CSCI 5832 Natural Language Processing

Lecture 3
Jim Martin

Today 1/22

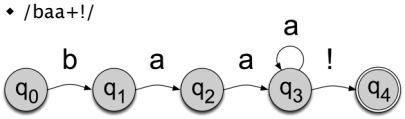
- Regexs, FSAs and languages
  - Determinism and Non-Determinism
- Combining FSAs
- English Morphology

Finite State Automata

- Regular expressions can be viewed as a textual way of specifying the structure of finite-state automata.
- FSAs and their probabilistic relatives are at the core of what we'll be doing all semester.
- They also conveniently (?) correspond closely to what linguists say we need for morphology and parts of syntax.
  - *Coincidence*?

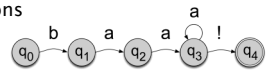
FSAs as Graphs

- Let's start with the sheep language from the text
  - /baa+!/



Sheep FSA

- We can say the following things about this machine
  - It has 5 states
  - b, a, and ! are in its alphabet
  - q0 is the start state
  - q4 is an accept state
  - It has 5 transitions



More Formally

- You can specify an FSA by enumerating the following things.
  - The set of states: Q
  - A finite alphabet: Σ
  - A start state
  - A set of accept/final states
  - A transition function that maps QxΣ to Q

Generative Formalisms

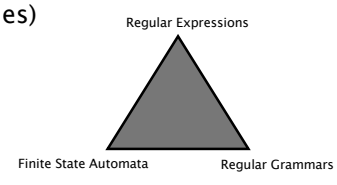
- Formal Languages are sets of strings composed of symbols from a finite set of symbols.
- Finite-state automata define formal languages (without having to enumerate all the strings in the language)
- The term Generative is based on the view that you can run the machine as a generator to get strings from the language.

Generative Formalisms

- FSAs can be viewed from two perspectives:
  - Acceptors that can tell you if a string is in the language
  - Generators to produce all and only the strings in the language

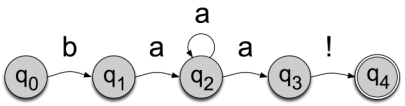
Three Views

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



But note

- There are other machines that correspond to this same language

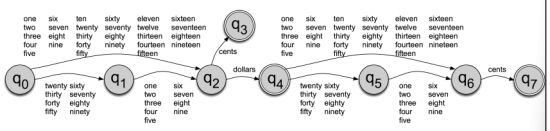


- More on this one later

About Alphabets

- Don't take that word to narrowly; it just means we need a finite set of symbols in the input.
- These symbols can and will stand for bigger objects that can have internal structure.

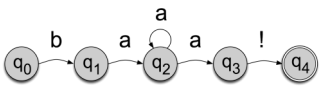
Dollars and Cents



QxΣ → Q

- The guts of FSAs can ultimately be represented as tables

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

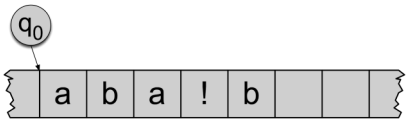


Recognition

- Recognition is the process of determining if a string should be accepted by a machine
- Or… it's the process of determining if a string is in the language defined by the machine
- Or… it's the process of determining if a regular expression matches a string
- Those all amount to the same thing in the end

Recognition

- Traditionally, (Turing's idea) this recognition process is depicted with a tape.



Recognition

- Simply a process of starting in the start state
- Examining the current input
- Consulting the table
- Going to a new state and updating the tape pointer.
- Until you run out of tape.

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
    elsif transition-table[current-state,tape[index]] is empty then
      return reject
    else
      current-state ← transition-table[current-state,tape[index]]
      index ← index + 1
  end
```


Key Points

- Deterministic means that at each point in processing there is always one unique thing to do (there are no choices to be made).
- D-recognize is a simple table-driven interpreter
- The algorithm is universal for all unambiguous regular languages.
  - To change the machine, you just change the table.

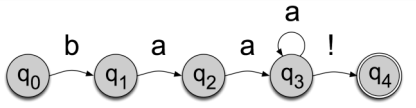
Key Points

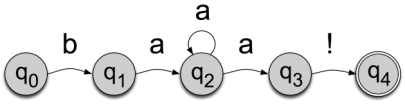
- Crudely therefore… matching strings with regular expressions (ala Perl, grep, etc.) is a matter of
  - translating the regular expression into a machine (a table) and
  - passing the table to an interpreter

Recognition as Search

- You can view this algorithm as a trivial kind of state-space search.
- States are pairings of tape positions and state numbers.
- Operators are compiled into the table
- Goal state is a pairing with the end of tape position and a final accept state
- Its trivial because?

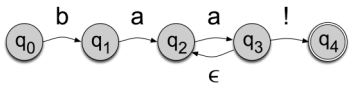
Non-Determinism





Non-Determinism

- 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 and can not accept

ND Recognition

- Two basic approaches (used in all major implementations of Regular Expressions)
  1. Either take a ND machine and convert it to a D machine and then do recognition with that.
  2. Or explicitly manage the process of recognition as a state-space search (leaving the machine as is).

Implementations

Program	(Original) Author	Version	Regex Engine
awk	Aho, Weinberger, Kernighan	*generic*	DFA
new *awk*	Brian Kernighan	*generic*	DFA
GNU awk	Arnold Robbins	*recent*	Mostly DFA, some NFA
MKS awk	Mortice Kern Systems		POSIX NFA
mawk	Mike Brennan	*all*	POSIX NFA
egrep	Alfred Aho	*generic*	DFA
MKS egrep	Mortice Kern Systems		POSIX NFA
GNU Emacs	Richard Stallman	*all*	Trad. NFA (POSIX NFA available)
Expect	Don Libes	*all*	Traditional NFA
exptr	Dick Haight	*generic*	Traditional NFA
grep	Ken Thompson	*generic*	Traditional NFA
GNU *grep*	Mike Haertel	Version 2.0	Mostly DFA, but some NFA
GNU find	GNU		Traditional NFA
lex	Mike Lesk	*generic*	DFA
flex	Vern Paxson	*all*	DFA
lcc	Mortice Kern Systems		POSIX NFA
msve	Eric Schienbrood	*generic*	Traditional NFA
bre	Mark Nudelman		Variable (usually Trad. NFA)
Perl	Larry Wall	*all*	Traditional NFA
Python	Guido van Rossum	*all*	Traditional NFA
sed	Lee McMahon	*generic*	Traditional NFA
Tcl	John Ousterhout	*all*	Traditional NFA
vi	Bill Joy	*generic*	Traditional NFA

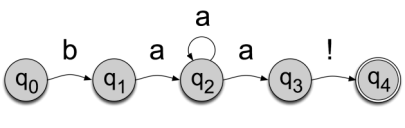
Non-Deterministic Recognition: Search

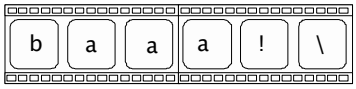
- In a ND FSA there exists at least one path through the machine for a string that is in the language defined by the machine.
- But not all paths directed through the machine for an accept string lead to an accept state.
- No paths through the machine lead to an accept state for a string not in the language.

Non-Deterministic Recognition

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

Example

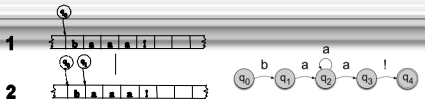




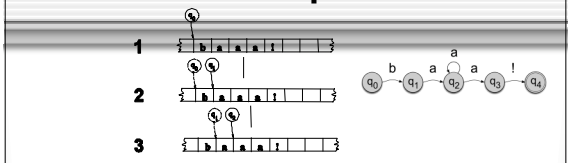
q_0 q_1 q_2 q_2 q_3 q_4

Example

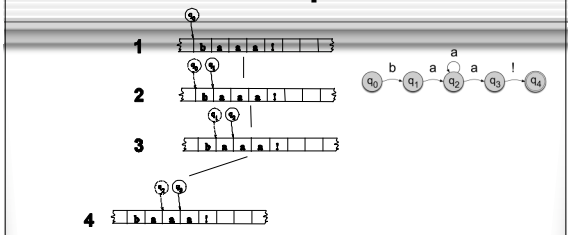
Example



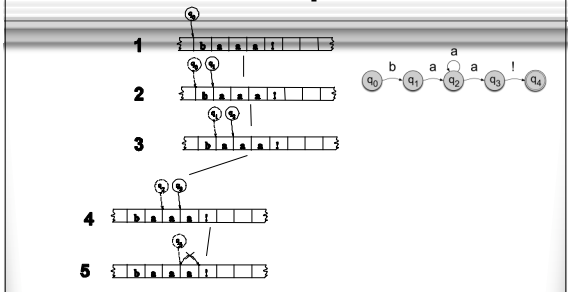
Example



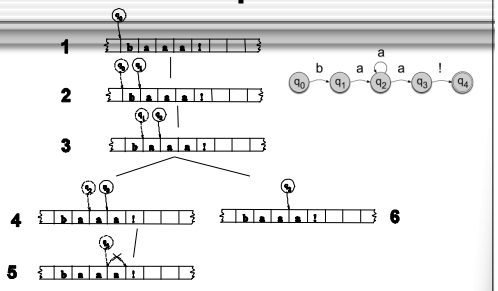
Example



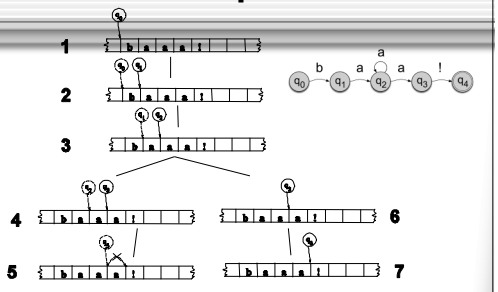
Example



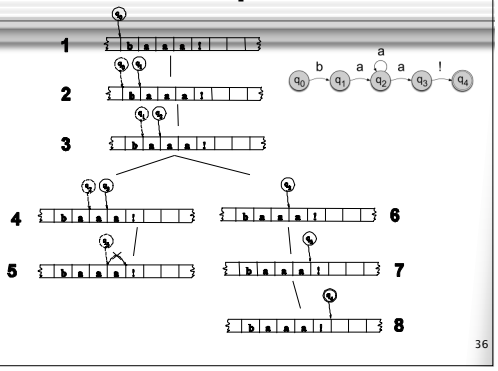
Example



Example



Example



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.

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
```


Infinite Search

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

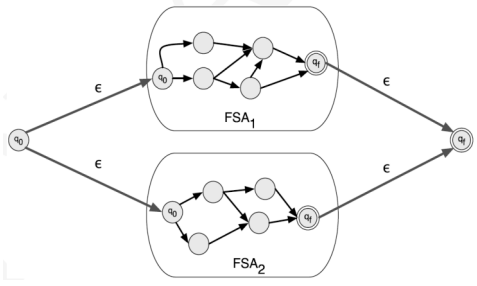
Why Bother?

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

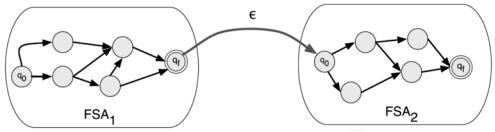
Compositional Machines

- 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



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?

Intersection

- Accept a string that is in both of two specified languages
- An indirect construction…
  - A^B = ~(~A or ~B)

Motivation

- Consider the expression

  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

Next Time

- Finish Chapter 3