Finite Automata

Machines for Regular Languages

Monday, October 1, 2007
Reading: Stoughton 3.3 – 3.5

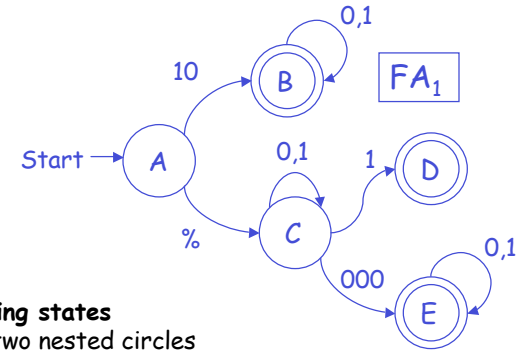
CS235 Languages and Automata

Department of Computer Science
Wellesley College

A Simple Finite Automaton (FA)

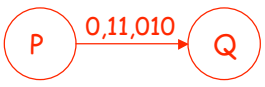
A finite automaton (FA) is a directed multigraph each of whose nodes is a **state** labeled with a symbol and each of whose edges is a **transition** labeled with a string.

There is a unique **start state** (A in this example).

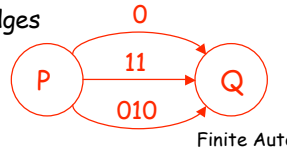


Any of the states may be **accepting states** (a.k.a. **final states**), denoted by two nested circles (B, D, E in this example).

An edge with comma-separated strings …



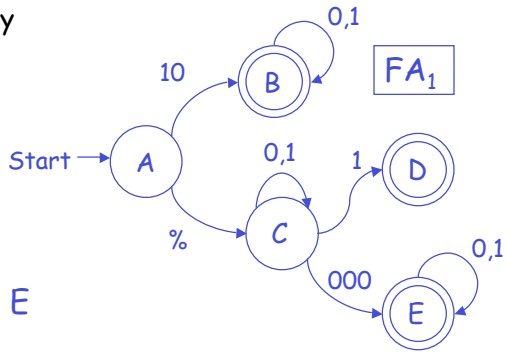
… is an abbreviation for multiple edges



Labeled Paths

A **labeled path** in an FA is any path in which each edge is labeled by the string on the transition taken.

Here are some labeled paths in our example:



$$A \overset{\%}{\Rightarrow} C \overset{1}{\Rightarrow} C \overset{0}{\Rightarrow} C \overset{000}{\Rightarrow} E \overset{1}{\Rightarrow} E$$

$$C \overset{0}{\Rightarrow} C \overset{0}{\Rightarrow} C \overset{1}{\Rightarrow} C$$

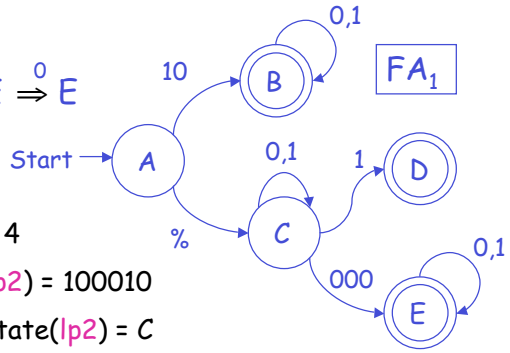
$$C$$

Operations on Labeled Paths

Let lp1 = $A \overset{\%}{\Rightarrow} C \overset{1}{\Rightarrow} C \overset{0}{\Rightarrow} C$

and lp2 = $C \overset{1}{\Rightarrow} C \overset{000}{\Rightarrow} E \overset{1}{\Rightarrow} E \overset{0}{\Rightarrow} E$

and lp3 = $C \overset{01}{\Rightarrow} A$



|lp1| = length (lp1) = 3; |lp2| = 4

label(lp1) = 10; label(lp2) = 100010

startState(lp1) = A; startState(lp2) = C

endState(lp1) = C; endState(lp2) = E

Lp4 = join(lp1,lp2) = $A \overset{\%}{\Rightarrow} C \overset{1}{\Rightarrow} C \overset{0}{\Rightarrow} C \overset{1}{\Rightarrow} C \overset{000}{\Rightarrow} E \overset{1}{\Rightarrow} E \overset{0}{\Rightarrow} E$

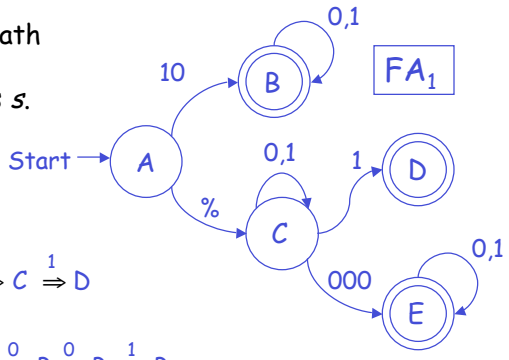
isValid(lp1,FA_1) = true; isValid(lp2, FA_1) = true; isValid(lp3, FA_1) = false

isValid(lp4, FA_1) = true; isValid(join(lp1,lp3), FA_1) = false

Accepted Strings

An FA **accepts** a string *s* if there is *some** valid labeled path from its start state to an accepting state whose label is *s*.

* The "some" means that FA are **nondeterministic**.



FA_1 accepts 011 via $A \overset{\%}{\Rightarrow} C \overset{0}{\Rightarrow} C \overset{1}{\Rightarrow} C \overset{1}{\Rightarrow} D$

FA_1 does not accept 110

FA_1 accepts 10001 via: (1) $A \overset{10}{\Rightarrow} B \overset{0}{\Rightarrow} B \overset{0}{\Rightarrow} B \overset{1}{\Rightarrow} B$

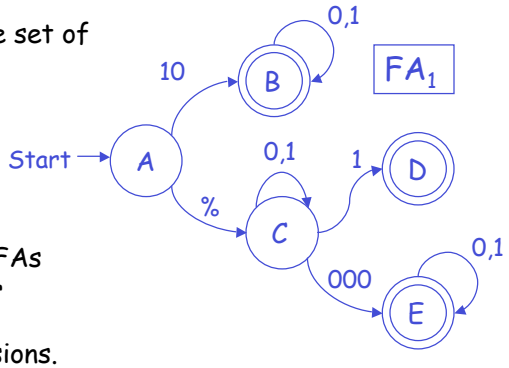
(2) $A \overset{\%}{\Rightarrow} C \overset{1}{\Rightarrow} C \overset{0}{\Rightarrow} C \overset{0}{\Rightarrow} C \overset{0}{\Rightarrow} C \overset{1}{\Rightarrow} D$

(3) $A \overset{\%}{\Rightarrow} C \overset{1}{\Rightarrow} C \overset{000}{\Rightarrow} E \overset{1}{\Rightarrow} E$

but not via $A \overset{\%}{\Rightarrow} C \overset{1}{\Rightarrow} C \overset{0}{\Rightarrow} C \overset{0}{\Rightarrow} C \overset{0}{\Rightarrow} C \overset{1}{\Rightarrow} C$

The Language of an FA

The **language** of an FA is the set of all strings it accepts.



Next time, we will see that FAs describe exactly the regular languages = those languages specified by regular expressions.

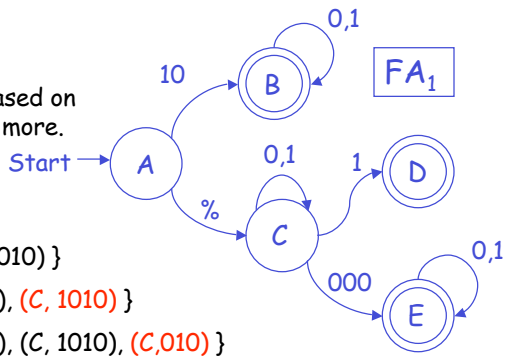
What is language(FA1) in English?

Expressed as a regular expression?

An Acceptance Algorithm by Example

Is 1010 accepted by FA_1?

1. Start with {(A, 1010)}
2. Add new state/string pairs based on transitions until there are no more.



{ (A, 1010) }

→ { (A, 1010), (B,10), (C, 1010) }

→ { (A, 1010), (B,10), (B,0), (C, 1010) }

→ { (A, 1010), (B,10), (B,0), (B,%), (C, 1010) }

→ { (A, 1010), (B,10), (B,0), (B,%), (C, 1010), (C,010) }

→ { (A, 1010), (B,10), (B,0), (B,%), (C, 1010), (C,010), (C,10) }

→ { (A, 1010), (B,10), (B,0), (B,%), (C, 1010), (C,010), (C,10), (C,0) }

→ { (A, 1010), (B,10), (B,0), (B,%), (C, 1010), (C,010), (C,10), (C,0), (C,%) }

Accept if (Q,%) is in set for some accepting state Q; otherwise **reject**.

Can terminate early in this case.

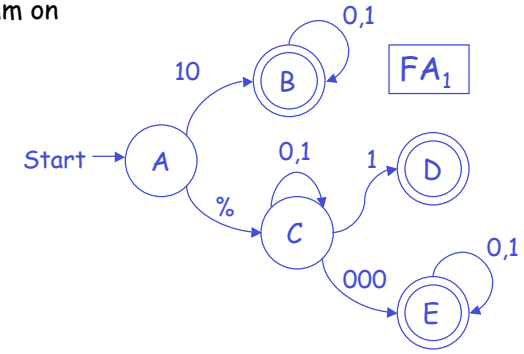
More Acceptance Examples

Run the acceptance algorithm on the following examples:

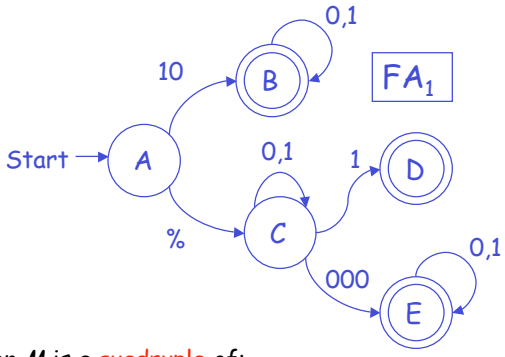
110

101

0000



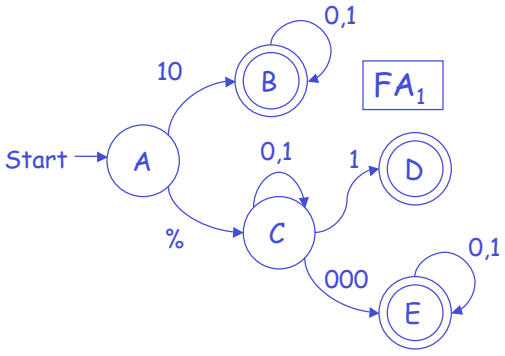
Another Formalization of Finite Automata



A finite automaton M is a quadruple of:

1. **STtates** = a finite set Q_M of symbols	({A,B,C,D,E} ,
2. A **start state** = an element of Q_M	A ,
3. **Accepting states** = a subset $\mathbf{A_M} \subseteq Q_M$	{B,D,E} ,
4. **Transitions** = a set of triples (q,x,r) where $q, r \in Q_M$ and $x \in$ **Str**.	{(A,10,B), (A,%,C), (B,0,B), (B,1,B), (C,0,C), (C,1,C), (C,1,D), (C,000,E) (E,0,E), (E,1,E)})

Our Sample Automaton in Forlan Syntax




```
{states}
A, B, C, D, E
{start state}
A
{accepting states}
B, D, E
{transitions}
A, 10 -> B; A, % -> C; B, 0 -> B; B, 1 -> B;
C, 0 -> C; C, 1 -> C | D; C, 000 -> E;
E, 0 -> E; E, 1 -> E;
```

Designing Finite Automata

Design (1) FAs and (2) regexps for each of the following languages in {0,1}*:

1. All strings containing one occurrence of 111 preceded only by an odd number of 0s and followed only by an even number of 0s.
2. All strings that contain any number of occurrences of 111, each of which is separated from others by an even number of 0s.
3. All strings that contain any number of occurrences of 111, each of which is separated from others by an odd number of 0s.
4. All strings that contain one occurrence of 111 and all of whose other characters are an odd number of 0s.

Isomorphisms Between FAs

Two FAs are isomorphic if a **state renaming** (a bijection on symbols) makes them the same. E.g.,

SR = { (P, A), (R, B), (A, C), (Q, D), (D, E)}