

CSCI 5832

Natural Language Processing

Lecture 3
Jim Martin

1/23/07

CSCI 5832 Spring 2006

1

Today 1/23

- **Review FSA**
 - **Determinism and Non-Determinism**
- **Combining FSA**
- **English Morphology**

1/23/07

CSCI 5832 Spring 2006

2

Review

- Regular expressions are just a compact textual representation of FSAs
- Recognition is the process of determining if a string/input is in the language defined by some machine.
 - Recognition is straightforward with deterministic machines.

1/23/07

CSCI 5832 Spring 2006

3

D-Recognize

```
function D-RECOGNIZE(tape, machine) returns accept or reject
  index ← Beginning of tape
  current-state ← Initial state of machine
  loop
    if End of input has been reached then
      if current-state is an accept state then
        return accept
      else
        return reject
    elseif transition-table[current-state, tape[index]] is empty then
      return reject
    else
      current-state ← transition-table[current-state, tape[index]]
      index ← index + 1
  end
```

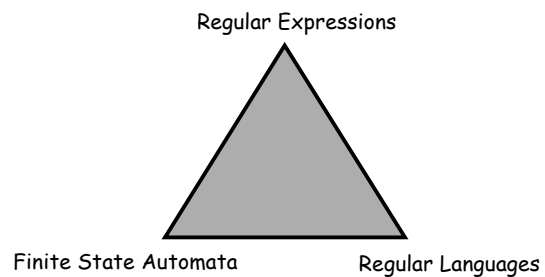
1/23/07

CSCI 5832 Spring 2006

4

Three Views

- **Three equivalent formal ways to look at what we're up to (not including tables)**



1/23/07

CSCI 5832 Spring 2006

5

Regular Languages

- **More on these in a couple of weeks**

$S \rightarrow b a a A$

$A \rightarrow a A$

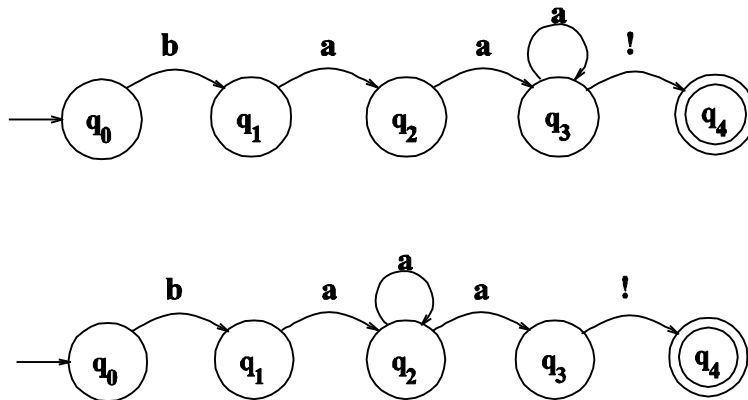
$A \rightarrow !$

1/23/07

CSCI 5832 Spring 2006

6

Non-Determinism



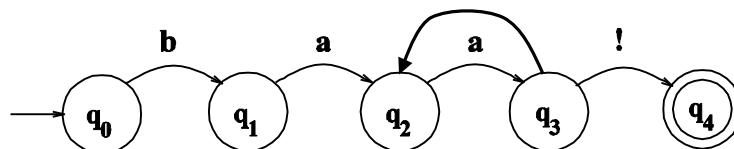
1/23/07

CSCI 5832 Spring 2006

7

Non-Determinism cont.

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



1/23/07

CSCI 5832 Spring 2006

8

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**
- **It also means that one way to do recognition with a non-deterministic machine is to turn it into a deterministic one.**

1/23/07

CSCI 5832 Spring 2006

9

Non-Deterministic Recognition

- **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.**

1/23/07

CSCI 5832 Spring 2006

10

Non-Deterministic Recognition

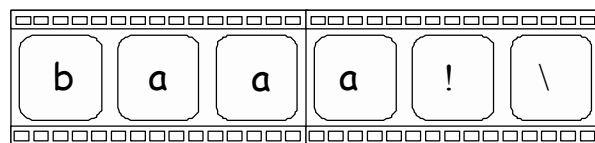
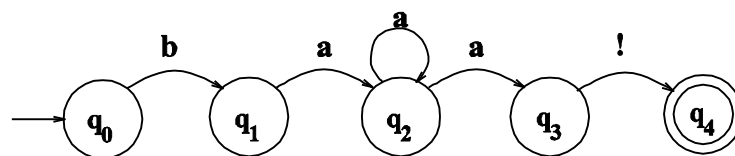
- So success in a non-deterministic recognition occurs when a path is found through the machine that ends in an accept.
- Failure occurs when all of the possible paths lead to failure.

1/23/07

CSCI 5832 Spring 2006

11

Example



q₀ q₁ q₂ q₂ q₃ q₄

1/23/07

CSCI 5832 Spring 2006

12

1 Example



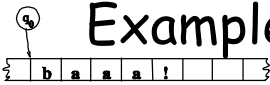
The diagram shows a horizontal array of eight cells. The first four cells contain the characters 'b', 'a', 'a', and 'a', followed by an exclamation mark '!' in the fifth cell. The remaining three cells are empty. A curly brace under the first four cells indicates the current string. A pointer 'q' is positioned above the first cell, with an arrow pointing to the character 'b'.

1/23/07

CSCI 5832 Spring 2006

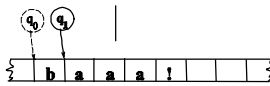
13

1 Example



The diagram shows a horizontal array of eight cells. The first four cells contain the characters 'b', 'a', 'a', and 'a', followed by an exclamation mark '!' in the fifth cell. The remaining three cells are empty. A curly brace under the first four cells indicates the current string. A pointer 'q' is positioned above the first cell, with an arrow pointing to the character 'b'.

2

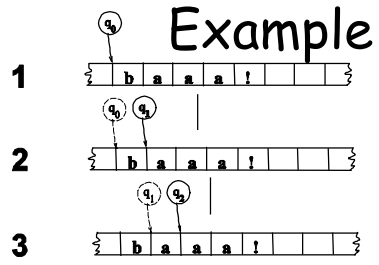


The diagram shows a horizontal array of eight cells. The first four cells contain the characters 'b', 'a', 'a', and 'a', followed by an exclamation mark '!' in the fifth cell. The remaining three cells are empty. A curly brace under the first four cells indicates the current string. Two pointers, 'q0' and 'q1', are positioned above the first and second cells respectively, with arrows pointing to the characters 'b' and 'a'.

1/23/07

CSCI 5832 Spring 2006

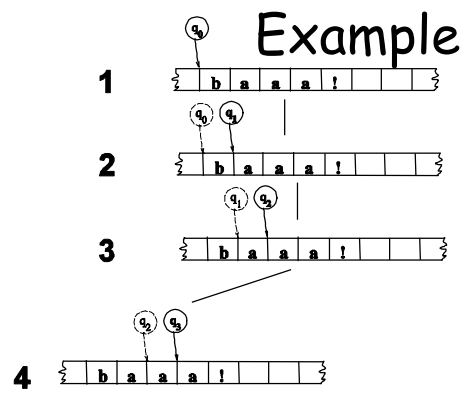
14



1/23/07

CSCI 5832 Spring 2006

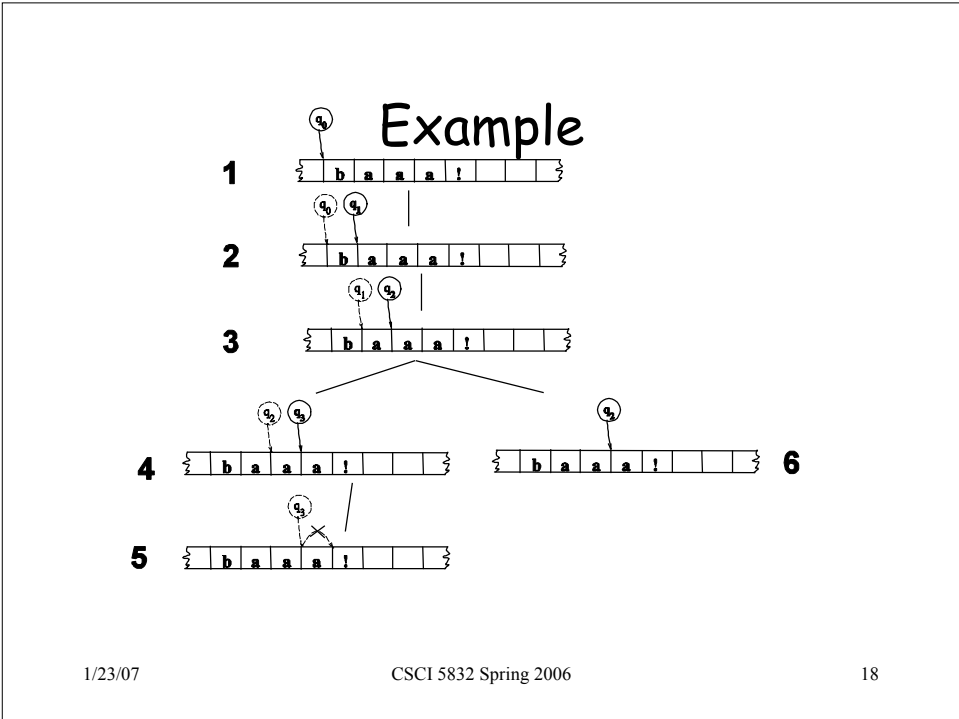
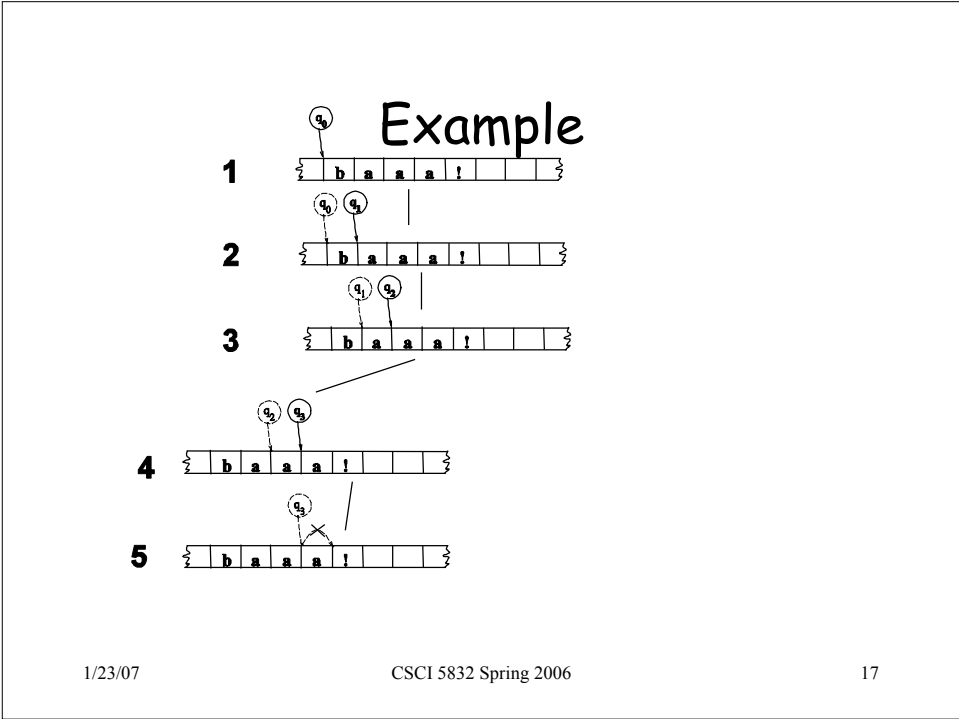
15

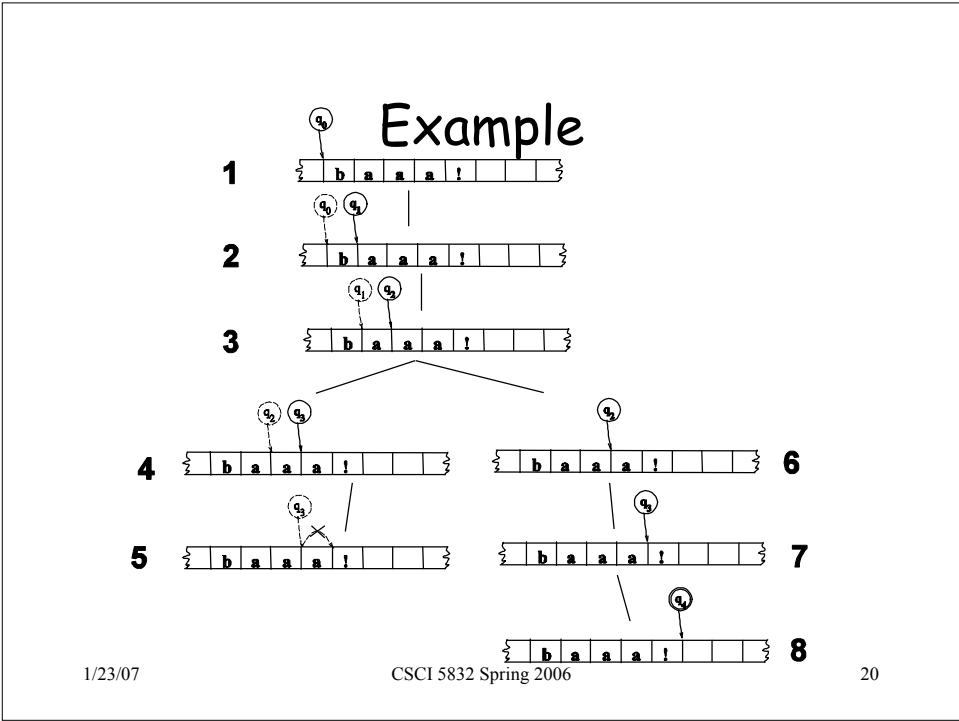
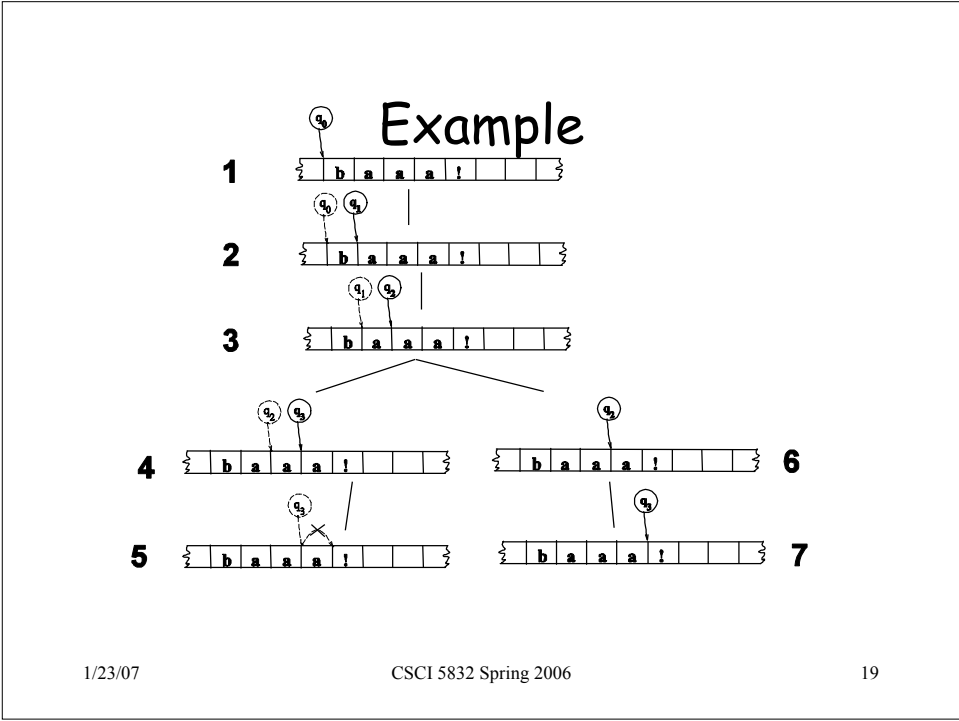


1/23/07

CSCI 5832 Spring 2006

16





Key Points

- States in the search space are pairings of tape positions and states in the machine.
- By keeping track of as yet unexplored states, a recognizer can systematically explore all the paths through the machine given an input.

1/23/07

CSCI 5832 Spring 2006

21

ND-Recognize

```
function ND-RECOGNIZE(tape, machine) returns accept or reject
  agenda ← {(Initial state of machine, beginning of tape)}
  current-search-state ← NEXT(agenda)
  loop
    if ACCEPT-STATE?(current-search-state) returns true then
      return accept
    else
      agenda ← agenda ∪ GENERATE-NEW-STATES(current-search-state)
    if agenda is empty then
      return reject
    else
      current-search-state ← NEXT(agenda)
  end
```

1/23/07

CSCI 5832 Spring 2006

22

Infinite Search

- **If you're not careful such searches can go into an infinite loop.**
- **How?**

1/23/07

CSCI 5832 Spring 2006

23

Why Bother?

- **Non-determinism doesn't get us more formal power and it causes headaches so why bother?**
 - **More natural (understandable) solutions**

1/23/07

CSCI 5832 Spring 2006

24

Compositional Machines

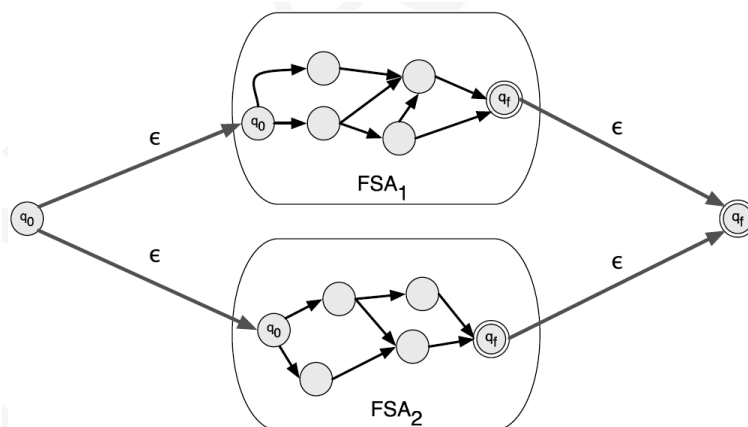
- Formal languages are just sets of strings
- Therefore, we can talk about various set operations (intersection, union, concatenation)
- This turns out to be a useful exercise

1/23/07

CSCI 5832 Spring 2006

25

Union

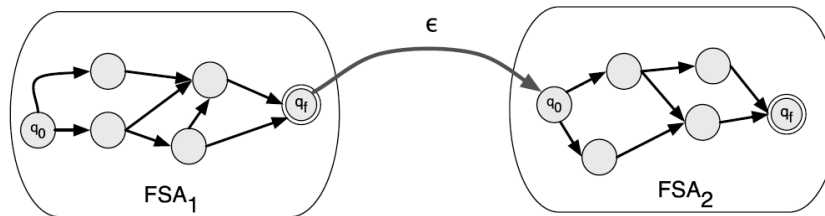


1/23/07

CSCI 5832 Spring 2006

26

Concatenation



1/23/07

CSCI 5832 Spring 2006

27

Negation

- **Construct a machine M2 to accept all strings not accepted by machine M1 and reject all the strings accepted by M1**
 - Invert all the accept and not accept states in M1
- **Does that work for non-deterministic machines?**

1/23/07

CSCI 5832 Spring 2006

28

Intersection

- **Accept a string that is in both of two specified languages**
- **An indirect construction...**
 - $A \wedge B = \sim(\sim A \text{ or } \sim B)$

1/23/07

CSCI 5832 Spring 2006

29

Motivation

- **Let's have a meeting on Thursday, Jan 26th.**
 - **Writing an FSA to recognize English date expressions is not terribly hard.**
 - **Except for the part about rejecting invalid dates.**
 - **Write two FSAs: one for the form of the dates, and one for the calendar arithmetic part**
 - **Intersect the two machines**

1/23/07

CSCI 5832 Spring 2006

30

Administration

- Homework questions?
- Anything else?

1/23/07

CSCI 5832 Spring 2006

31

Assignment 1

- **Strings are an easy and not very good way to represent texts**
- **Normally, we want lists of sentences that consist of lists of tokens, that ultimately may point to strings representing words (lexemes)**
- **Lists are central to Python and will make your life easy if you let them**

1/23/07

CSCI 5832 Spring 2006

32

Transition

- **Finite-state methods are particularly useful in dealing with a lexicon.**
- **Lots of devices, some with limited memory, need access to big lists of words.**
- **So we'll switch to talking about some facts about words and then come back to computational methods**

1/23/07

CSCI 5832 Spring 2006

33

English Morphology

- **Morphology is the study of the ways that words are built up from smaller meaningful units called morphemes**
- **We can usefully divide morphemes into two classes**
 - **Stems: The core meaning-bearing units**
 - **Affixes: Bits and pieces that adhere to stems to change their meanings and grammatical functions**

1/23/07

CSCI 5832 Spring 2006

34

English Morphology

- **We can also divide morphology up into two broad classes**
 - **Inflectional**
 - **Derivational**

1/23/07

CSCI 5832 Spring 2006

35

Word Classes

- **By word class, we have in mind familiar notions like noun and verb**
- **We'll go into the gory details in Ch 5**
- **Right now we're concerned with word classes because the way that stems and affixes combine is based to a large degree on the word class of the stem**

1/23/07

CSCI 5832 Spring 2006

36

Inflectional Morphology

- **Inflectional morphology concerns the combination of stems and affixes where the resulting word**
 - Has the same word class as the original
 - Serves a grammatical/semantic purpose that is
 - Different from the original
 - But nevertheless transparently related to the original

1/23/07

CSCI 5832 Spring 2006

37

Nouns and Verbs (English)

- **Nouns are simple**
 - Markers for plural and possessive
- **Verbs are only slightly more complex**
 - Markers appropriate to the tense of the verb

1/23/07

CSCI 5832 Spring 2006

38

Regulars and Irregulars

- **Ok so it gets a little complicated by the fact that some words misbehave (refuse to follow the rules)**
 - *Mouse/mice, goose/geese, ox/oxen*
 - *Go/went, fly/flew*
- **The terms regular and irregular will be used to refer to words that follow the rules and those that don't.**

1/23/07

CSCI 5832 Spring 2006

39

Regular and Irregular Verbs

- **Regulars...**
 - *Walk, walks, walking, walked, walked*
- **Irregulars**
 - *Eat, eats, eating, ate, eaten*
 - *Catch, catches, catching, caught, caught*
 - *Cut, cuts, cutting, cut, cut*

1/23/07

CSCI 5832 Spring 2006

40

Derivational Morphology

- Derivational morphology is the messy stuff that no one ever taught you.
 - Quasi-systematicity
 - Irregular meaning change
 - Changes of word class

1/23/07

CSCI 5832 Spring 2006

41

Derivational Examples

- Verb/Adj to Noun

-ation	computerize	computerization
-ee	appoint	appointee
-er	kill	killer
-ness	fuzzy	fuzziness

1/23/07

CSCI 5832 Spring 2006

42

Derivational Examples

- Noun/Verb to Adj

-al	Computation	Computational
-able	Embrace	Embraceable
-less	Clue	Clueless

1/23/07

CSCI 5832 Spring 2006

43

Compute

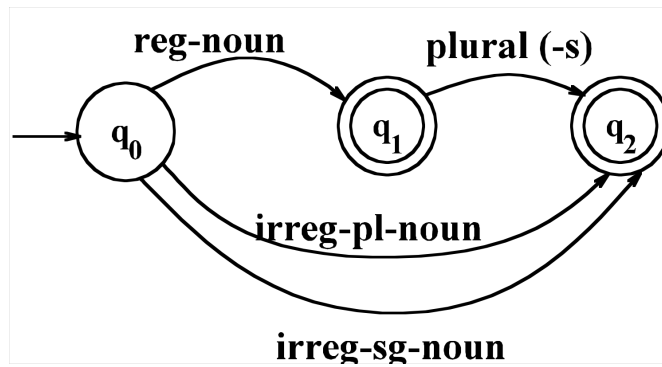
- Many paths are possible...
- Start with compute
 - Computer -> computerize -> computerization
 - Computation -> computational
 - Computer -> computerize -> computerizable
 - Compute -> computee

1/23/07

CSCI 5832 Spring 2006

44

Simple Rules

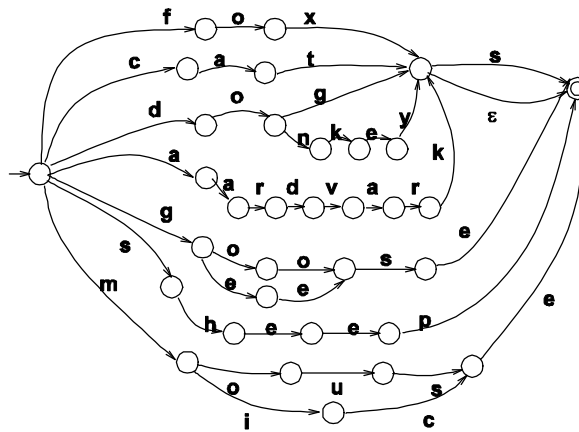


1/23/07

CSCI 5832 Spring 2006

45

Adding in the Words

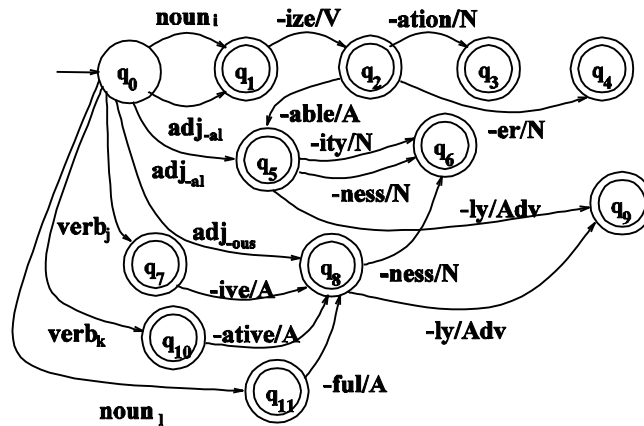


1/23/07

CSCI 5832 Spring 2006

46

Derivational Rules



1/23/07

CSCI 5832 Spring 2006

47

Parsing/Generation vs. Recognition

- **Recognition is usually not quite what we need.**
 - Usually if we find some string in the language we need to find the structure in it (parsing)
 - Or we have some structure and we want to produce a surface form (production/generation)
- **Example**
 - From "cats" to "cat +N +PL"

1/23/07

CSCI 5832 Spring 2006

48

Finite State Transducers

- **The simple story**
 - Add another tape
 - Add extra symbols to the transitions

 - On one tape we read "cats", on the other we write "cat +N +PL"

1/23/07

CSCI 5832 Spring 2006

49

Next Time

- **On to Chapter 3**

1/23/07

CSCI 5832 Spring 2006

50

FSAs and the Lexicon

- **First we'll capture the morphotactics**
 - The rules governing the ordering of affixes in a language.
- **Then we'll add in the actual words**