CS310

Finite Automata

Aug 29, 2014

Quick Review

- Alphabet: $\sum = \{a,b\}$
  $\sum^*$: Closure:
- String: any finite sequence of symbols from a given alphabet. $|w|$ = length
  Concatenation/Prefix/Suffix/Reverse
- Language L over $\sum$ is a subset of $\sum^*$
  $L= \{ x \mid \text{rule about } x\}$
  Concatenation/Union/Kleene Star
  Recursive Definition

Finite State Automata

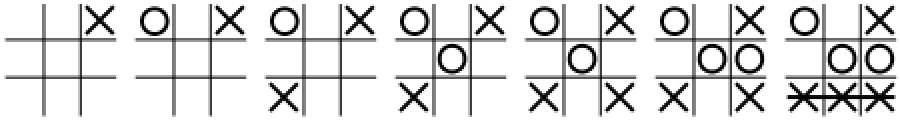
- How can we reason about computation?

- Simple model of computation
  - Finite
  - State
  - Automata

  - Memory?

- Many Automata
- One automaton

Example



How would we represent Tic-tac-toe in C/C++?

How is this different than a finite state automata?

X always goes first.

How many possible board configurations (ignore the rules)?

How many possible *valid* tic-tac-toe configurations?

http://en.wikipedia.org/wiki/File:Tic-tac-toe-game-1.svg

Computation

- Recognize patterns in data

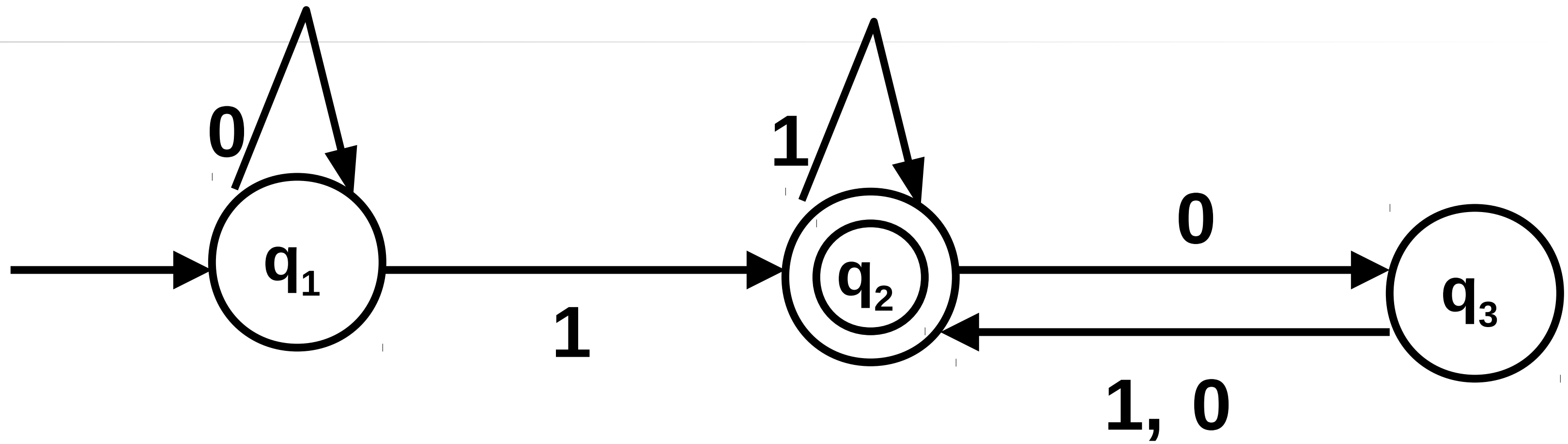
- Build an automaton that can classify a string as part of a language or not
- Why?

Language:

$L = \{ x \in \{0,1\}^* \mid x$ contains at least one 1 and the last 1 is followed by even number of 0s$\}$

$T = \{ x \mid x$ represents a winning tic-tac-toe board$\}$

Deterministic Finite Automata



Inputs: Accept or Reject?

$\sum = \{0, 1\}$

1101	**1110**	**0**
1100	**1**	**11**

Set of all strings (A) accepted by a machine (M) is the *Language of the Machine*
M *recognizes* A or M *accepts* A

Formal Definition

- Deterministic Finite Automata:|

5-tuple (Q, ∑ , δ,q_0, F)

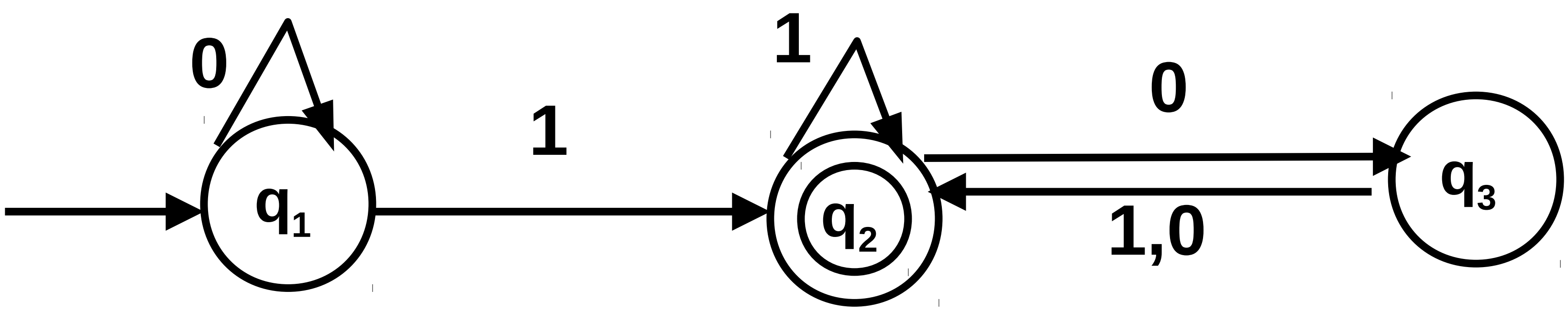
Q: finite set of states

∑: alphabet (finite set)

δ : transition function (δ: Qx∑ **->**Q)

q_0: start state

F : accepting states (subset of Q)



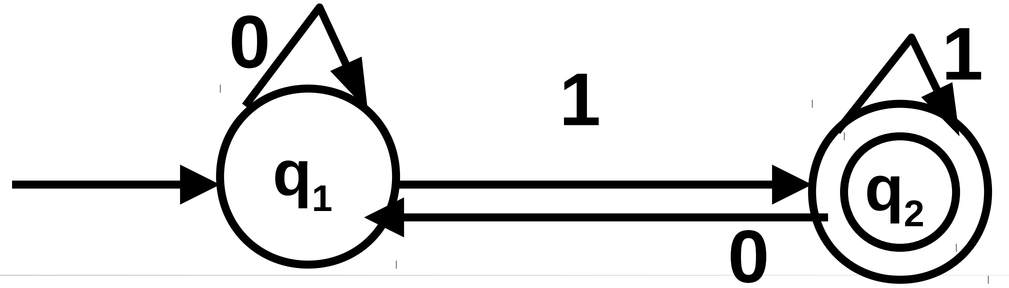
Q: finite set of states

∑: alphabet

δ : transition function

q_0: start state

F : accepting states



Q:

∑:

δ :

q_0:

F :

What strings get accepted?

L(M) = { }

Designing a DFA

- Identify small pieces
  - alphabet, each state needs a transition for each symbol
  - finite memory, what crucial data does the machine look for?
  - can things get hopeless? do we need a trap?
  - where should the empty string be?
  - what is the transition into the accept state?
  - can you transition out of the accept state?
- Practice!

$L(M) = \{ w \mid w = \varepsilon \text{ or } w \text{ ends in } 1\}$

$\Sigma = \{ 0,1\}$

Q:

δ :

q_0:

F :

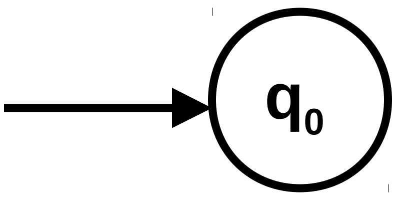
- $\Sigma = \{0,1\}$, $L(M)=\{w \mid \text{odd \# of 1s}\}$

Build a DFA to do math!

L(M) = Accept sums that are multiples of 3

∑ = { 0,1,2, <Reset>}

Keep a running total of input, modulo 3



- $\sum = \{0,1\}$, L(M)={w | begins with 1, ends with 0}

- $\sum = \{0,1\}$, L(M)={w | contains 110}

- $\sum = \{0,1\}$, L(M)={w | does not contain 110}

- $\sum = \{0,1\}$, $L(M)=\{w \mid (01)^* \}$

- $\sum = \{0,1\}$, $L(M)=\{w \mid w \text{ even } \#0\text{s, odd } \#1\text{s} \}$

- $\sum = \{0,1\}$, L(M)={w | w any string except 11 and 111 }

Formal Definition of Computing

- Given a machine $M = (Q, \Sigma, \delta, q_0, F)$ and a string $w = w_1 w_2 \ldots w_n$ over Σ, then M ***accepts*** w if there exists a sequence of states $r_0, r_1 \ldots r_n$ in Q such that:
  - $r_0 = q_0$: r_0 is the start state
  - $\delta(r_i, w_{i+1}) = r_{i+1}$, $i = 0, \ldots, n-1$: legal transitions
  - $r_n \in F$: stop in an accept state
- M ***recognizes*** A if A={w | M accepts w}
- Language A is ***regular*** if there exists a Finite Automaton that recognizes A.