Finite State Methods

Kevin Duh

Intro to NLP, Fall 2019

Outline

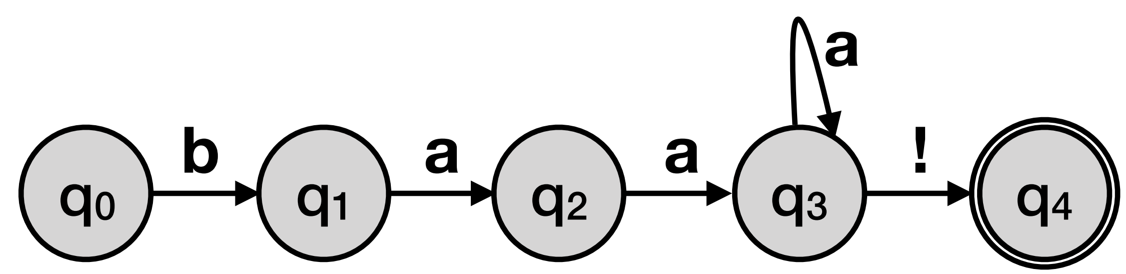
1. Finite-State Automaton (FSA)
2. Finite-State Transducer (FST)
3. Applications in Morphology
4. Semiring

Finite-State Methods in NLP

- Application of Automata Theory, focusing on
  - properties of string sets or string relations
  - with a notion of “bounded dependency”
    - e.g. phonology and morphology. some syntax

Example: SheepTalk

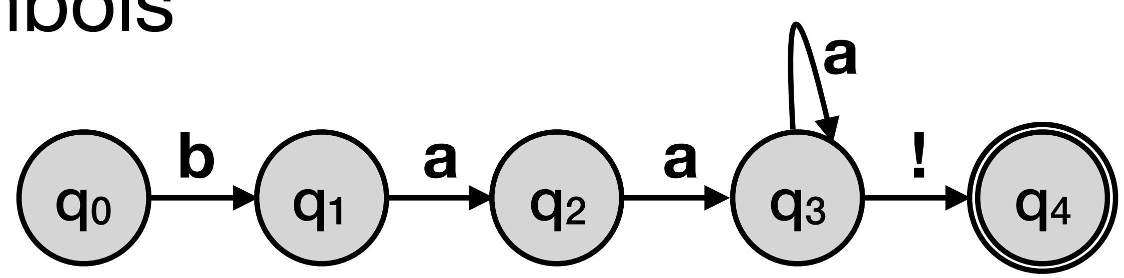
- Here are some strings produced by a sheep:
  - baa!
  - baaa!
  - baaaaa!



- Some strings not produced by a sheep:
  - baabaa!
- We can model this with a regular expression /baa*!/
- We can also model this with a finite-state automaton

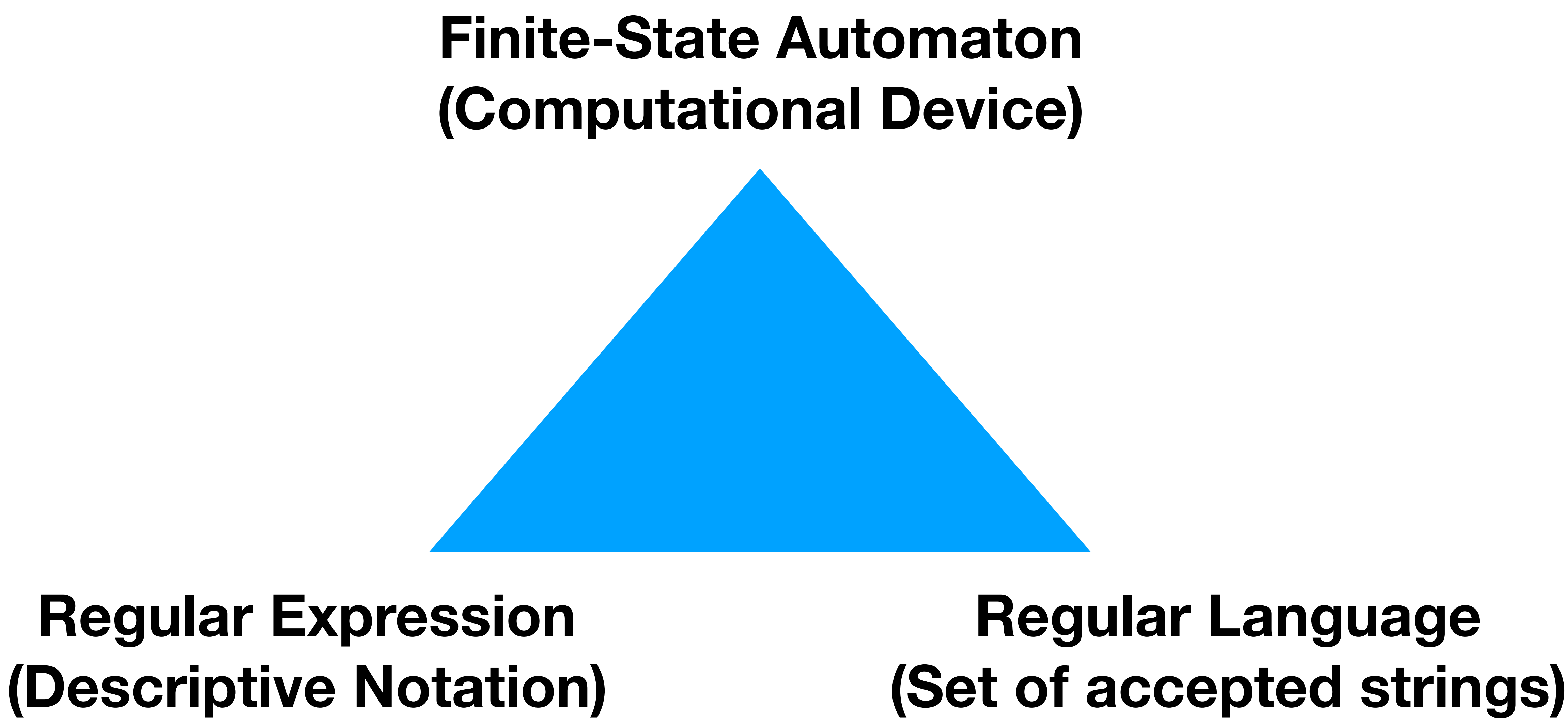
Finite-State Automaton (FSA)

- Definition:
  - $Q = q_0 q_1, \ldots q_{N-1}$: finite set of N states, q_0 is start state
  - Σ: finite input alphabet of symbols
  - F: set of final states in Q
  - Transition function: given state q and input i, returns new state q’



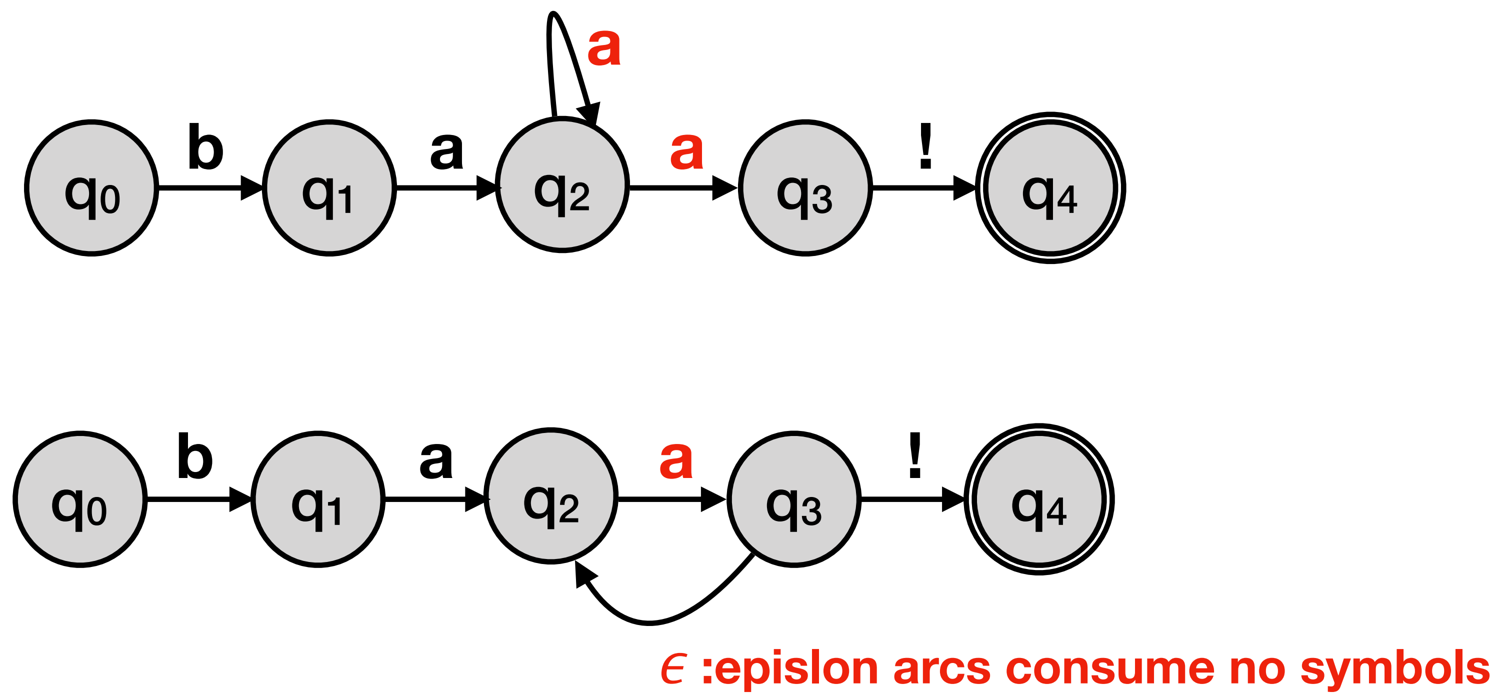
- A string is accepted/recognized by the FSA if it starts in q0 and reaches a valid final state

FSA, Regex, Regular Language



Note: A regular language (e.g. SheepTalk) can contain infinitely many strings, but modeled by one FSA

Non-Deterministic FSA

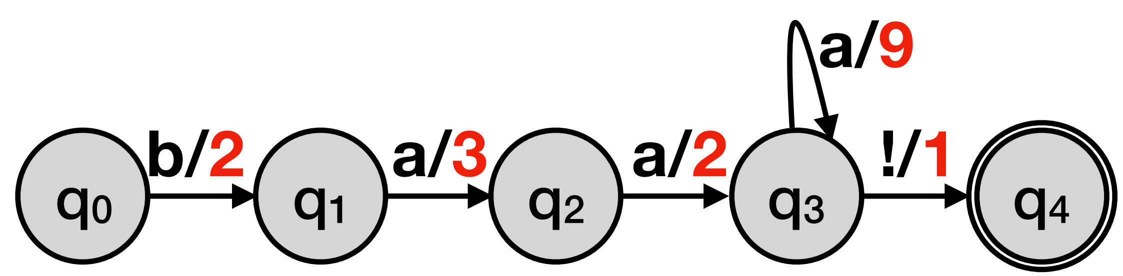


- There are ways to check if a string is recognized by a Non-Deterministic FSAs
- Also possible to convert Non-Deterministic FSAs to Deterministic ones

Operations on Regular Languages

- Suppose L_1 and L_2 are regular languages
  - L_1 is sheeptalk— /baa*!/
  - L_2: 3 strings— ba! baba! bababa!
- **Intersection** $L_1 \cap L_2$: the language consisting of strings in both languages
- **Difference** L_1-L_2: the language consisting of strings that are in L_1 but not in L_2
- **Complementation**: Σ^* - L_1, i.e. set of strings that aren't in L_1

Weighted Finite-State Automata (WFSA)

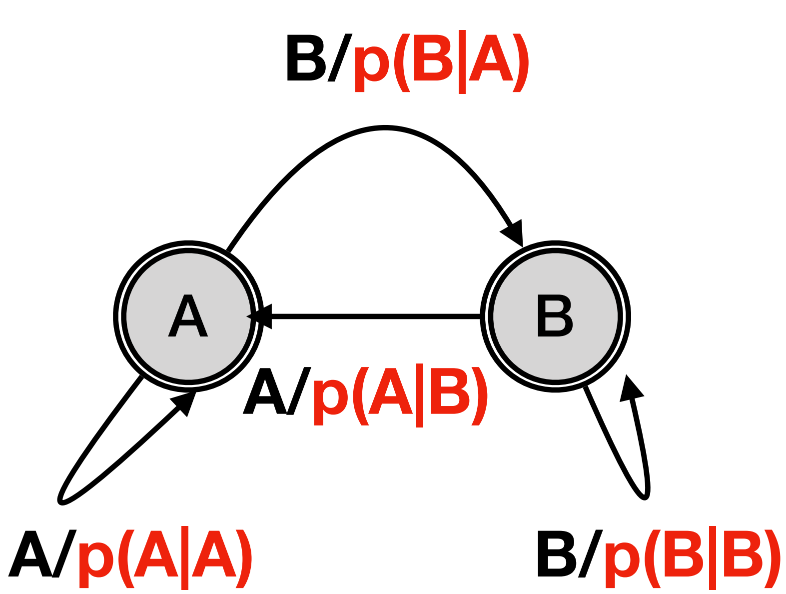


- Associates some weight (e.g. number) to each arc
- This can be useful!

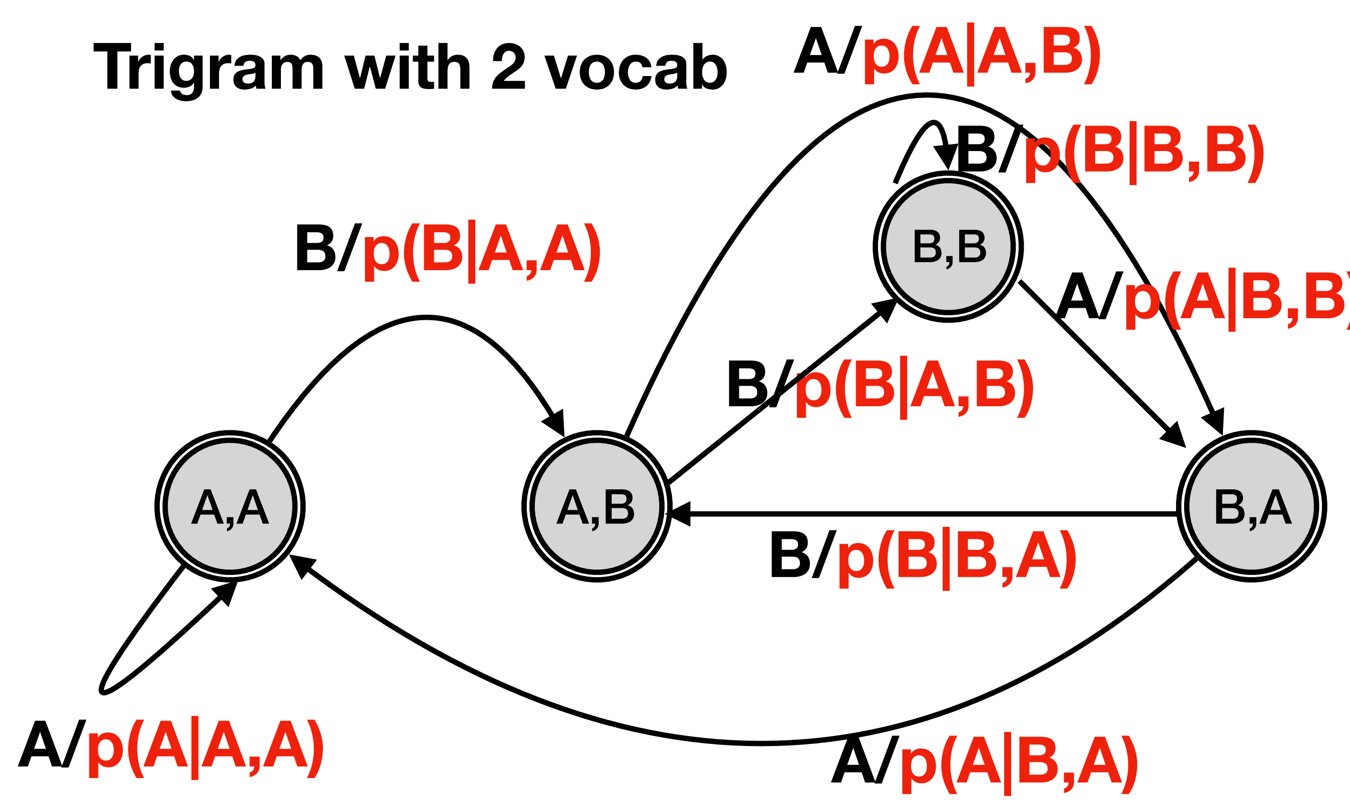
Can N-grams be represented by a WFSA?

- Yes!
- One state for each (n-1)-gram history
- Each arc is a word & probability p(word|history)

Bigram with 2 vocab: A, B



Trigram with 2 vocab

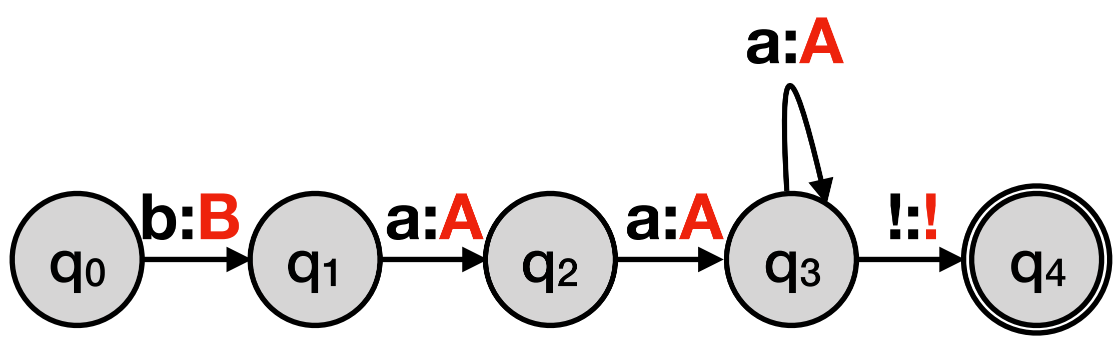


Outline

1. Finite-State Automaton (FSA)
2. Finite-State Transducer (FST)
3. Applications in Morphology
4. Semiring

Finite-State Transducer (FST)

- Maps between two sets of symbols and defines relation between sets of strings

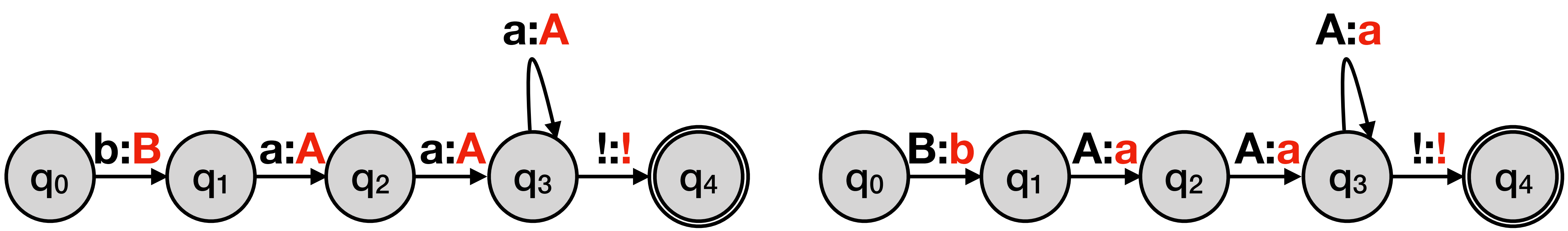


- Can view a recognizer of string pairs
  - Is baa! with BAA! recognized?
- Alternatively, can view as a translator
  - baa! => BAA!

Note: FSA is a special case of FST where input=output

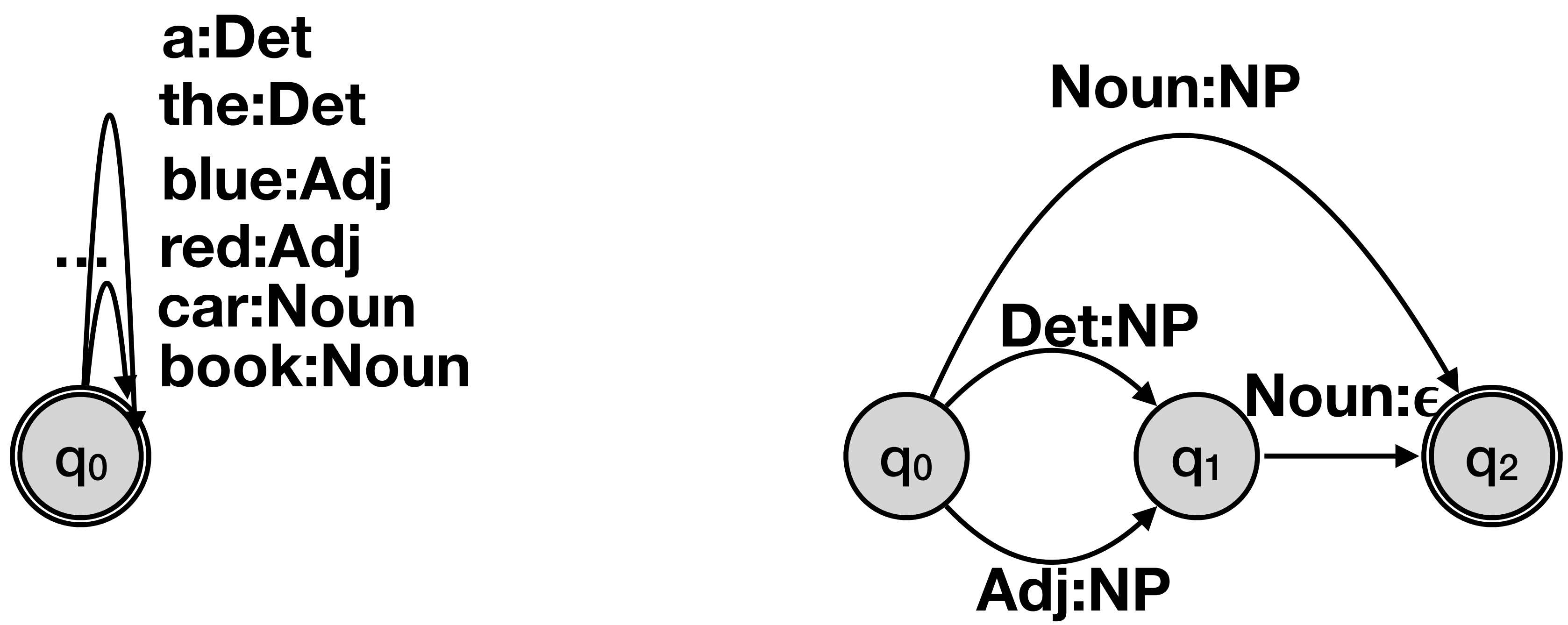
Operations on FSTs

- Inversion: Flips the input and output symbols on each arc



- Composition:
  - T_1 is tranducer from I_1 to O_1
  - T_2 is transducer from O_1 to O_2
  - $T_1 \circ T_2$ maps I_1 to O_2

Composition is very useful!



What happens when you compose these two transducers?

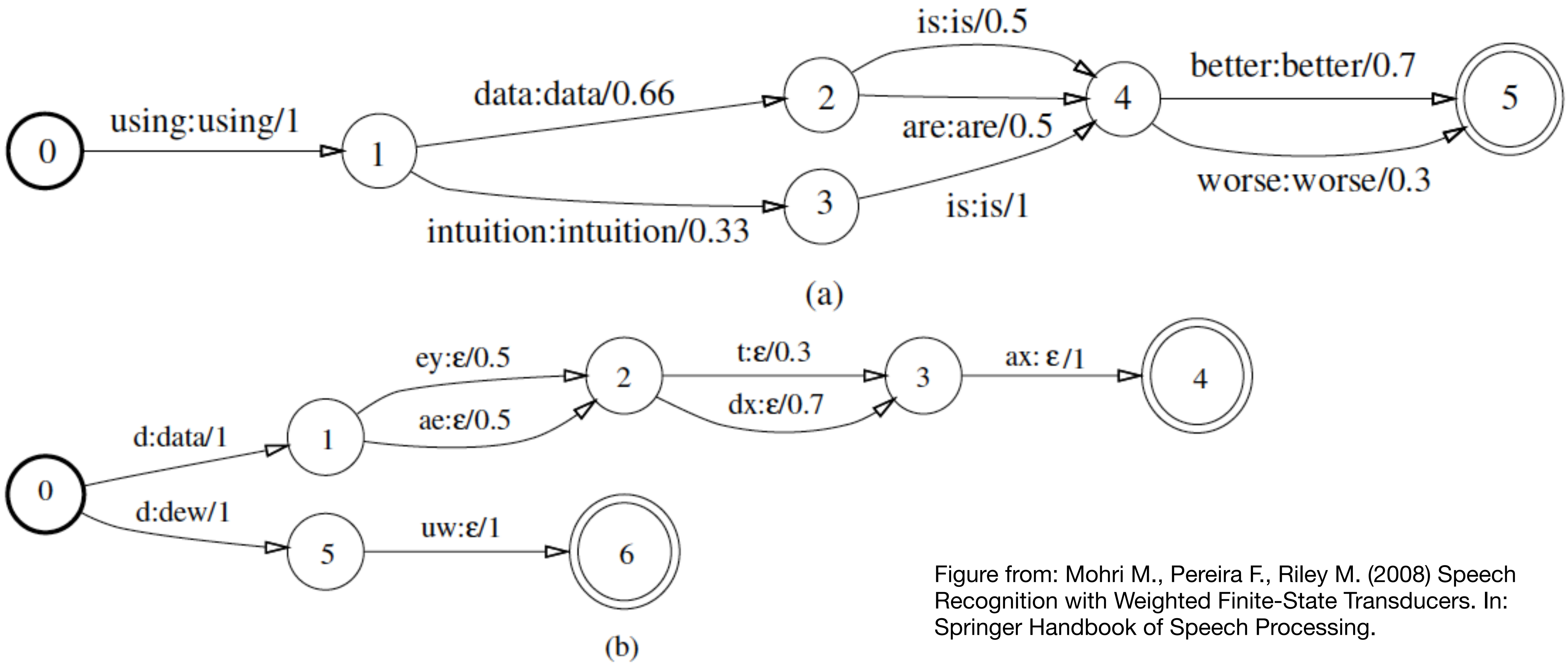


Figure from: Mohri M., Pereira F., Riley M. (2008) Speech Recognition with Weighted Finite-State Transducers. In: Springer Handbook of Speech Processing.

<u>Composition allows us to declaritively describe transform on a set of strings:</u>

e.g. Speech Recognition: H o C o L o G

G = Language Model FSA

L = Pronunciation Lexicon, FST mapping (context-independent) phones to words

C = FST mapping context-dependent phone sequence to context-independent ones

H = FST mapping acoustic model states to context-dependent phones

How to compose FSTs?

- For every state in $s \in T_1$ and $t \in T_2$, create (s,t)
- Create arc from (s_1,t_1) to (s_2,t_2) if
    - There's arc from s_1 to s_2 with output label i
    - And there's arc from t_1 to t_2 with input label i
- A little bit more complicated for epsilon arcs

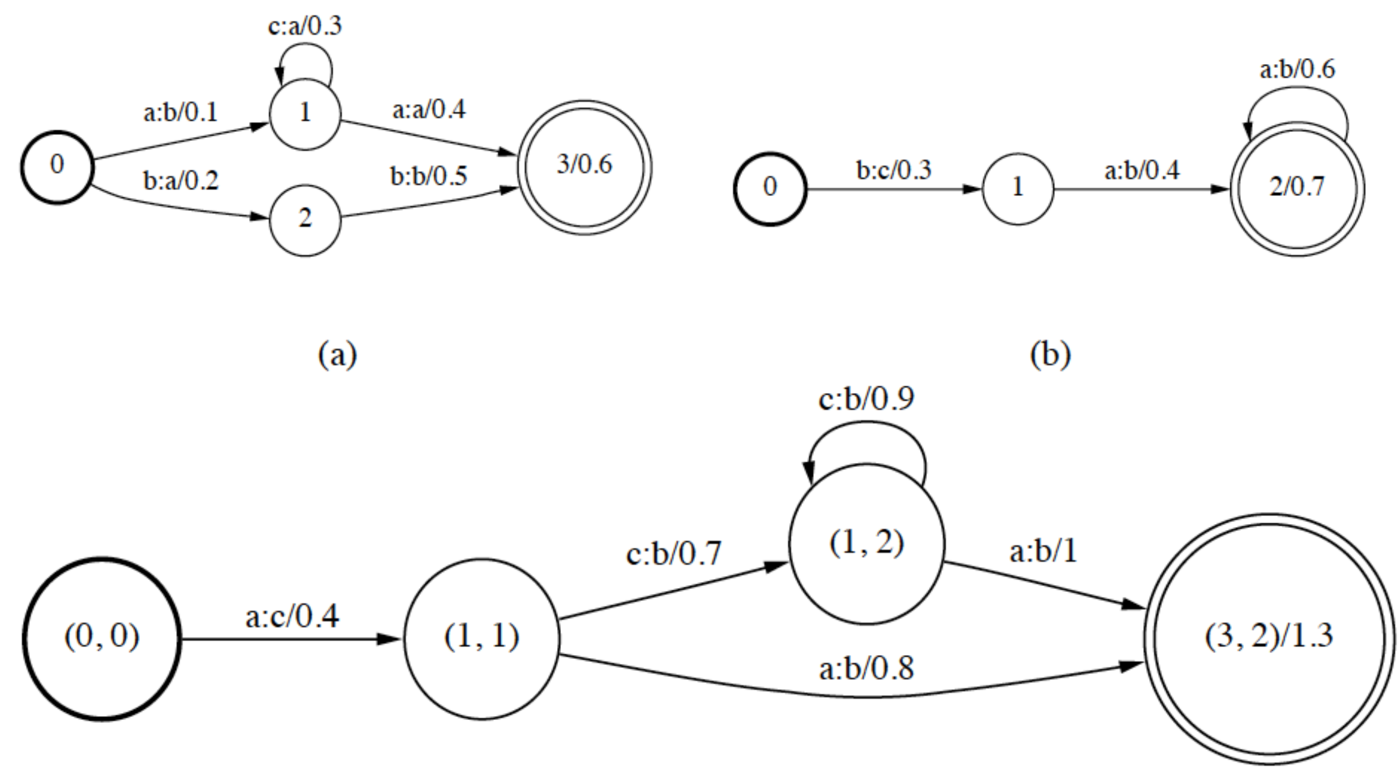


Figure from: Mohri M., Pereira F., Riley M. (2008) Speech Recognition with Weighted Finite-State Transducers. In: Springer Handbook of Speech Processing.

Outline

1. Finite-State Automaton (FSA)
2. Finite-State Transducer (FST)
3. Applications in Morphology
4. Semiring

Morphological Parsing

Input	**Output**
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +PL
caught	(catch +V +PAST-PART) or (catch +V +PAST)
merging	merge +V +PRES-PART
(word in surface form)	(lemma and morphological features)

Morphological Parsing with FSTs

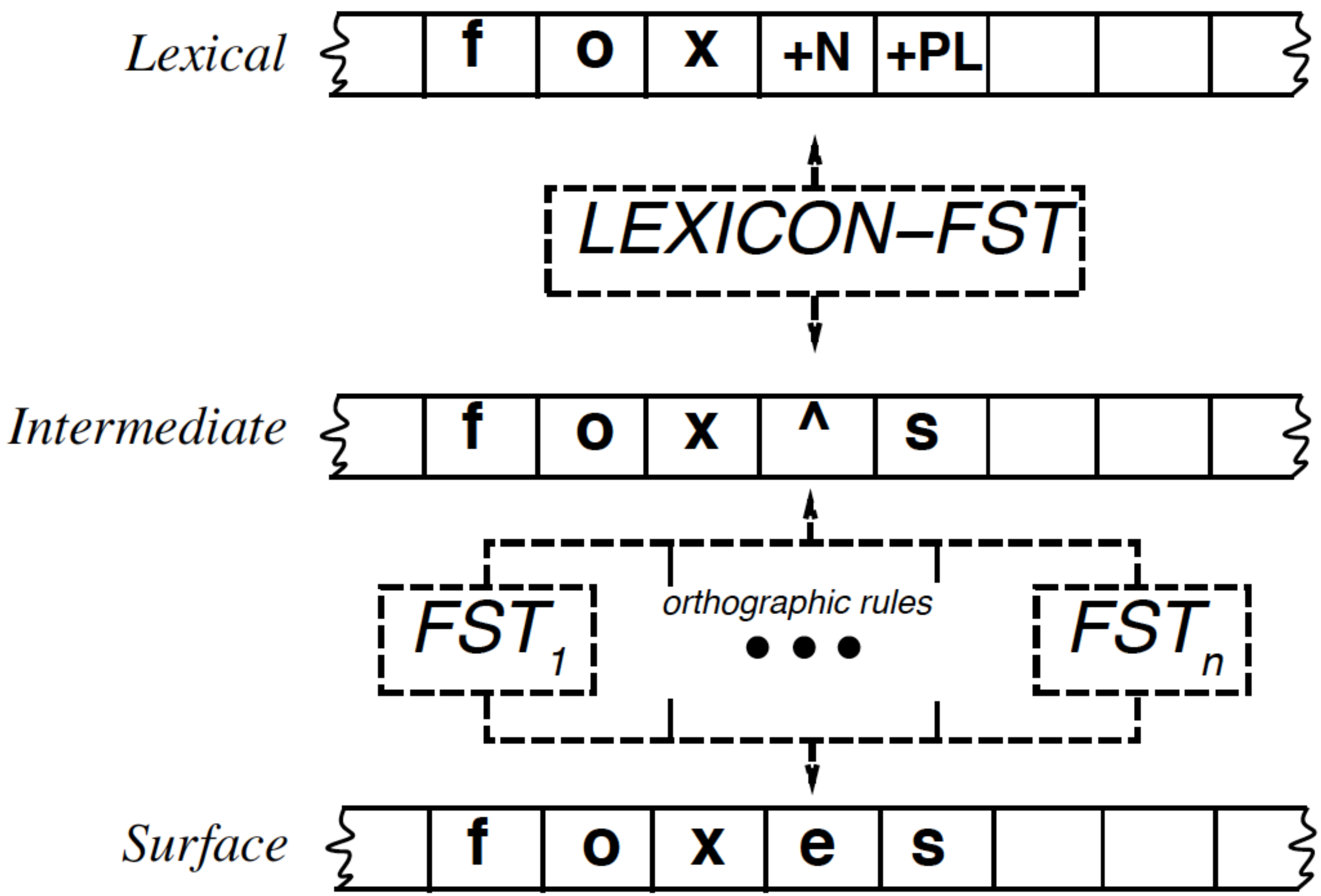
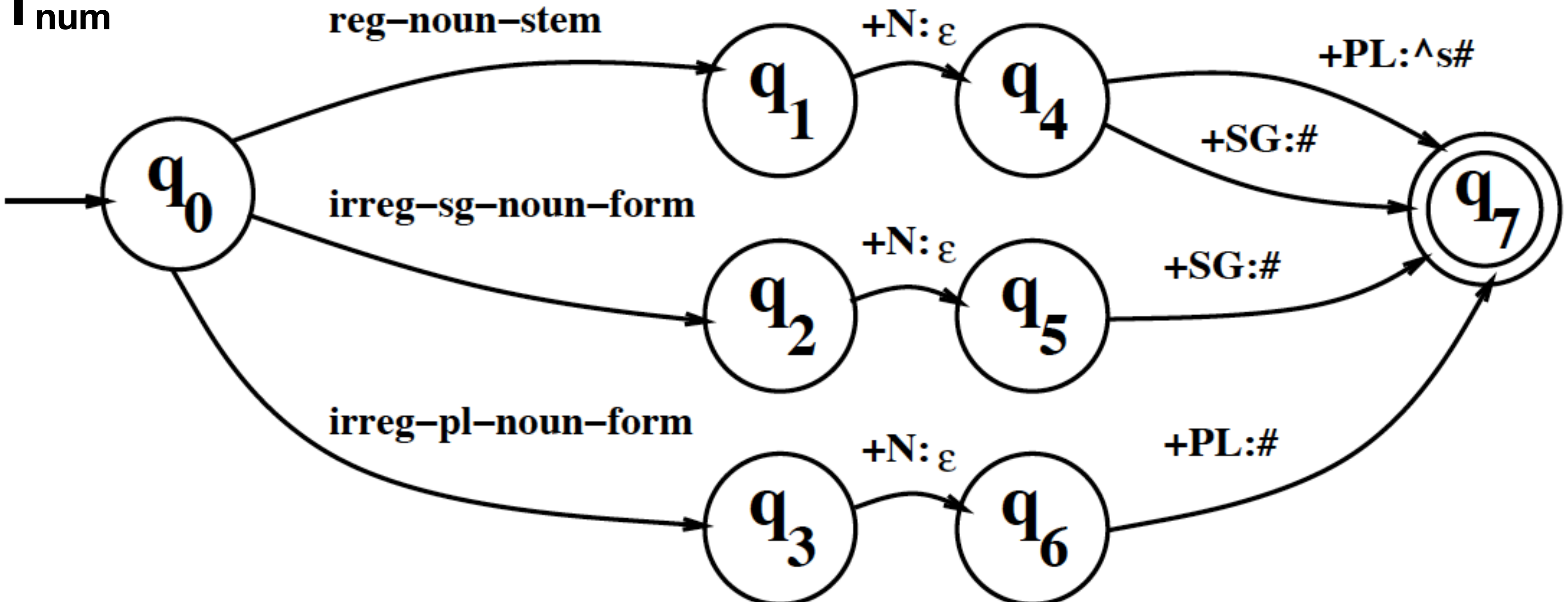


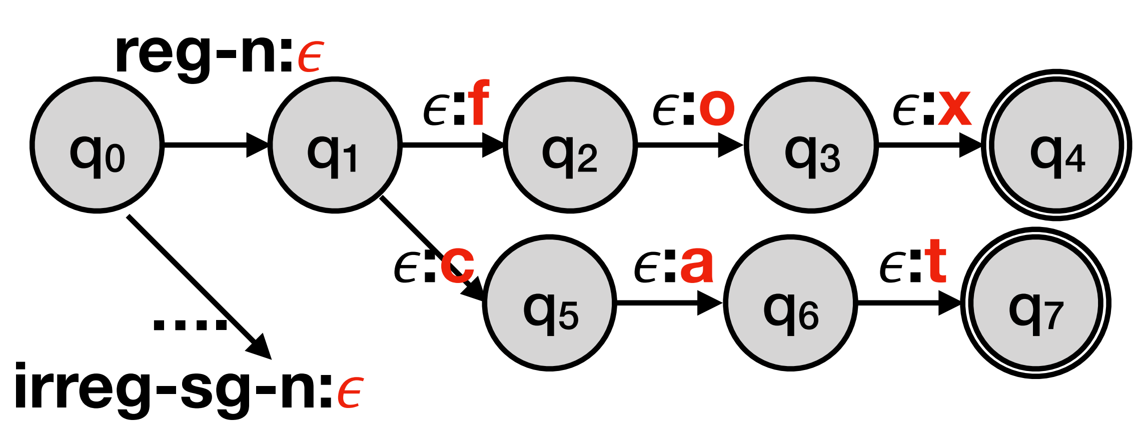
Figure: Jurafsky & Martin (2009) Speech & Language Processing, 2nd ed., Pearson

T_{num}



T_{stem}

reg-noun-stem:fox
reg-noun-stem:cat
irreg-sg-noun-form:sheep
irreg-pl-noun-form:sheep
irreg-sg-noun-form:mouse
irreg-pl-noun-form:mice



$T_{num} \circ T_{stem}$

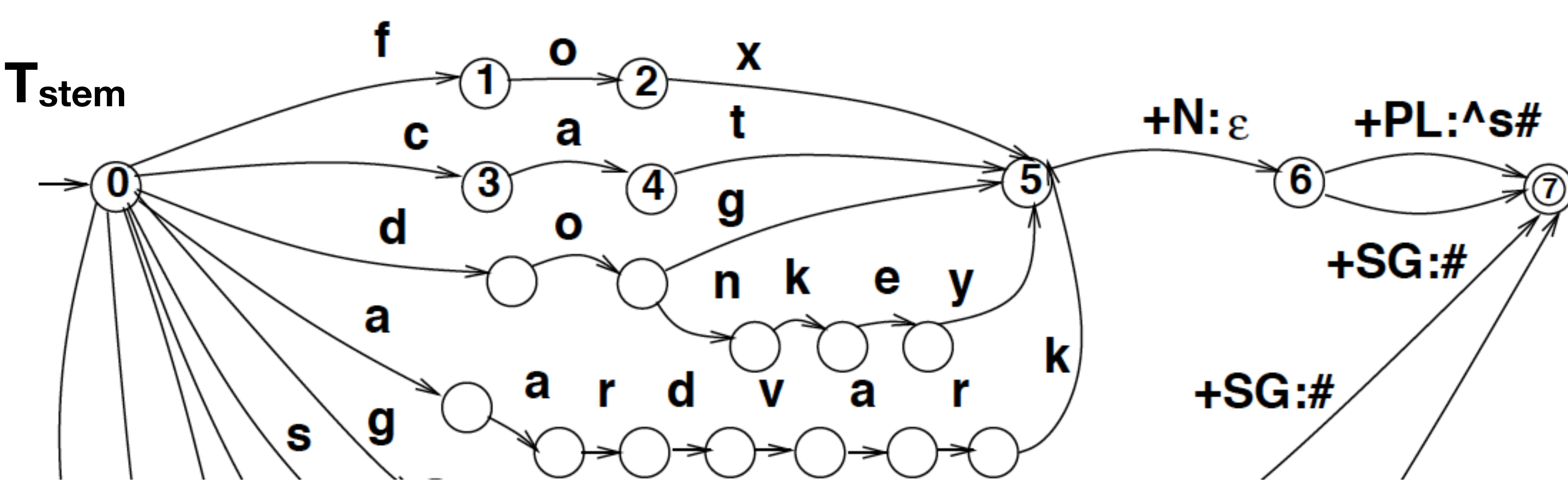
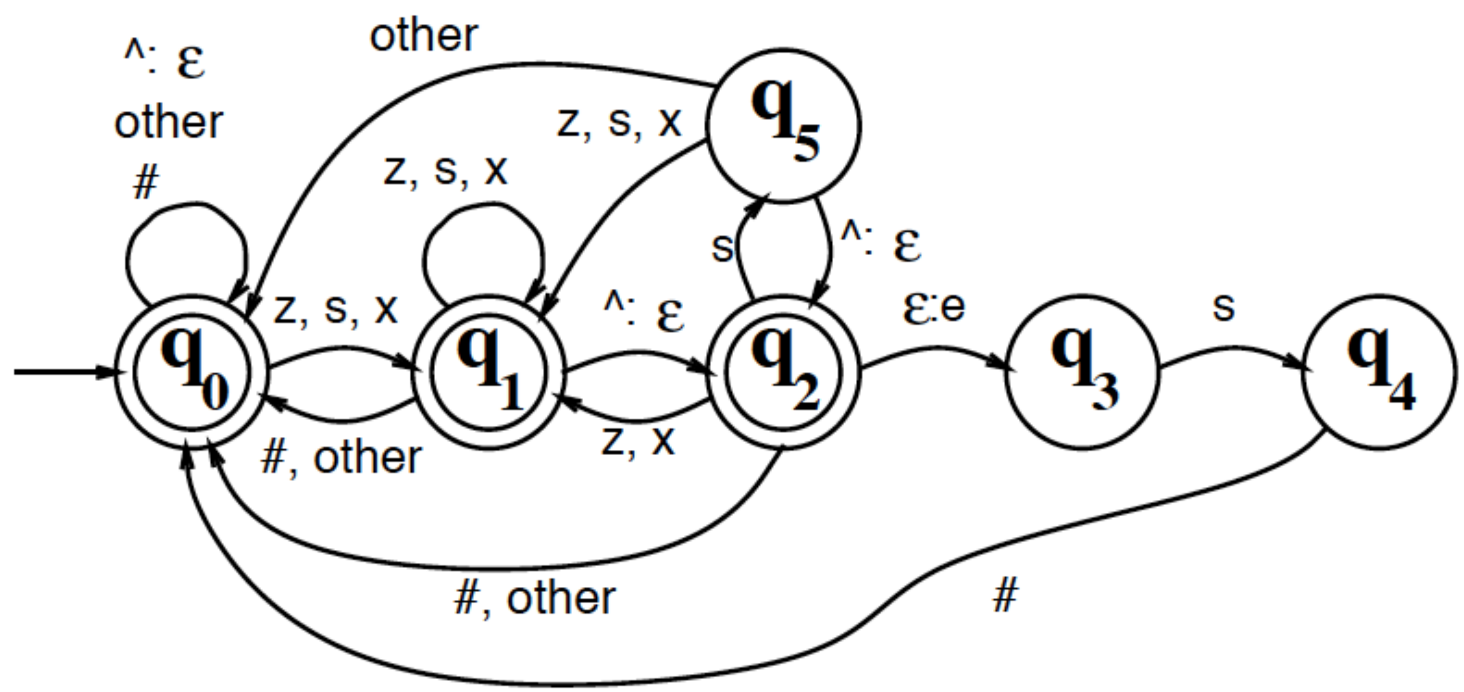
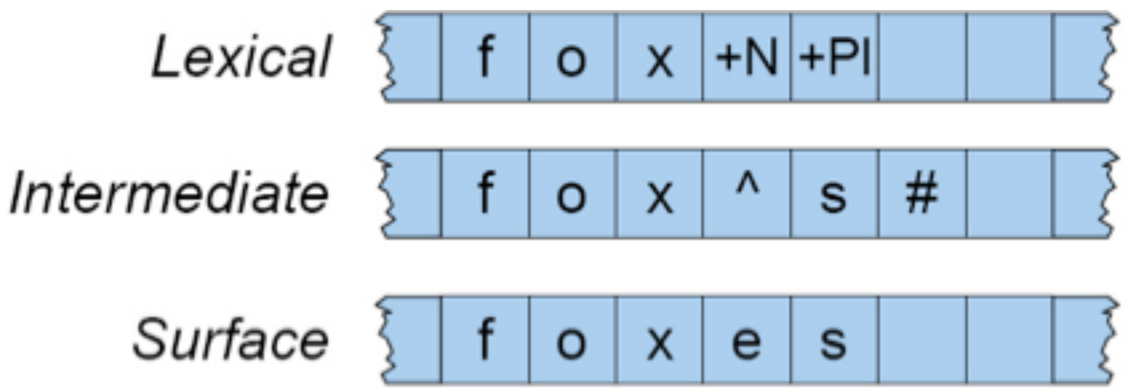


Figure: Jurafsky & Martin (2009) Speech & Language Processing, 2nd ed., Pearson

Orthography Rules:
“foxs” is not correct

$$\varepsilon \rightarrow e \,/ \left\{ \begin{array}{c} x \\ s \\ z \end{array} \right\} \hat{}\ \_\_\_\ s\#$$





$T_{num} \circ T_{stem}$

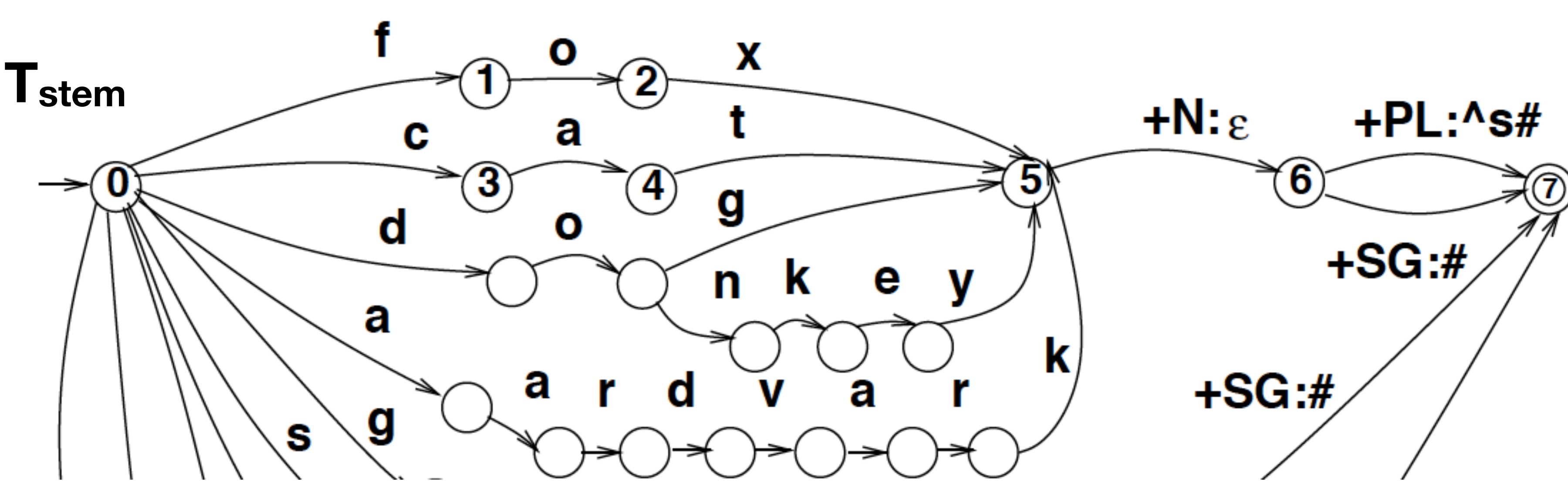


Figure: Jurafsky & Martin (2009) Speech & Language Processing, 2nd ed., Pearson

Example in Turkish

[Oflazer (1993), Two-Level Description of Turkish Morphology]

Input		
	Morpheme Structure	Gloss
		English meaning
çalışmanın		
	çalış+mA+Hn+nHn	[V(çalış)+VtoN(ma)+2PS-POSS+GEN]
		of your work(ing)
	çalış+mA+nHn	[V(çalış)+VtoN(ma)+GEN]
		of the work(ing)

Rules for Phonology/Orthography, e.g.

H:0 ⇒ V (':') +:0 ___

An H vowel is deleted if the last phoneme of the stem it is being affixed to is a vowel. For example:

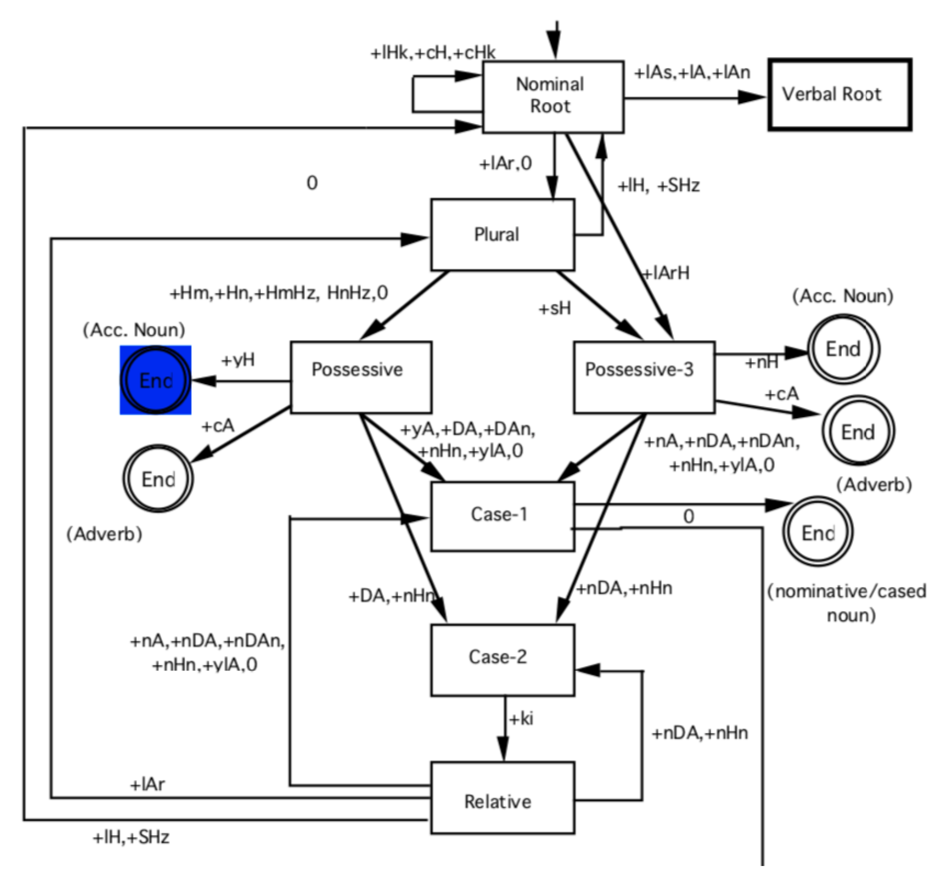
Lexical: `masa+Hm` `N(table)+1PS-POSS`
Surface: `masa00m` `masam`

X:0 ⇔ C (':') +:0 ___ (C) V

This rule deletes the beginning s, n, or a y of a suffix when it gets affixed to a stem ending in a consonant.

Lexical: `ev+nHn` `N(house)+GEN`
Surface: `ev00in` `evin`

Morphotactics FST: order of morphemes

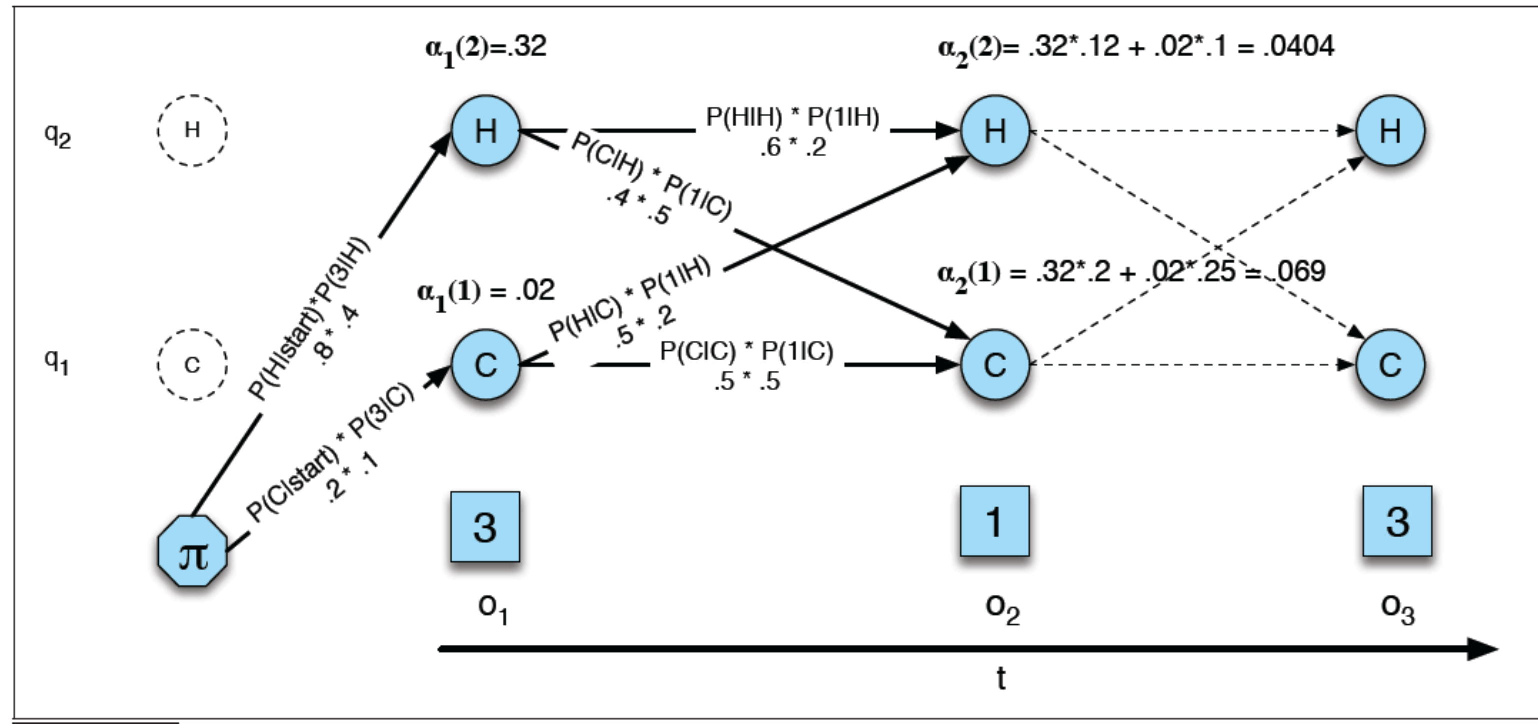


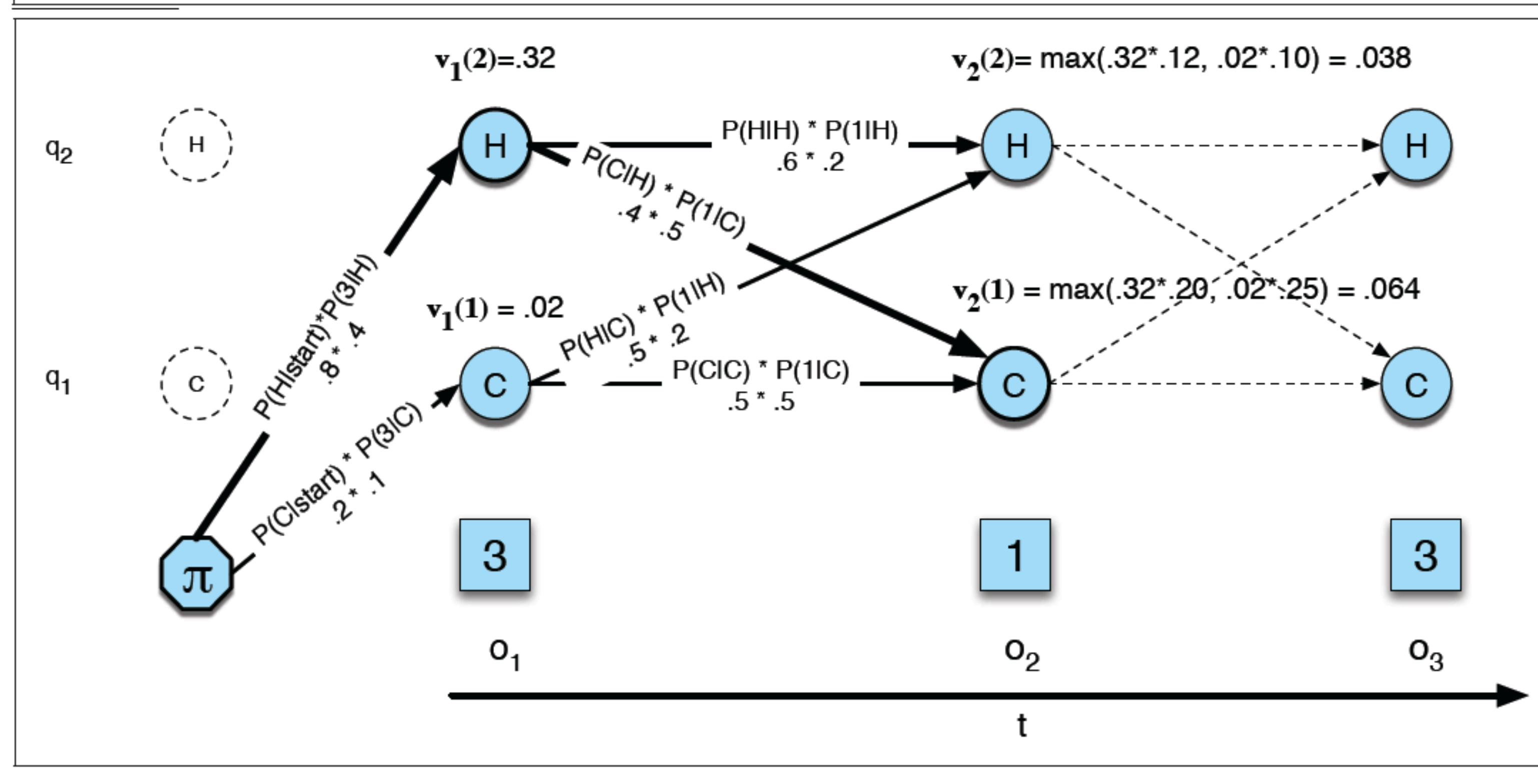
Outline

1. Finite-State Automaton (FSA)
2. Finite-State Transducer (FST)
3. Applications in Morphology
4. Semiring

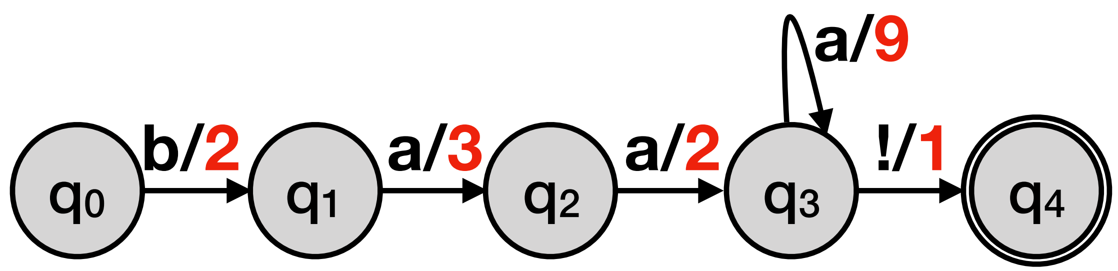
Recall Viterbi vs Forward Algorithms in HMMs

- The way we compute is almost the same! Just swap max & sum
- Is there a way to generalize this on arbitrary FSA/FST?
  - Yes! We'll do so with semirings





Semirings



- Define weights as a semiring:
  - K: set of values
  - ⊕ : associative, commutative, has $\underline{0}$ as identity
  - ⊗ : associative, has 1 as identity, has $\underline{0}$ as annihilator, distributes with respect to ⊕
  - $\underline{0}$ and $\underline{1}$

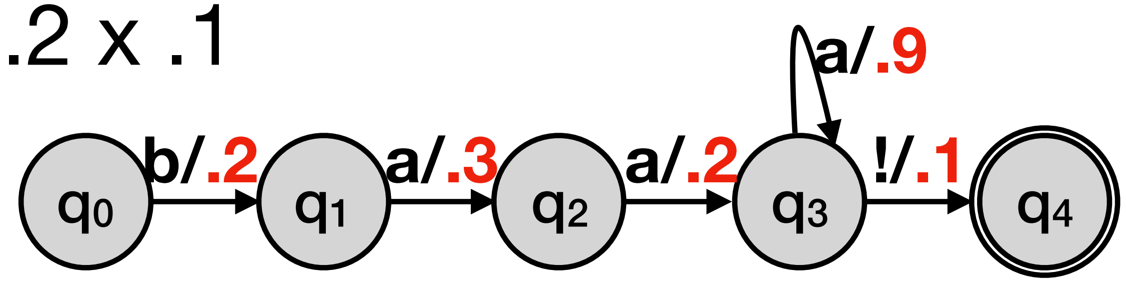
- Example: Tropical
  - K: All Reals $\mathbb{R}$, $\pm\infty$
  - ⊕ : min
  - ⊗ : +
  - $\underline{0}$: ∞
  - $\underline{1}$: 0

- Example: Boolean
  - K: {0,1}
  - ⊕ : OR
  - ⊗ : AND
  - $\underline{0}$: 0
  - $\underline{1}$: 1

from Abstract Algebra: Semiring is like a ring but does not require additive inverse

Weighted FSA/FST

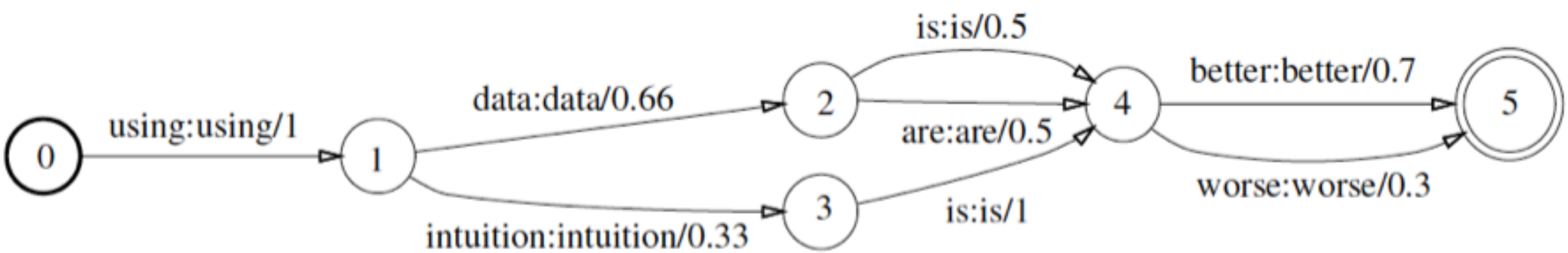
- We can use a general algorithm but change semirings on weights
- baa! in probability semiring: .2 x .3 x .2 x .1
- baa! in tropical semiring: .2+.3+.2+.1



SEMIRING	SET	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
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

$$x \oplus_{\log} y = -\log(e^{-x} + e^{-y})$$

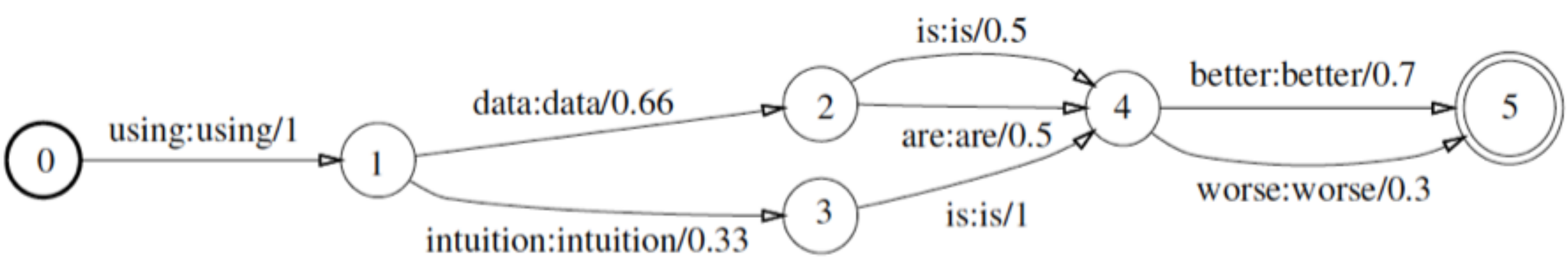
Weighted FSA/FST



- We define two operators in a weighted graph
  - the weight of a path uses ⊗ on arcs:
    - weight of path “using data is” = 1 ⊗ 0.66 ⊗ 0.5
  - the weight of a vertex with multiple incoming paths uses ⊕:
    - weight at state 4 = weight(using data is) ⊕ weight(“using data are”) ⊕ weight(“using intution is”)

Algorithms on WFST

- Single-source shortest-path
  - Question: Which semiring is used in Viterbi Algo in HMM? How about Forward Algo?
- N shortest paths
- etc



Optimizing WFST

Determinization:

- Eliminates ambiguity in input paths, so each state won't have outgoing arc with same labels
- May improve efficiency of shortest-path

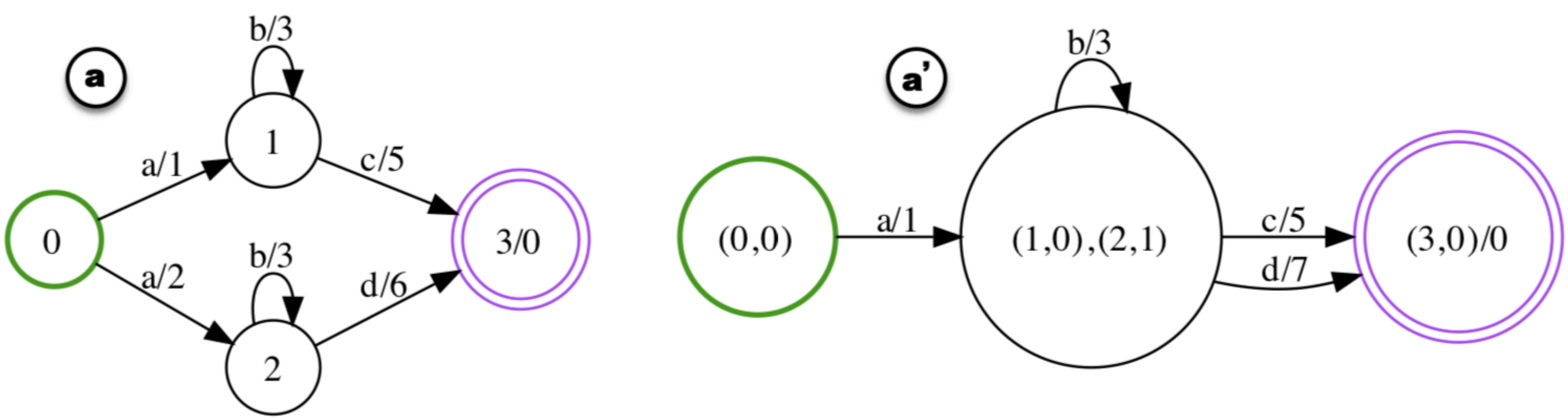
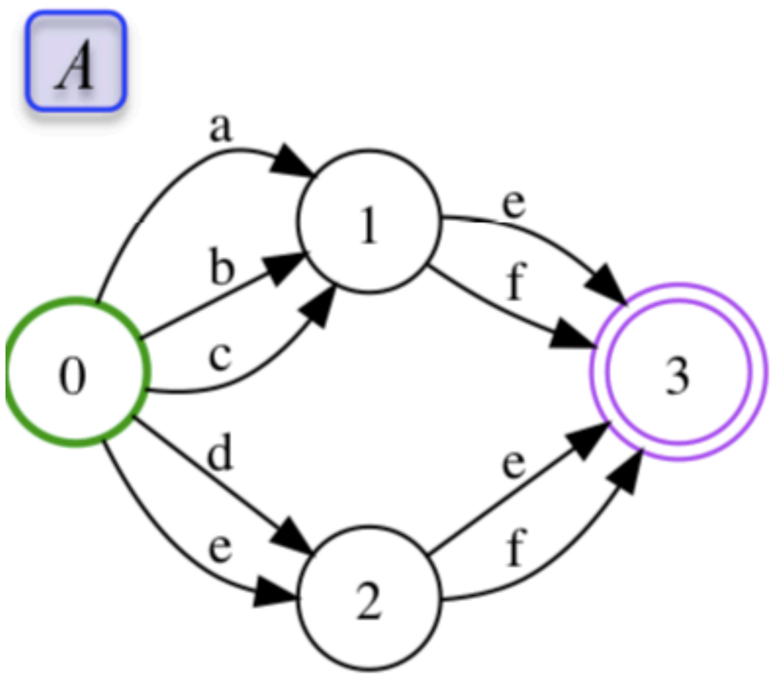
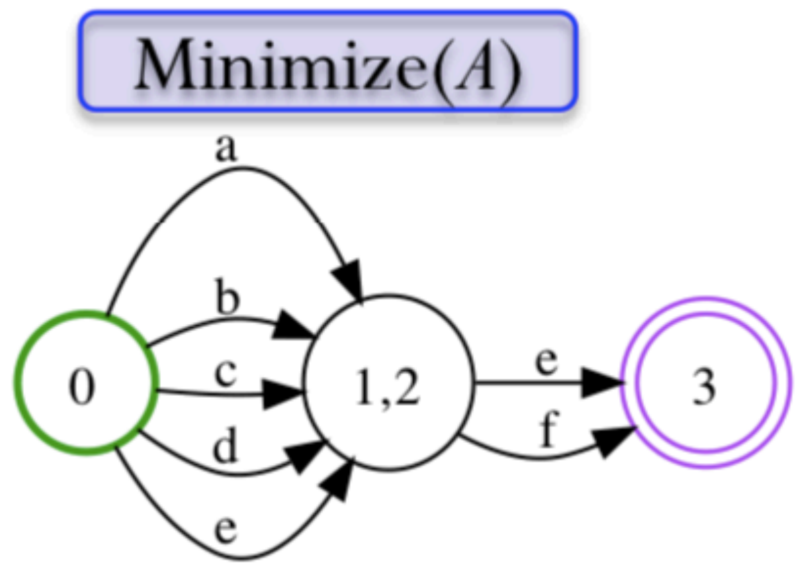


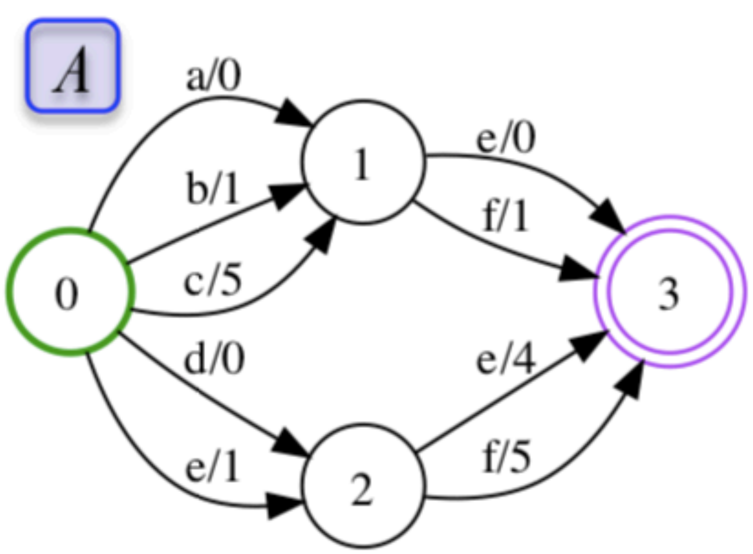
Figure from: J. Novak. http://www.gavo.t.u-tokyo.ac.jp/~novakj/wfst-algorithms.pdf

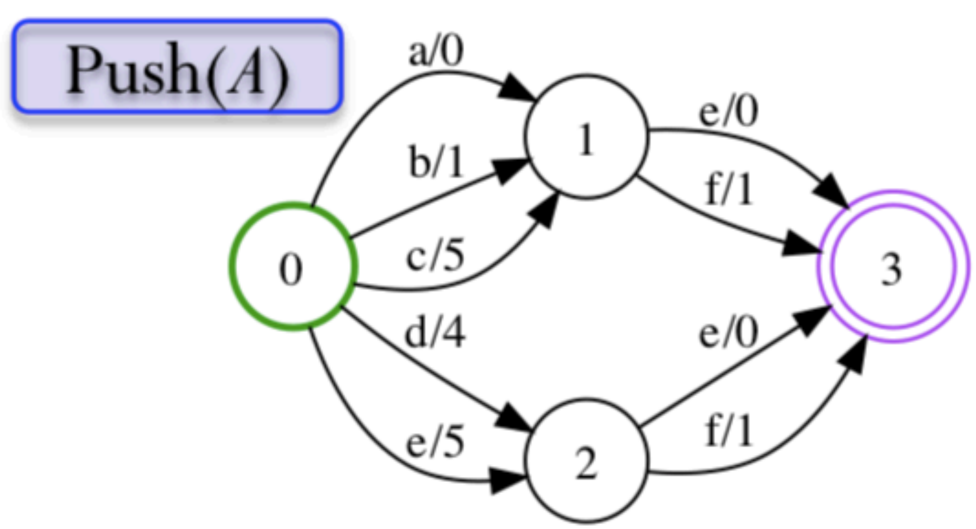
Optimizing WFST





Minimize #states





Push weights to beginning if possible

Figure from: J. Novak. http://www.gavo.t.u-tokyo.ac.jp/~novakj/wfst-algorithms.pdf

Summary

1. Finite-State Automaton (FSA)

2. Finite-State Transducer (FST)

3. Applications in Morphology

4. Semiring