COMS3261: Computer Science Theory

Fall 2013

Mihalis Yannakakis

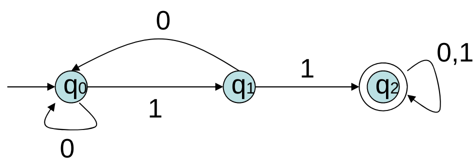
Lecture 2, 9/9/13

Definition of (Deterministic) Finite Automaton

- $A = (Q, \Sigma, \delta, q_0, F)$
- Q = finite set of states
- Σ= finite (input) alphabet
- δ = transition function: $\delta : Q \times \Sigma \rightarrow Q$
  i.e., for each q in Q, a in Σ, $\delta(q,a) \in Q$
  (the function is completely and uniquely defined for all input pairs (q,a) : *deterministic* FA)
- q_0 = start (or initial) state
- $F \subseteq Q$ is the set of accepting (or final) states

Example

- Transition Diagram representation



Transition table representation

- Rows correspond to states, columns to input symbols, entry for q,a is $\delta(q,a)$, start state marked with $\rightarrow$, accepting states marked with *

Symbols

States		0	1
	$\rightarrow q_0$	q_0	q_1
States	q_1	q_0	q_2
	$* q_2$	q_2	q_2

Processing of input by FA

- Given input string $x = a_1 a_2 \ldots a_n$, the DFA starts in state q_0, reads a_1 and moves to state $\delta(q_0, a_1)$ = say q_1, then reads a_2 and moves to state $\delta(q_1, a_2)$ = say q_2, etc. , i.e, the DFA goes through a sequence of states $q_1\, q_2\, q_3 \ldots q_n$ such that $\delta(q_{i-1}, a_i) = q_i$, for each $i=1,\ldots,n$.
- The input is accepted by the automaton iff the last state q_n is in F, and otherwise it is rejected.
- The language of the automaton A, denoted L(A), is the set of all input strings that are accepted by A.

- Regular languages = languages that are accepted (recognized) by some Finite Automaton

Accepting paths

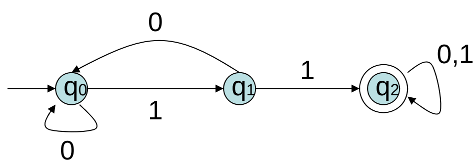
- For every node q and every label $a \in \Sigma$, there is a unique outgoing edge from q labeled a (because δ is a function)
- Computation (or run) of A on input $x=a_1\ldots a_n$:

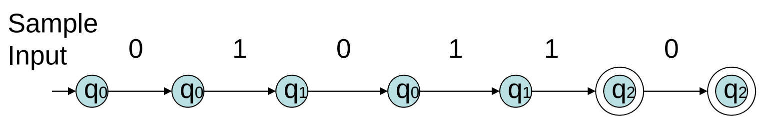
  the unique path that starts at the start state and has the sequence of labels $x = a_1\ldots a_n$ on the edges
- the sequence of nodes on the path = sequence of states of the DFA on input x
- Accepting computation (path) if ends in state in F

  $\Leftrightarrow$ Input string x accepted by A

- Language L(A) of automaton = set of labels of all accepting paths (i.e. paths from start state to states in F)
- L(A) = { x | there is a path π from the start state q_0 to a state in F whose label is x }

Example





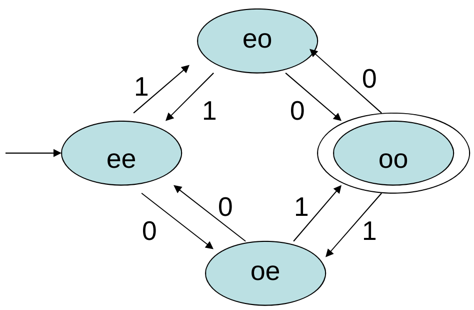
Language = set of strings that contain two consecutive 1's

Design of Finite Automata

- Determine the input symbols/actions
- Determine the states: After having seen part of the input, what do we need to remember about the past in order to make correct decisions in the future?
- Ascribe meaning to the states

Examples of FA

- FA that accepts all binary strings with an odd # of 0's and an odd # of 1's.
- $\Sigma=\{0,1\}$, Q={ee,eo,oe,oo}, q_0 = ee, F ={oo}
- State keeps track of parity of # of 0's and parity of # of 1's seen so far



Proving formally correctness of a FA

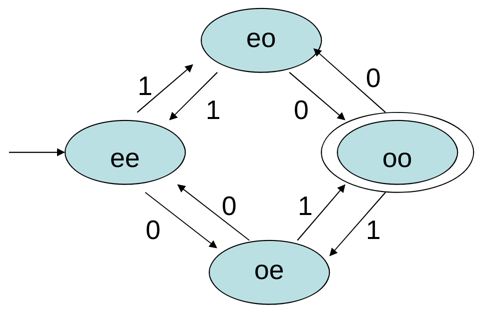
- Two directions to prove that L(A) = desired language L
- $L(A) \subseteq L$: for every input string x, if x is accepted by A then x is in L
- $L \subseteq L(A)$: for every input string x, $x \in L \Rightarrow x \in L(A)$

Can often do both directions at same time (if and only if)

Proof Method: Induction on length of x,

- but strengthen the claim (the induction hypothesis) to classify the strings according to the state to which they lead the FA

Example: odd # of 0's and 1's



Induction Hypothesis Claim: δ^(q_0,x) = ij where
i =e if x has an even number of 0's, and i=o otherwise
j= e if x has an even number of 1's, and i=o otherwise

Formal Proof

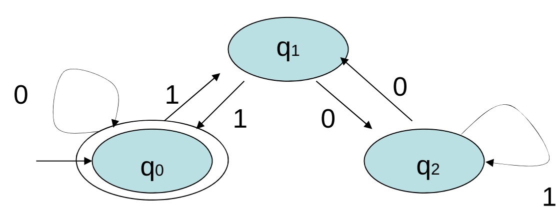
- Basis: |x| = 0, i.e. x= ε
  δ^(q_0,ε) = q_0 =ee, and ε has even # (0) of 0's, 1's, so ok.
- Induction Step: x=ya, where y$\in\Sigma$*, a$\in\Sigma$
  By induction hypothesis, claim holds for y.
  δ^(q_0,y)=i'j' where i'=parity of #0's in y, j'=parity of #1's in y
  δ^(q_0,x)= δ(i'j',a) = ij
  Case analysis.
  1. a=0: parity of 0's changes from y to x, parity of 1's same
     From transition diagram, first component of state changes, second stays same, so ok.
  2. a=1. Similar.

Practice Problems

- Construct DFA which accepts all strings whose length is divisible by 5
- Construct DFA which accepts all strings over {0,1,..,9} that represent a number divisible by 5
- Construct DFA which accepts all strings over {0,1,..,9} that represent a number divisible by 3
- Construct DFA which accepts all binary strings that represent in binary a number divisible by 3

  (e.g. $\varepsilon \leftrightarrow 0$, $0 \leftrightarrow 0$, $11 \leftrightarrow 3$, $011 \leftrightarrow 3$, $110 \leftrightarrow 6$, …)

Example: Binary numbers divisible by 3



Induction Hypothesis Claim:

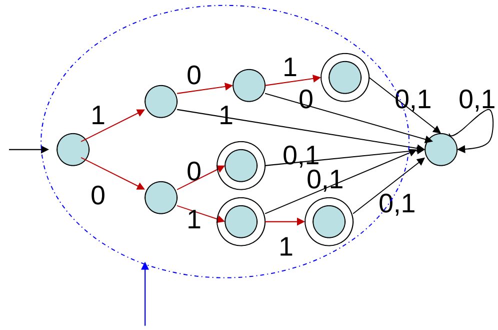
$\delta\hat{}(q_0,x) = q_0$ iff $x \bmod 3 = 0$

$\delta\hat{}(q_0,x) = q_1$ iff $x \bmod 3 = 1$

$\delta\hat{}(q_0,x) = q_2$ iff $x \bmod 3 = 2$

All finite sets are regular

- Example: L = { 00, 01, 011, 101 }

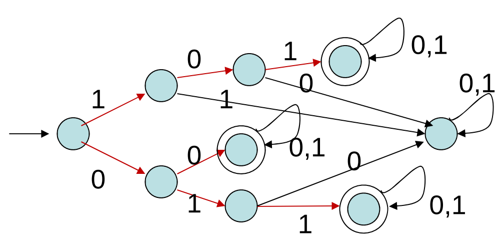


Automaton = Trie data structure for the set of strings +

"dead" absorbing state that rejects

FA Example: Prefixes

- L = set of all binary strings with prefix 00 or 011 or 101



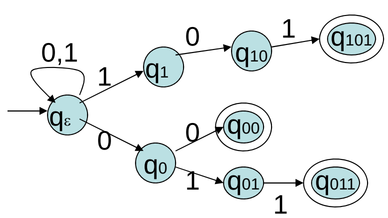
FA Example: Suffixes

- L = set of binary strings ending in one of 00, 011, 101
- Not so easy to design a DFA because we do not know when the string will finish and what we might see next

- If somebody would tell us (or we could guess) when the suffix is starting then easy:

  Wait in the initial state until it is time to start checking the suffix, then check if it is one of 00, 011, 101

Nondeterminism



Nondeterministic FA

Can have more than one transitions from a state on the same input symbol

Nondeterminism

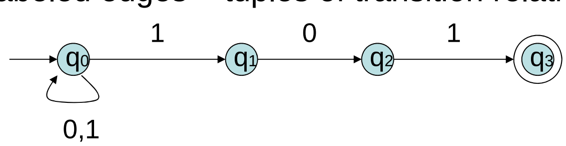
- Can think of it in two ways:

1. Run all possible computations in parallel. If any one computation succeeds (reaches an accepting state at the end of the input) then input is accepted.
2. Guess one computation path, assuming lucky: if there is any 'good' (accepting) path then we will guess such a path.

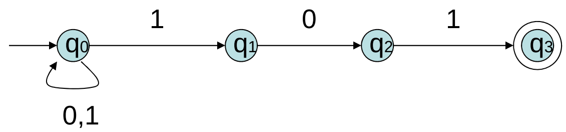
Powerful primitive.

Nondeterministic Finite Automaton

- A = (Q, Σ, δ, q_0, F)
- Only difference that transition function $\delta : Q \times \Sigma \rightarrow 2^Q = \mathcal{P}(Q)$
  i.e., for each q in Q, a in Σ, $\delta(q,a) \subseteq Q$ is a set of 0, 1 or more states
- Alternatively (equivalently), can represent it as a *transition relation* R ={(q,a,p) | p $\in$ δ(q,a) }
- Transition Diagram:
  Nodes = States
  Labeled edges = tuples of transition relation



Transition table representation

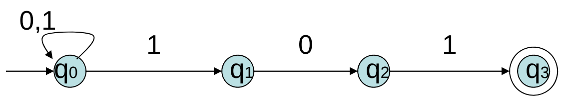


	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\emptyset$
q_2	$\emptyset$	$\{q_3\}$
$* q_3$	$\emptyset$	$\emptyset$

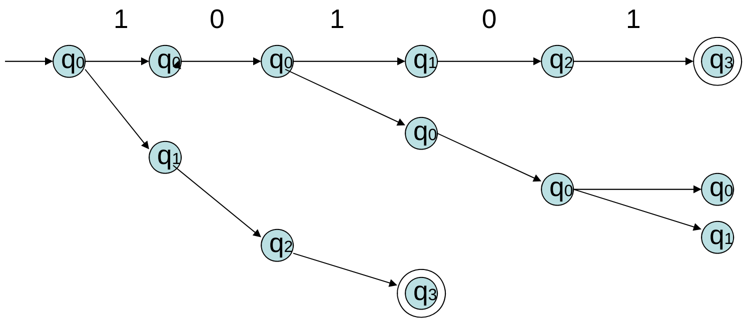
Computations, Acceptance

- Computation (run) of NFA on input $x = a_1 \ldots a_n$:
  sequence of states (not necessarily distinct) starting with initial state: $q_0\, q_1 \ldots q_n$ such that $q_i \in \delta(q_{i-1}, a_i)$, for all $i=1,\ldots,n$
  = path in transition diagram starting from q_0 with label x
  (label of path = sequence of labels of the edges)
- Accepting computation (run, path) = path from start state q_0 to a state in F.
- Input string x is accepted by A iff there is an accepting computation on input x, i.e. there is a path from q_0 to a node in F labeled by x.
- Language of A, L(A) = $\{ x \in \Sigma^* \mid x \text{ is accepted by } A \}$
  = set of labels of all accepting paths

Finding all computations for an input



- Grow a tree of computations



1 accepting computation