Finite State Machines 1

95-771 Data Structures and Algorithms for Information Processing

Some Results First

Computing Model	Finite Automata	Pushdown Automata	Linear Bounded Automata	Turing Machines
Language Class	Regular Languages	Context-Free Languages	Context-Sensitive Languages	Recursively Enumerable Languages
Non-determinism	Makes no difference	Makes a difference	No one knows	Makes no difference

Alphabets

- Σ (sigma): a finite (non-empty) set of symbols called the *alphabet.*

- Each symbol in Σ is a *letter.*

- Letters in the alphabet are usually denoted by lower case letters: a, b, c, …

Strings

- A *word* w is a string of letters from Σ in a linear sequence.
- We are interested only in finite words (bounded length).
- $|w|$ denotes the *length* of word w.
- The *empty string* contains no letters and is written as ε.

Languages

- A *language* L is a set (finite or infinite) of words from a given Σ.

- The set of all strings over some fixed alphabet Σ is denoted by Σ^*.
- For example, if $\Sigma = \{a\}$,

  then $\Sigma^* = \{\varepsilon, a, aa, aaa, \ldots\}$.

Languages

- The set of all strings of length i over some fixed alphabet Σ is denoted by Σ^i.
- For example, let $\Sigma = \{a, b\}$.
- Then $L = \Sigma^2 = \{aa, ab, ba, bb\}$ is the set of words w such that $|w| = 2$.

Operations on Words and Languages

- Concatenation: putting two strings together

  x = aa; y=bb; x.y = xy = aabb

- Power: concatenating multiple copies of a letter or word

  $a^n = a.a^{n-1}$; $a^1 = a$; $a^2 = a.a$; etc.

  x = ab; x^3 = ababab

- Kleene Star: zero or more copies of a letter or word

  a* = {ε, a, aa, aaa, …}

  x = ab; x* = {ε, ab, abab, ababab, …}

Deterministic, Finite State Automata

- A *finite-state automaton* comprises the following elements:
  - A sequence of *input symbols* (the input "tape").
  - The current location in the input, which indicates the current input symbol (the read "head").
  - The current *state* of the machine (denoted $q_0, q_1, \ldots, q_n$).
  - A *transition function* which inputs the current state and the current input, and outputs a new (next) state.

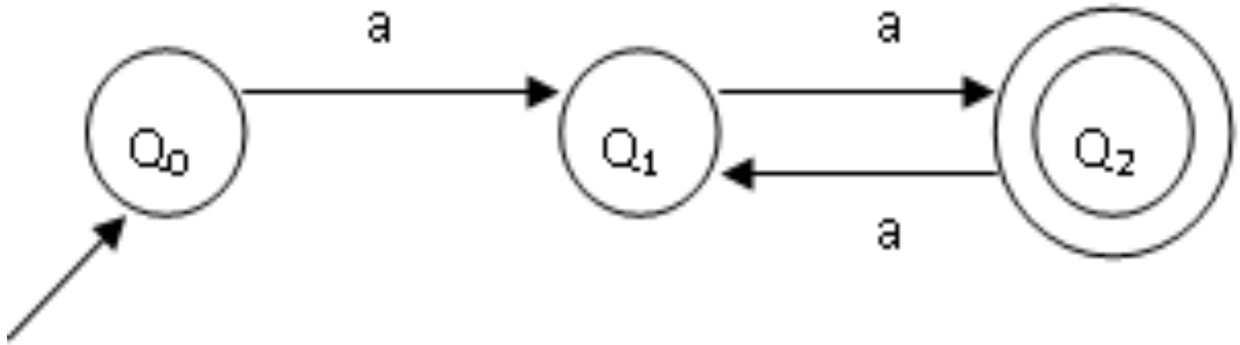
During computation,

- The FSA begins in the start state (usually, q0).
- At each step, the transition function is called on the current input symbol and the current state, the state is updated to the new state, and the read head moves one symbol to the right.
- The end of computation is reached when the FSA reaches the end of the input.

One or more states may be marked as final states, such that the computation is considered successful if and only if computation ends in a final state.

An Example

- An FSA can be represented graphically as a directed graph, where the nodes in the graph denote states and the edges in the graph denote transitions. Final states are denoted by a double circle.
- For example, here is a graphical representation of a DFSA that accepts the language $L = \{a^{2n} : n \geq 1\}$:



- Input: aaa

  States: q_0, q_1, q_2, q_1 (not accepted)

- Input: aaaa

  States: q_0, q_1, q_2, q_1, q_2 (accepted)

DFSA Definition

A DFSA can be formally defined as

$A = (Q, \Sigma, \partial, q_0, F)$:

- Q, a finite set of states
- Σ, a finite alphabet of input symbols
- $q_0 \in Q$, an initial start state
- $F \subseteq Q$, a set of final states
- ∂ (delta): $Q \times \Sigma \rightarrow Q$, a transition function

Transition function - ∂

- We can expand the notion of ∂ on letters to ∂ on words, ∂_w, by using a recursive definition:
- $\partial_w : Q \times \Sigma^* \rightarrow Q$ - (a function of (state, word) to a state)
- $\partial_w(q,\varepsilon) = q$ - (in state q, output state q if word is ε)
- $\partial_w(q,xa) = \partial(\partial_w(q,x),a)$ - (otherwise, use ∂ for one step and recurse)

Language Recognition

- For an automaton A, we can define the language of A:

  - $L(A) = \{w \in \Sigma^* : \partial_w(q_0,w) \in F\}$
  - L(A) is a subset of all words w of finite length over Σ, such that the transition function $\partial_w(q_0,w)$ produces a state in the set of final states (F).
  - Intuitively, if we think of the automaton as a graph structure, then the words in L(A) represent the "paths" which end in a final state. If we concatenate the labels from the edges in each such path, we derive a string $w \in L(A)$.

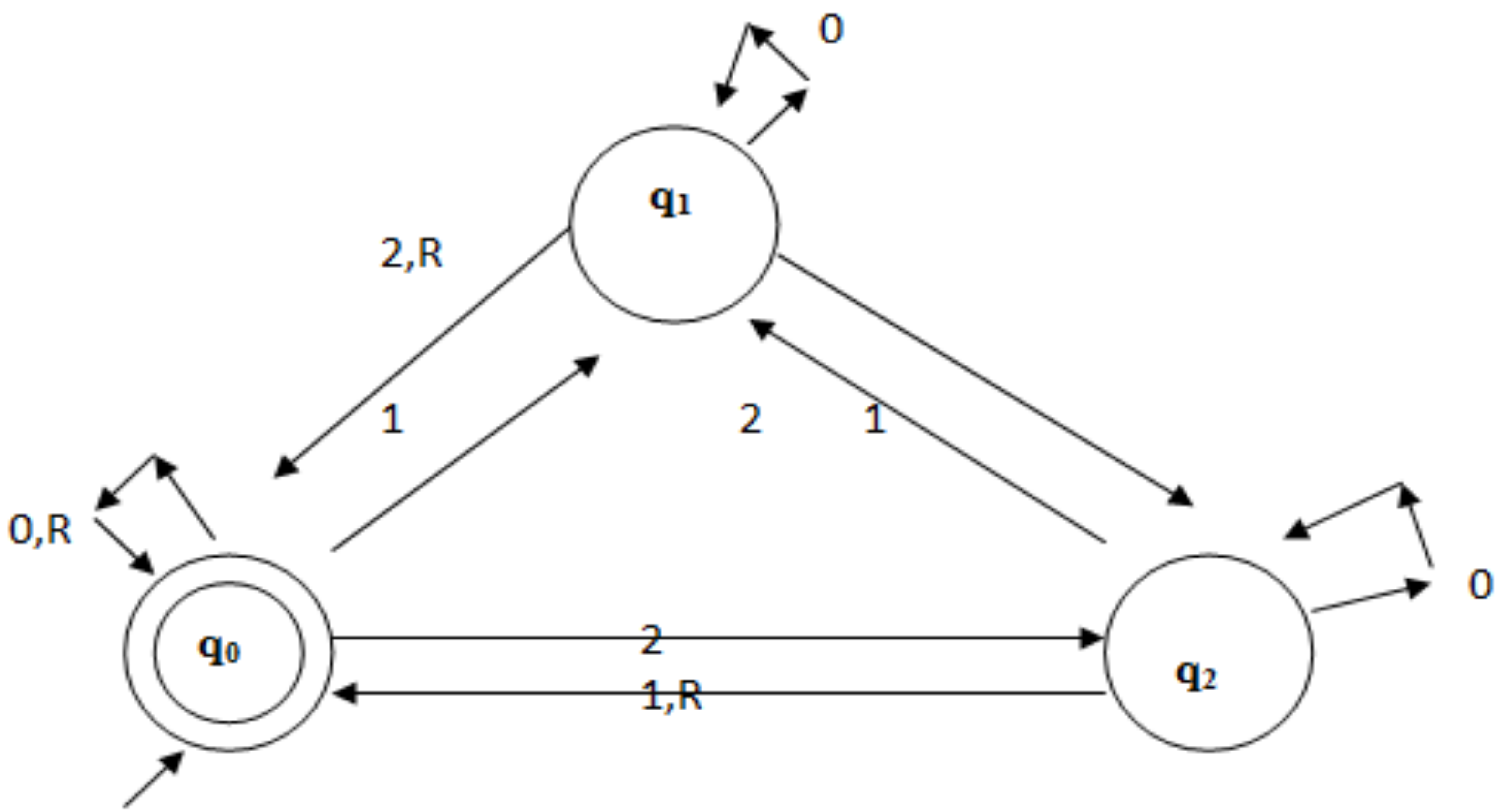
Another Example

$Q = \{ q_0, q_1, q_2 \}$
$\Sigma = \{ R,0,1,2 \}$
q_0 : the start state
$F = \{ q_0 \}$
∂ (delta): $Q \times \Sigma \rightarrow Q$

Next, we define delta with a directed graph.

This automaton keeps a running count of the sum of the numerical input symbols it reads, modulo 3. Every time it receives the R (reset) symbol it resets the count to 0. It accepts the if the sum is 0, modulo 3, or in other words, if the sum is a multiple of 3.

This automaton is from "Introduction to the Theory of Computation" by Michael Sipser



Homework Questions (not to be turned in but to prepare for exam)

Give state diagrams for DFA's recognizing the following languages. $\Sigma = \{ 0,1 \}$.

1. { w | w begins with a 1 and ends with a 0 }
2. { w | w contains at least three 1's }
3. { w | the length of w is at most 5 }
4. { w | w contains at least two 0's and at most one 1. }
5. { w | w contains an even number of 0's, or exactly two 1's }
6. { w | w is not ε }