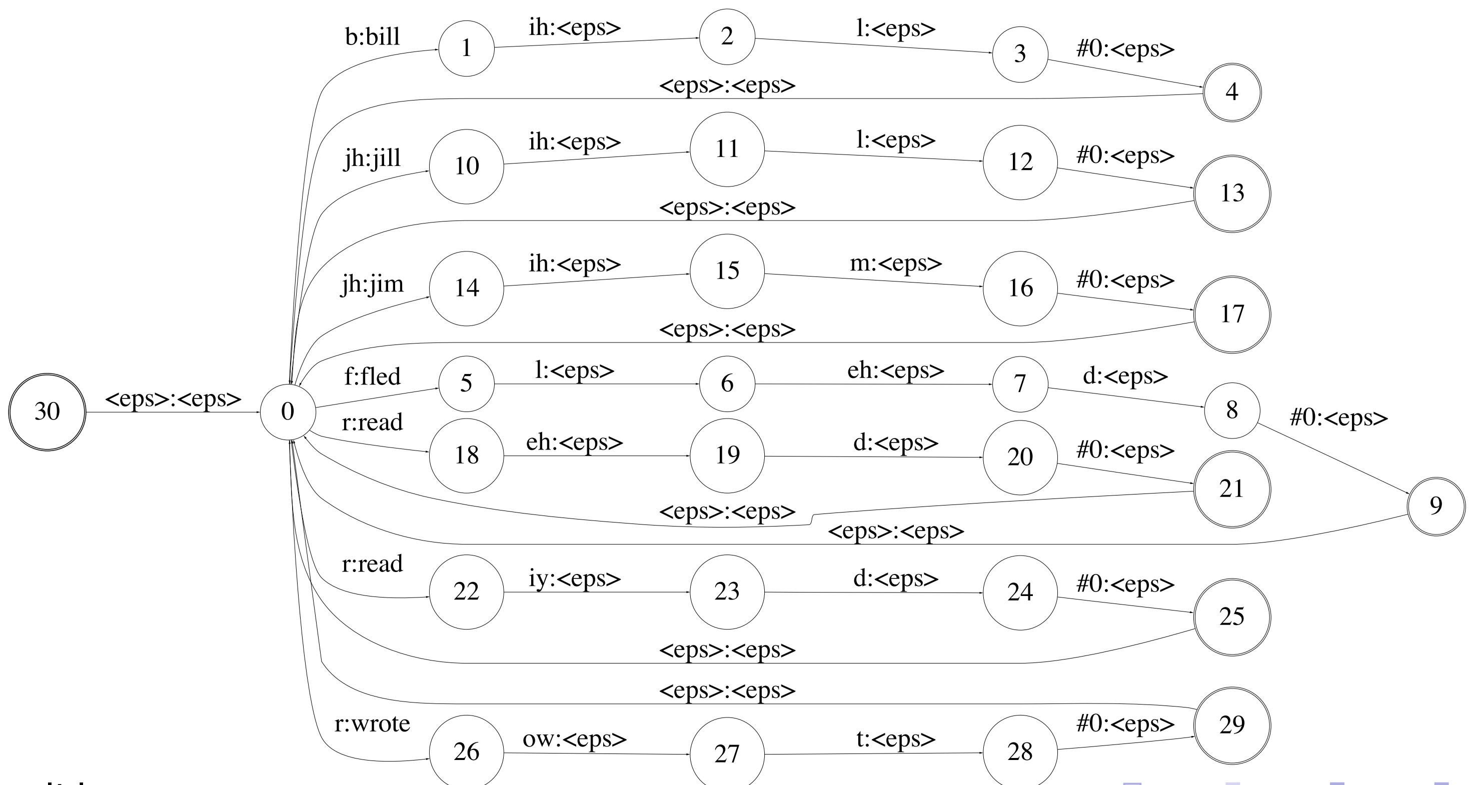
Thanks to Steve Renals for these slides.

+ L, G

G



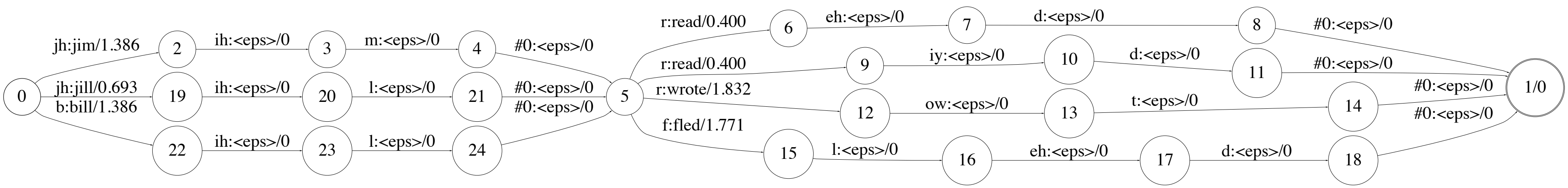
L



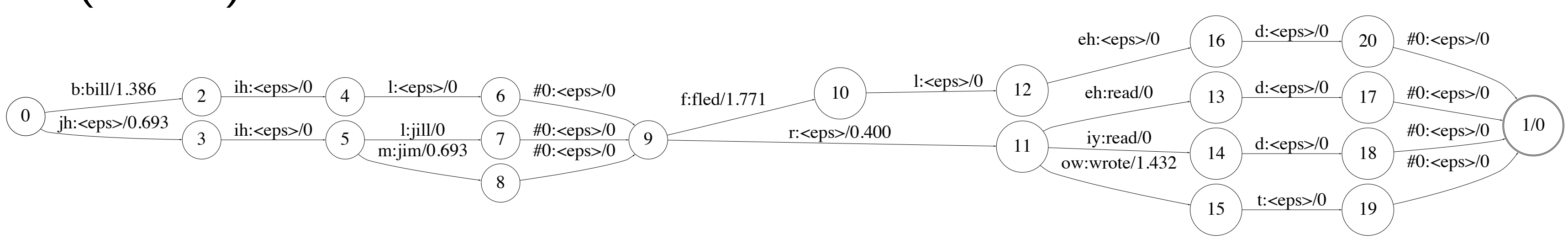
Thanks to Steve Renals for these slides.

+ Determinization (making the FSA deterministic)

$L \circ G$



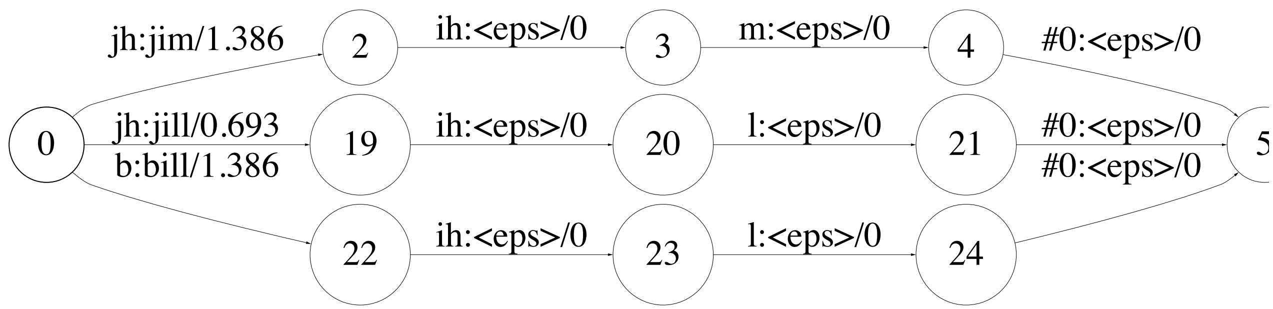
$\det(L \circ G)$



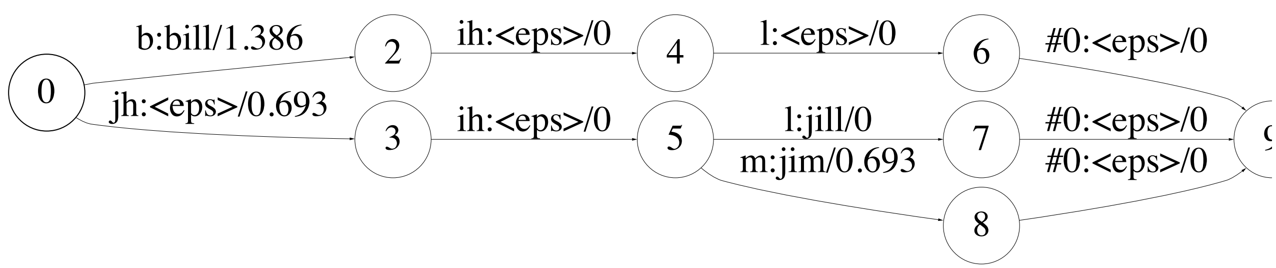
Thanks to Steve Renals for these slides.

Determinization

$$L \circ G$$

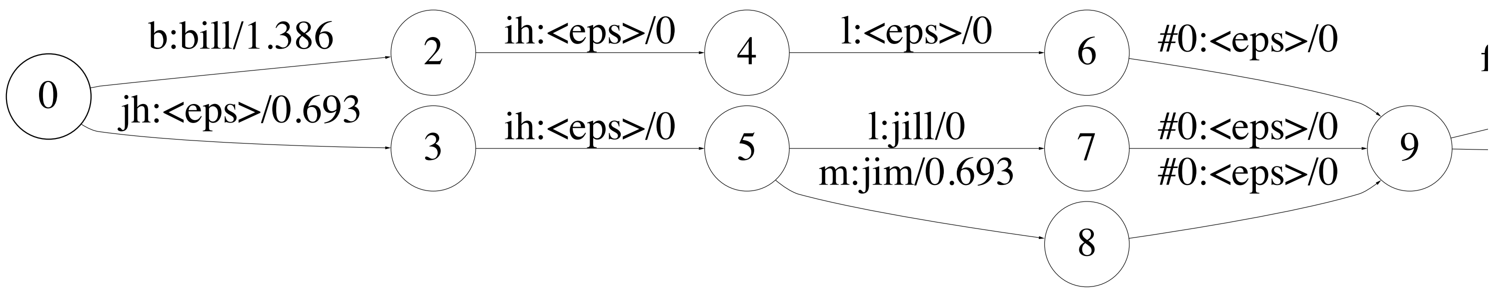


$$\det(L \circ G)$$

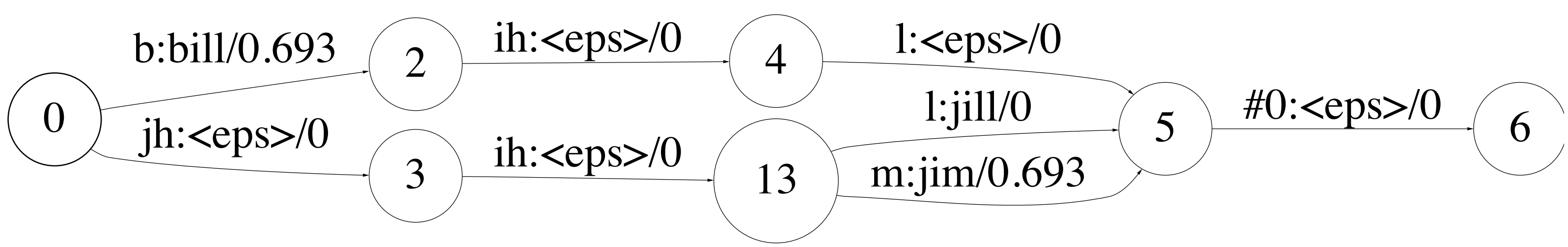


Thanks to Steve Renals for these slides.

+ Minimization



$$\min(\det(L \circ G))$$

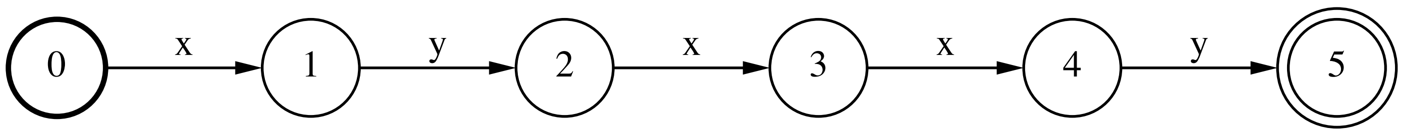


Thanks to Steve Renals for these slides.

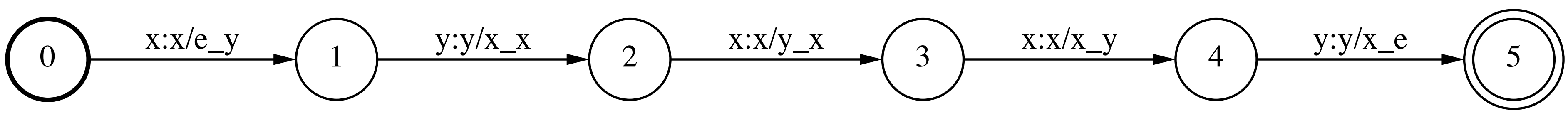
Context Dependency transducer

From phones to triphones

Context-independent "string"



Context-dependency transducer (weights not shown)



(x/e_y – x with left context e (start/end) and right context y)

Thanks to Steve Renals for these slides.

+ Decoding using WFSTs

- We can represent the HMM acoustic model, pronunciation lexicon and n-gram language model as four transducers: H, C, L, G

- Combining the transducers gives an overall “decoding graph” for our ASR system – but minimization and determinization means it is much smaller than naively combining the transducers

- But it is important in which order the algorithms are combined otherwise the transducers may “blow-up” – basically after each composition, first determinize then minimize

- In Kaldi, ignoring one or two details

$$HCLG = \min(\det(H \circ \min(\det(C \circ \min(\det(L \circ G))))))$$

Thanks to Steve Renals for these slides.