Quick Review

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

Computation

• Recognize patterns in data
• Build an automaton that can classify a string as part of a language or not
• Why?

Language:

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

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

Deterministic Finite Automata

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 \((Q, \Sigma, \delta, q_0, F)\)
- \(Q\): finite set of states
- \(\Sigma\): alphabet (finite set)
- \(\delta\): transition function \((\delta: Q \times \Sigma \rightarrow Q)\)
- \(q_0\): start state
- \(F\): accepting states (subset of \(Q\))
Q: What strings get accepted?
\[ \Sigma: \]
\[ \delta: \]
\[ q_0: \]
\[ F: \]
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:
\delta :
q_0 :
F :

\bullet \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
$\Sigma = \{0,1,2, \text{<Reset>}\}$

Keep a running total of input, modulo 3

$\bullet \Sigma = \{0,1\}, \ L(M)=\{w \mid \text{begins with 1, ends with 0}\}$
\[ \Sigma = \{0,1\}, \ L(M) = \{w \mid \text{contains 110}\} \]

\[ \Sigma = \{0,1\}, \ L(M) = \{w \mid \text{does not contain 110}\} \]
• $\Sigma = \{0, 1\}$, $L(M) = \{w \mid (01)^* \}$

• $\Sigma = \{0, 1\}$, $L(M) = \{w \mid w$ even #0s, odd #1s $\}$
• $\Sigma = \{0,1\}$, $L(M) = \{w \mid w \text{ any string except 11 and 111}\}$

---

**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 $\Sigma$, 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

• If $A = \{w \mid M \text{ accepts } w\}$

• Language $A$ is **regular** if there exists a Finite Automaton that recognizes $A$. 

---

CS 310 – Fall 2016  
Pacific University