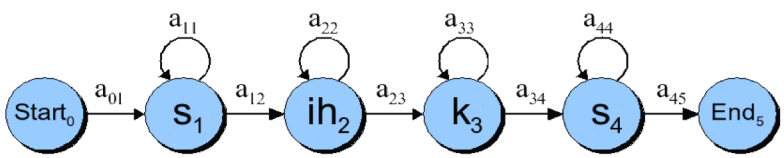
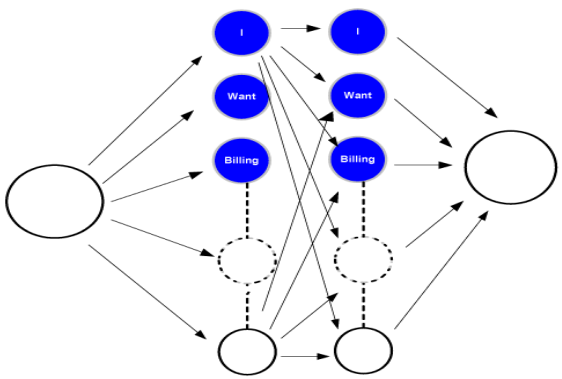
WFST: Weighted Finite State Transducer

September 14, 2018
Prof. Marie Meteer

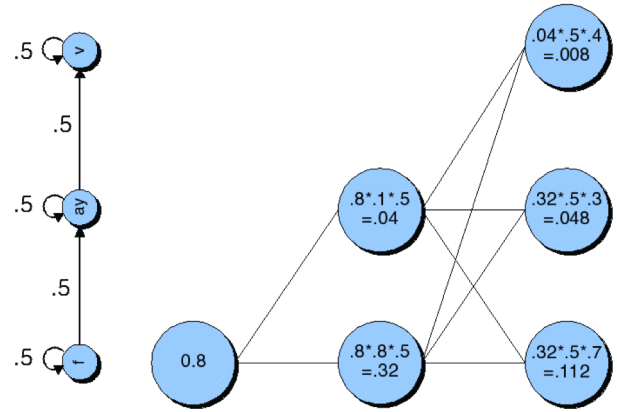
FSAs: A recurring structure in speech



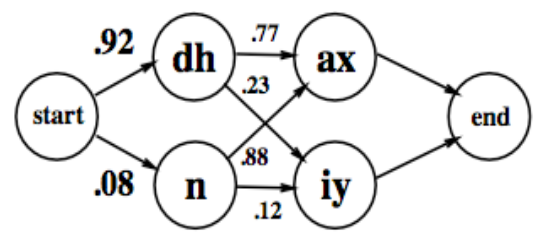
Phonetic HMM



Language model



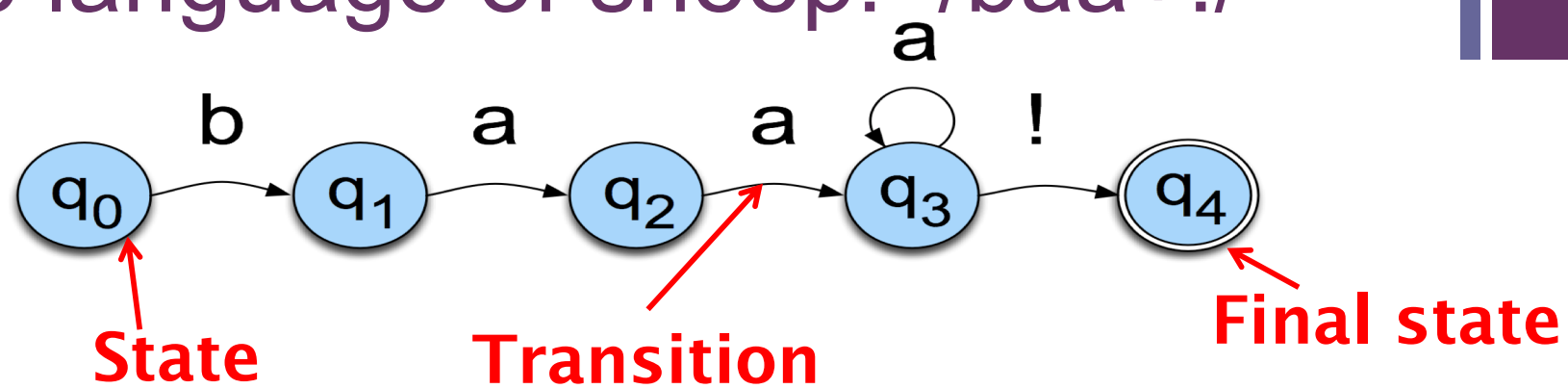
Viterbi trellis



Word model for "the"

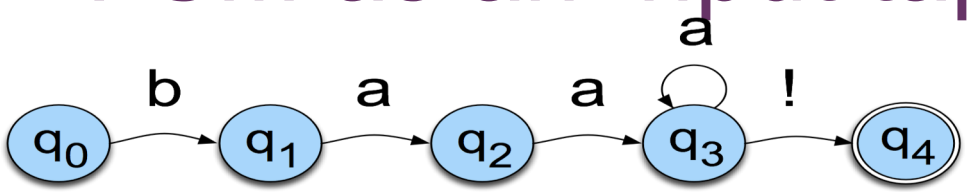
Pronunciation modeling

+ The language of sheep: /baa+!/



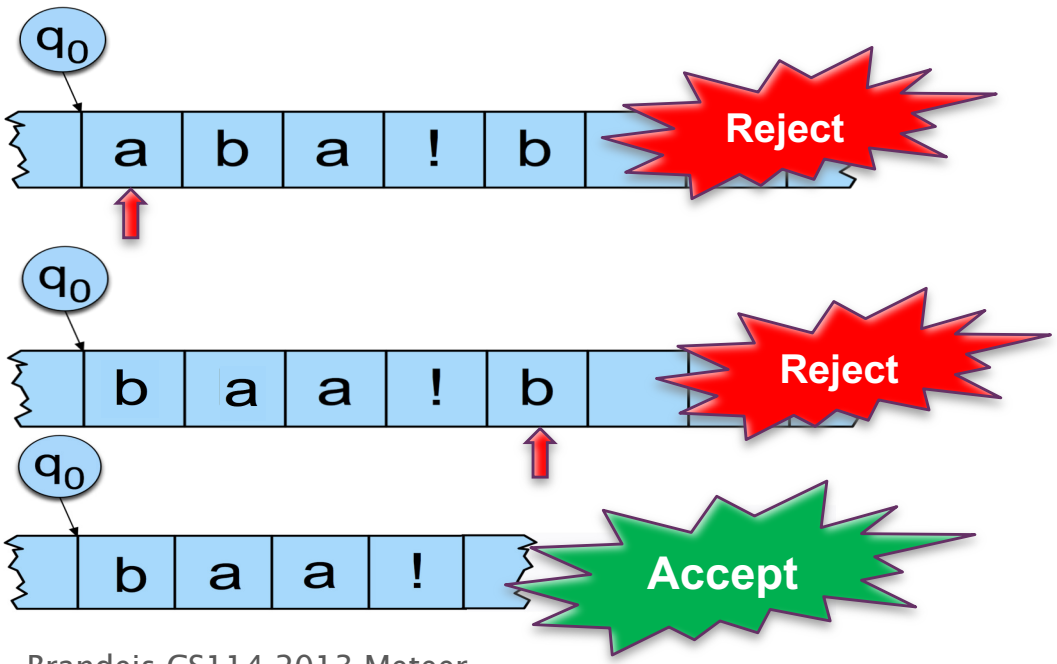
- We can say the following things about this machine
  - It has 5 states
  - b, a, and ! are in its alphabet
  - q_0 is the start state
  - q_4 is an accept state
  - It has 5 transitions

+ FSM as an Input tape acceptor



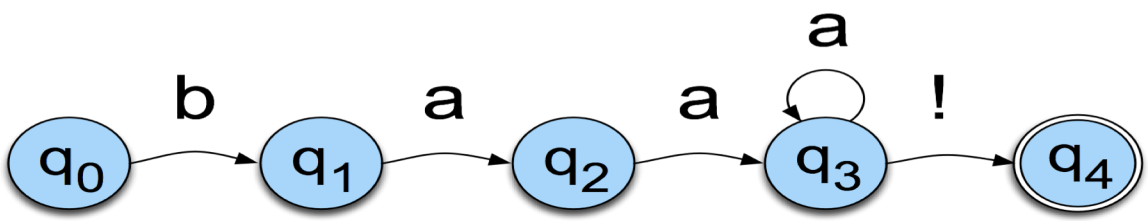
Transition Table

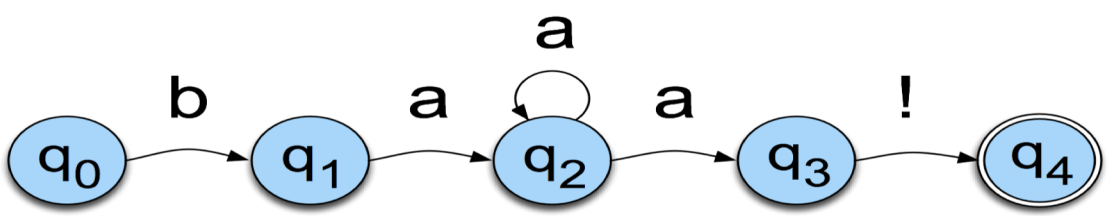
Given an input "tape", does my machine accept or reject that input?



	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				

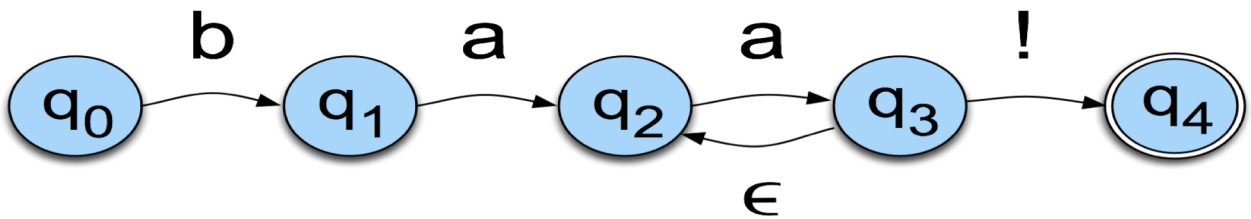
Input:

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



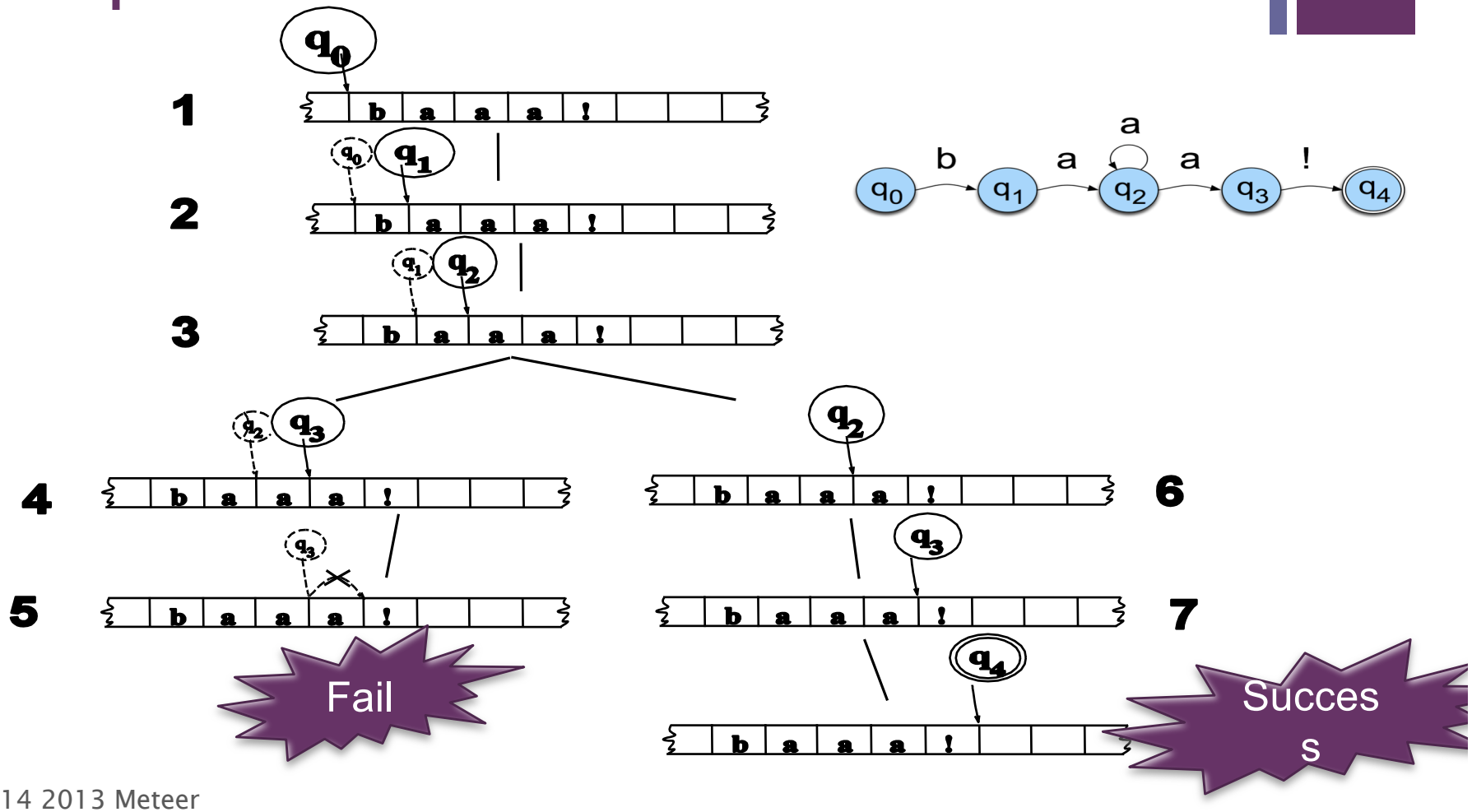
+ Equivalence

- 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

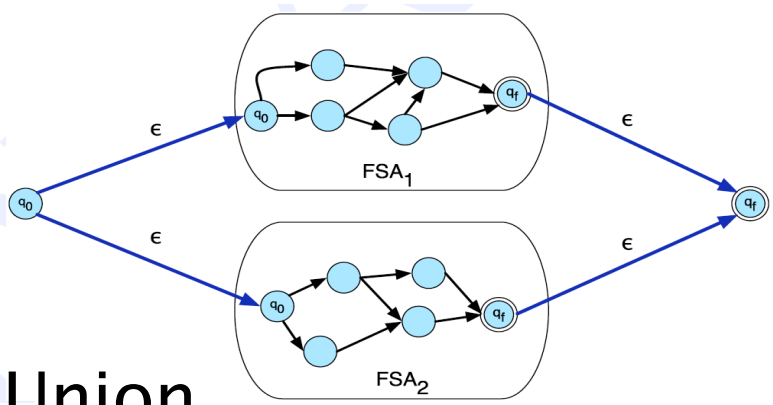
- In a ND FSA *there exists at least one path* through the machine for a string that is in the language defined by the machine.
- *But not all paths* directed through the machine for an accept string lead to an accept state.
- *No paths* through the machine lead to an accept state for a string not in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

+ Example

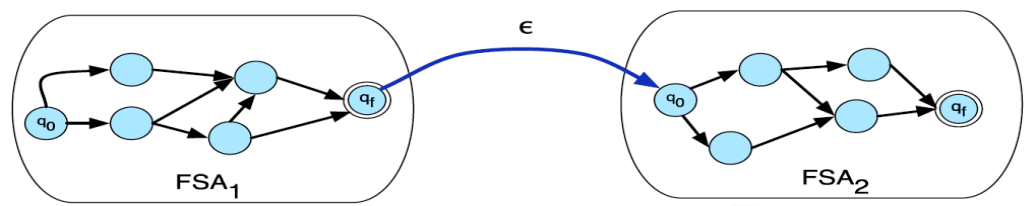


+ Compositional Machines

- 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

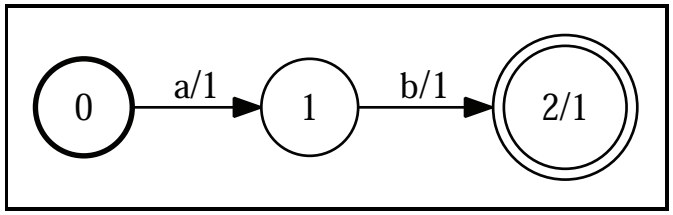


Union



Concatenation

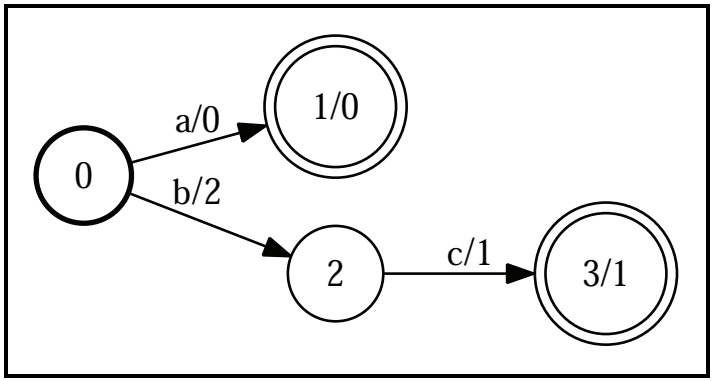
Weighted finite state acceptors



- 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

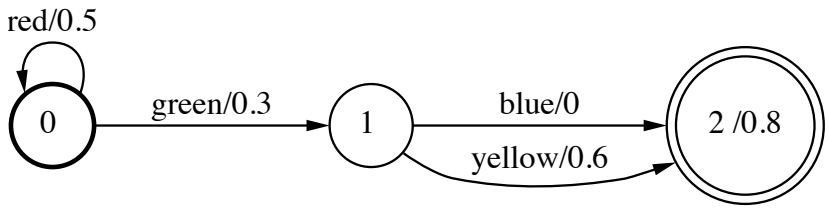
+ Weights vs. costs

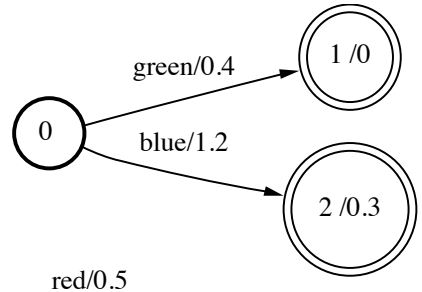


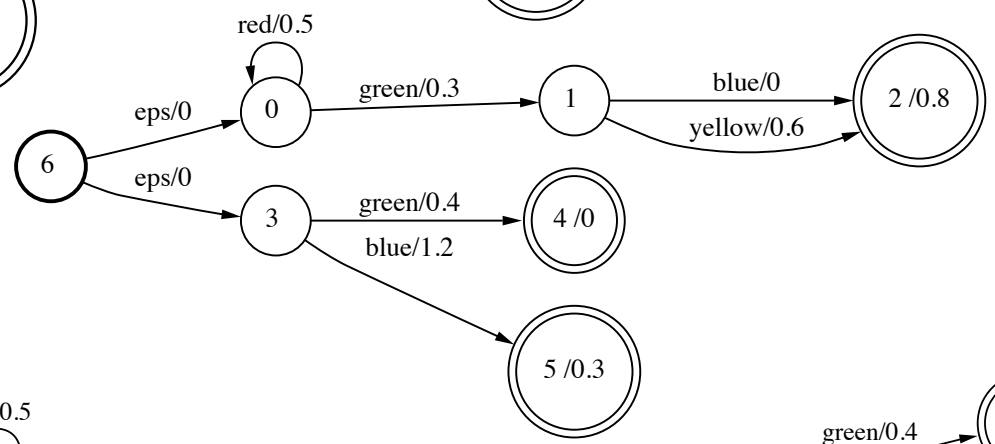
- Use “cost” to refer to the numeric value, and “weight” when speaking abstractly, e.g.:
  - The acceptor above accepts a with unit weight.
  - It accepts a with zero cost.
  - It accepts bc with cost 4=2+1+1
  - State 1 is final with unit weight.
  - The acceptor assigns zero weight to xyz.
  - It assigns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

Thanks for Mirko Hannemann for this slide

+ Combinations

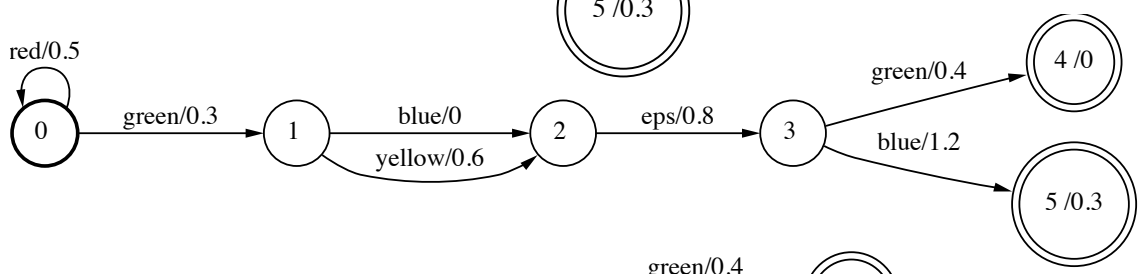




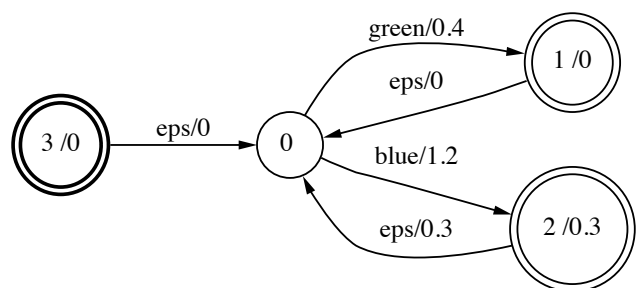


- Union: parallel

- Concatenation: series

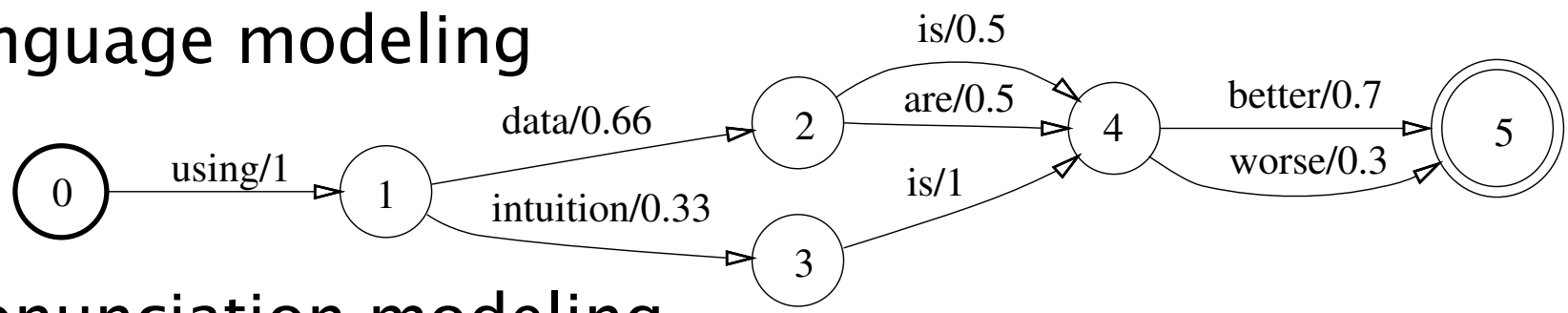


- Kleene closure: arbitrary repetition

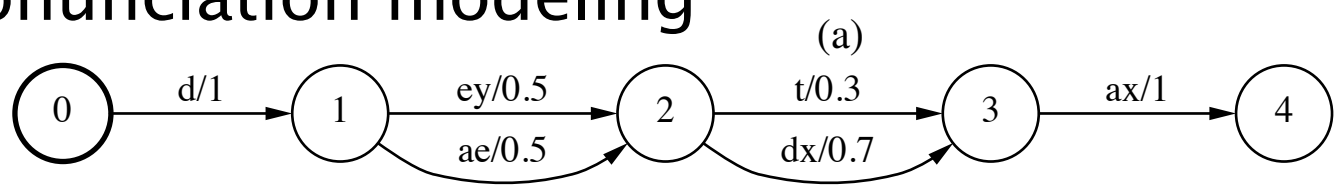


+ WSFAs in speech

Language modeling

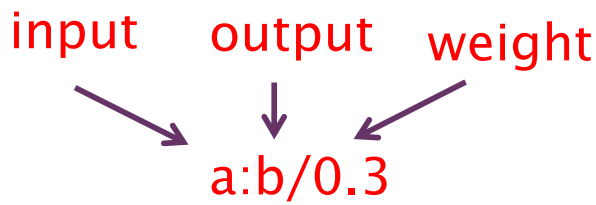


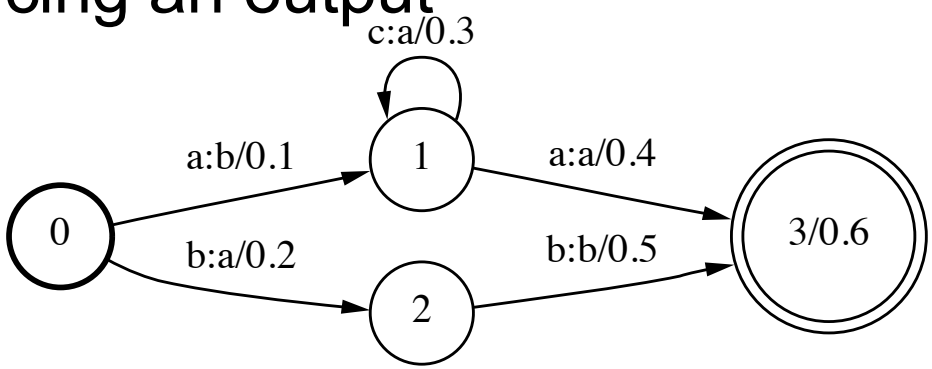
Pronunciation modeling



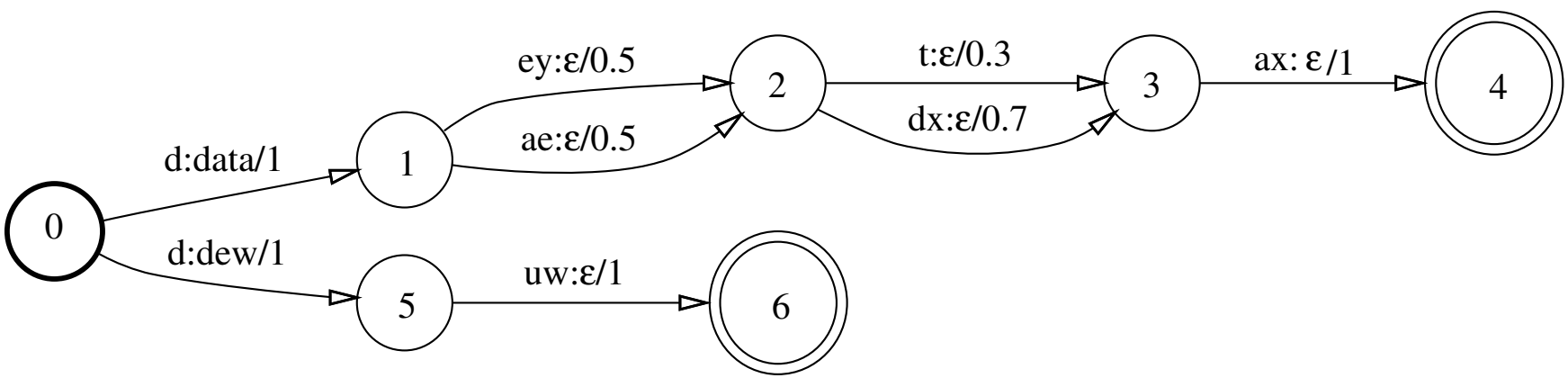
WFS Transducer

- Accept an input while producing an output

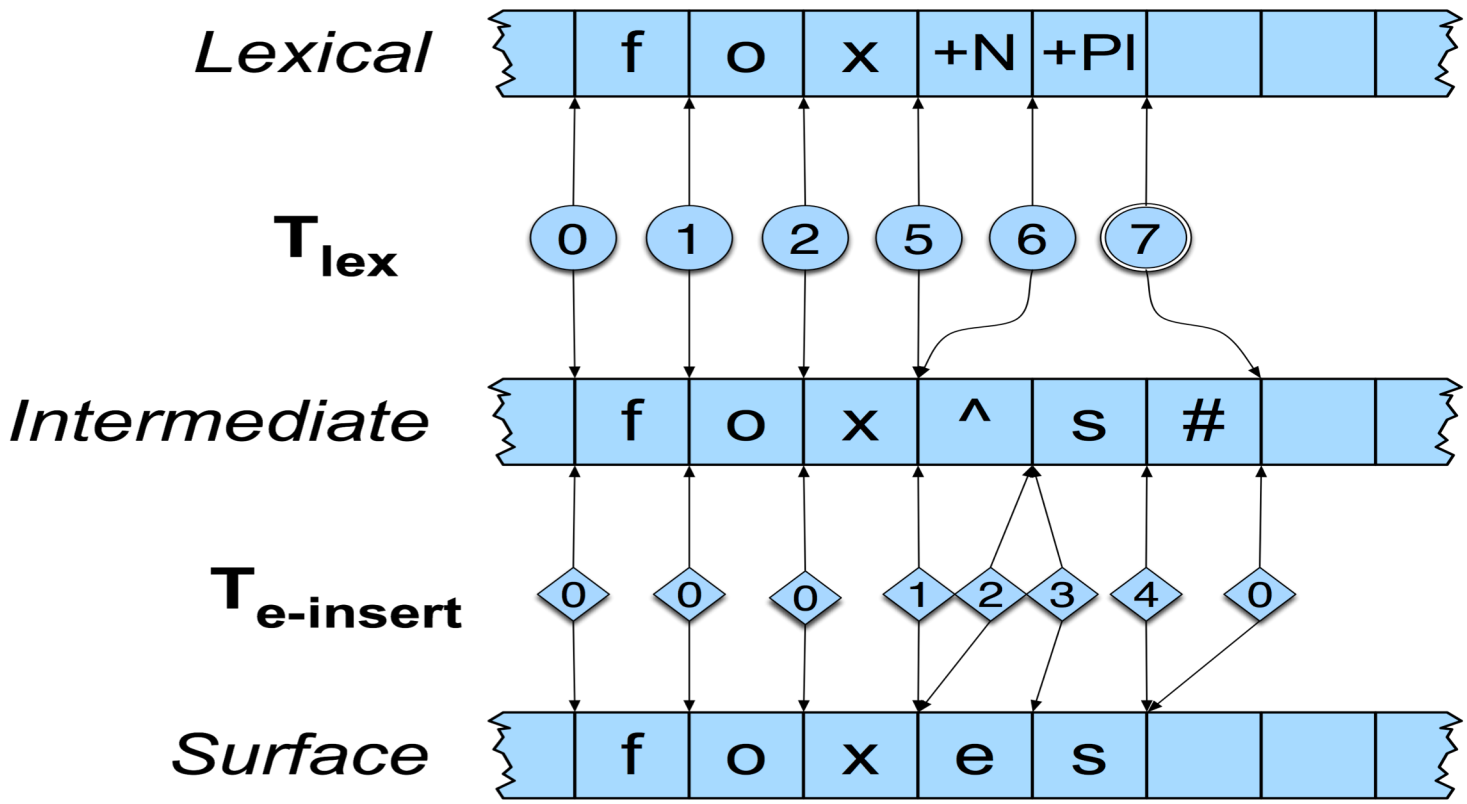




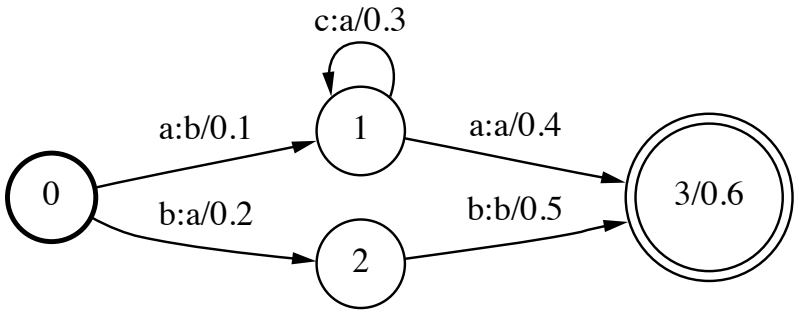
- Input phonemes: output words

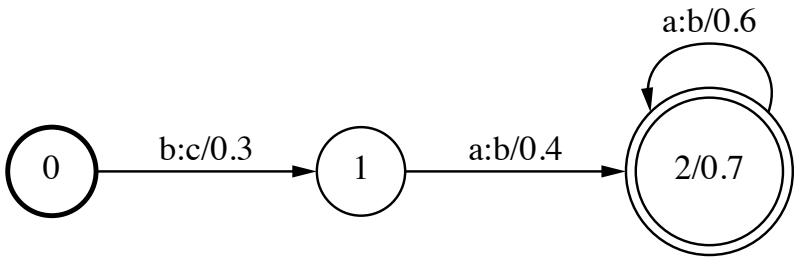


+ FST for morphology: Foxes and Cats

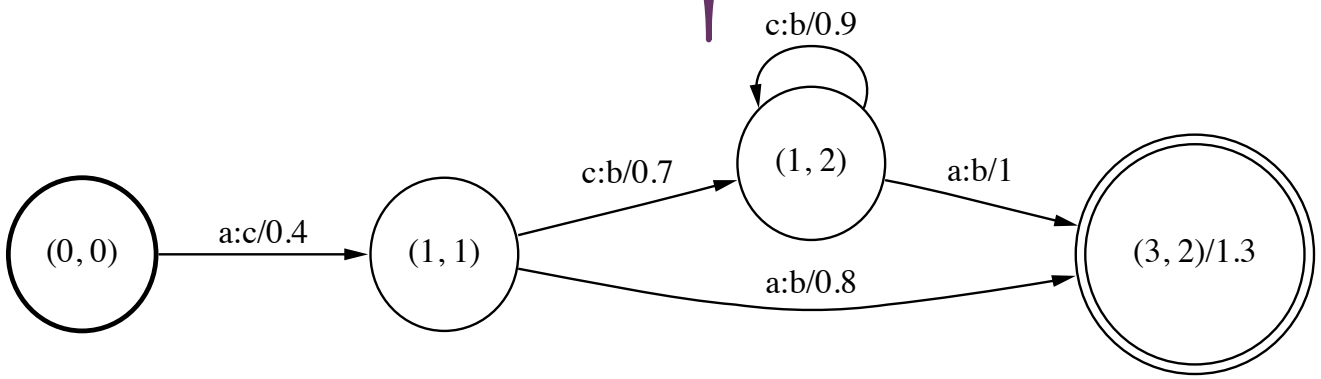


+ Composition



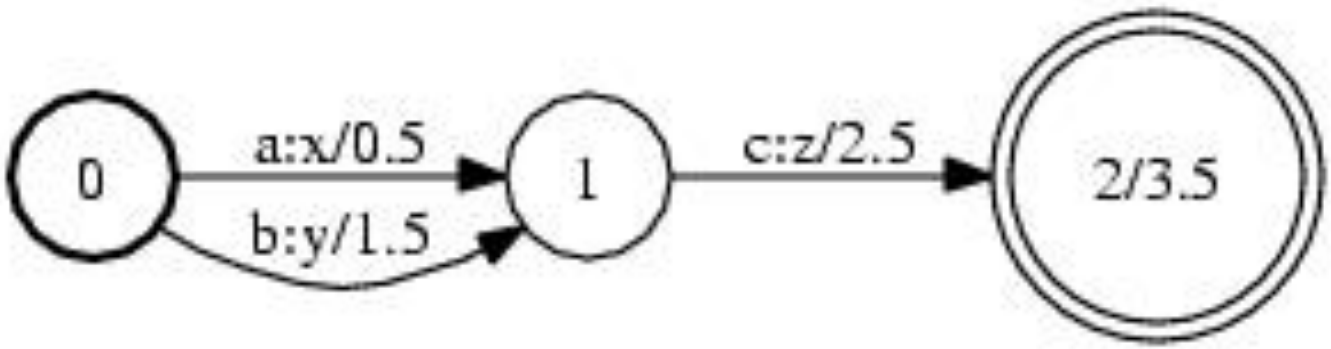


a c a → b a a → c b b



a c a → c b b

+ WFSTs in Action with OpenFST



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