Lecture 1 - DFAs

Eric A. Autry

Last time: Review Material

This time: Review of Graphs, DFAs

Next time: NFAs

Homework 1 will be assigned tonight through Sakai and due next Thursday in class.

Graphs - Definitions

- An undirected graph is a set of points with lines connecting some of them.
- The points are called the vertices, and the lines are called the edges.
- The number of edges at a particular vertex is the degree of that vertex.

Graphs - Definitions

- A path in a graph is a sequence of connected vertices.
- A simple path is a path that does not repeat vertices.
- A cycle is a path that starts and ends at the same vertex.
- A simple cycle is a cycle that contains at least three vertices, with the only repeated vertex being the first and last.

Graphs - Definitions

- A graph is connected if every two vertices have a path connecting them.
- A graph that is connected and contains no simple cycles is called a tree.
- A tree can contain a specially designated vertex called the root.
- The vertices of degree one in a tree, other than the root, are called leaves.

Graphs - Definitions

- A directed graph has arrows instead of lines to represent directed edges, where the connections are one-way.
- The outdegree of a vertex is the number of edges pointing out of the vertex.
- The indegree of a vertex is the number of edges pointing into the vertex.
- A path in which all the edges point in the same direction as its steps is called a directed path.
- A directed graph where every two vertices are connected by a directed path is called strongly connected.

Graphs - Notation

We can label the vertices in a graph (typically by numbering them), and can use these labels to create a mathematical representation of the graph:

- If a graph G contains vertices i and j, the pair (i,j) represents an edge connecting the two.
- In an undirected graph, the order of the pair does not matter, with (i,j) and (j,i) representing the same edge.
- In a directed path, the order of the pair matters: (i,j) is the edge pointing from i to j, while (j,i) is the edge pointing from j to i.
- If V is the set of vertices in a graph, and E is the set of edges, we say $G=(V,E)$.

$$G = (\ \{1,2,3,4\},\ \{\,(1,2),\ (1,3),\ (1,4),\ (2,3),\ (2,4),\ (3,4)\,\}\ ).$$

Graphs - Subgraphs

We say that a graph H is a subgraph of a graph G if:

- The vertices in H are a subset of the vertices in G,
- The edges of H are the edges of G on the corresponding vertices.

Finite State Machines

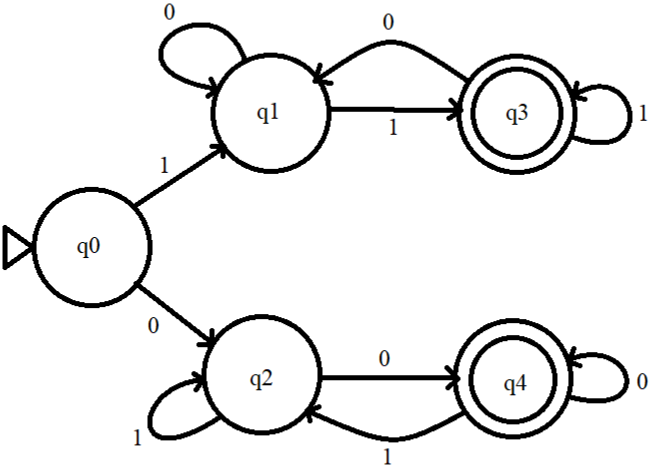
aka

Finite Automata (singular: automaton)

Example: an automatic door

What if we want a one way door?

State Diagram of M_1



State Diagram of M_1

- This image is the state diagram of the automaton M_1.
- M_1 has 5 states labeled q_0, q_1, q_2, q_3, and q_4.
- The start state is q_0 as indicated by the wedge on the left.
- The accepting states are q_3 and q_4 as indicated by the double circle.
- The labeled arrows between states are transitions.

When a finite automaton is given an input string, it processes that string and produces an output: it either accepts or rejects the string.

Ex: 1001101

Let's experiment with some strings: ε, 0, 1, 00, 01, 11, 101

State Diagram of M_1

Start	$\rightarrow$	q_0
1	$\rightarrow$	q_1
0	$\rightarrow$	q_1
0	$\rightarrow$	q_1
1	$\rightarrow$	q_3
1	$\rightarrow$	q_3
0	$\rightarrow$	q_1
1	$\rightarrow$	q_3

ε	-	q_0	-	Reject
0	-	q_2	-	Reject
1	-	q_1	-	Reject
00	-	q_4	-	Accept
01	-	q_2	-	Reject
11	-	q_3	-	Accept
101	-	q_3	-	Accept

Accept string 1001101 in state q_3.

It accepts strings that being and end with the same symbol AND that are at least length 2.

Formal Definition

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the states,
2. Σ is a finite set of symbols called the alphabet,
3. $\delta : Q \times \Sigma \to Q$ is the transition function,
4. $q_0 \in Q$ is the start state,
5. $F \subseteq Q$ is the set of accepting states.

Formal Definition - Example

For our example machine M_1, we have:

1. $Q = \{q_0, q_1, q_2, q_3, q_4\}$,
2. $\Sigma = \{0, 1\}$,
3. δ is defined by:

$$\begin{aligned} \delta(q_0, 0) &= q_2, & \delta(q_0, 1) &= q_1, \\ \delta(q_1, 0) &= q_1, & \delta(q_1, 1) &= q_3, \\ \delta(q_2, 0) &= q_4, & \delta(q_2, 1) &= q_2, \\ \delta(q_3, 0) &= q_1, & \delta(q_3, 1) &= q_3, \\ \delta(q_4, 0) &= q_4, & \delta(q_4, 1) &= q_2. \end{aligned}$$

4. q_0 is given as the start state,
5. $F = \{q_3, q_4\}$.

Recognizing Languages

Recall: a language is a set of strings.

If A is the set of all strings that a machine M accepts, we say that A is the language of the machine M, and write $L(M) = A$.

We say that machine M recognizes language A.

It is also valid to say machine M accepts language A, but this terminology can get confusing.

Ex:

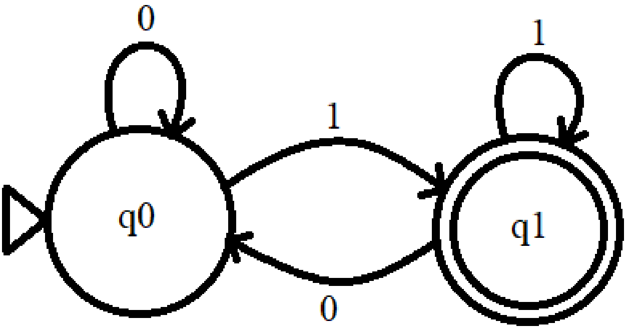
$L(M_1) = \{w \mid w \text{ starts and ends with the same symbol and } |w| \geq 2\}$

Regular Languages

Definition:

A language is called a regular language if some finite automaton recognizes it.

Example M_2



Example M_2

1. $Q = \{q_0, q_1\}$,
2. $\Sigma = \{0, 1\}$,
3. δ is defined by:

	0	1
q_0	q_0	q_1
q_1	q_0	q_1

4. q_0 is given as the start state,
5. $F = \{q_1\}$.

Example M_2

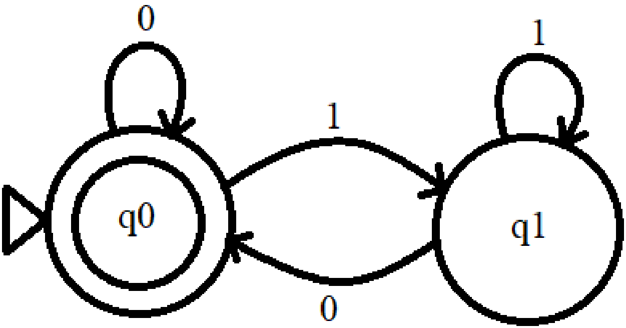
ε	-	Reject	11	-	Accept
0	-	Reject	10	-	Reject
1	-	Accept	101	-	Accept
00	-	Reject	010	-	Reject
01	-	Accept	110	-	Reject

- q_0 is the state ‘empty or the last thing we saw was a 0’
- q_1 is the state ‘the last thing we saw was a 1’

Since we accept in state q_1:

$$L(M_2) = \{w \mid w \text{ ends in } 1\}.$$

Example M_3



Example M_3

Swapped accepting states.

1. $Q = \{q_0, q_1\}$,
2. $\Sigma = \{0, 1\}$,
3. δ is defined by:

	0	1
q_0	q_0	q_1
q_1	q_0	q_1

4. q_0 is given as the start state,
5. $F = \{q_0\}$.

Now:

$$L(M_3) = \{w \mid w = \varepsilon \text{ or } w \text{ ends in } 0\}.$$

Designing DFAs

How to create a DFA? Think like the machine:

- Step 1: What information do we need to store? What should our states be in order to store that information?
- Step 2: Which state is the starting state?
- Step 3: Which state(s) are the accepting state(s)?
- Step 4: Fill in the transitions one state at a time.

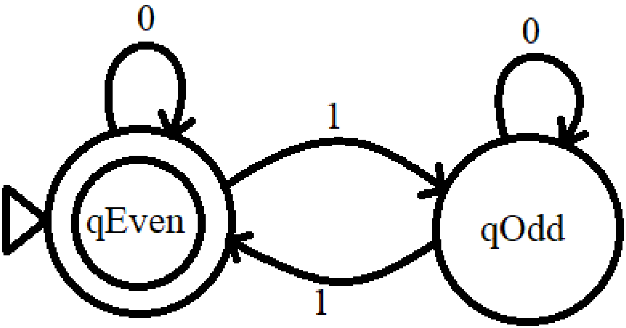
Example M_4

Create a DFA that recognizes the language:

$$L(M_4) = \{w \mid w \text{ has an even number of } 1\text{'s}\}.$$

- Step 1: What info?
  - Even 1's vs Odd 1's. Have two states.
- Step 2: Starting state?
  - Empty string has even number of 1's. Start in state $qEven$.
- Step 3: Accepting state(s)?
  - Want even number of 1's, so $qEven$.
- Step 4: Transitions?

Example M_4



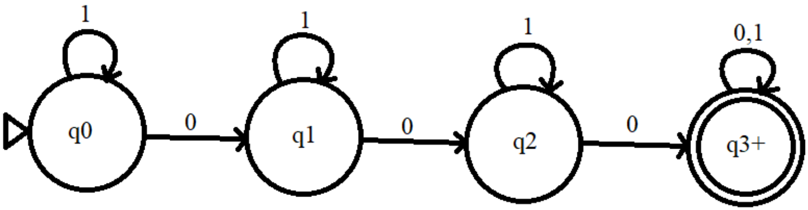
Example M_5

Create a DFA that recognizes the language:

$$L(M_5) = \{w \mid w \text{ contains at least three } 0\text{'s}\}.$$

- Step 1: What info?
  - Number of 0's: q_0, q_1, q_2, q_{3+}
- Step 2: Starting state?
  - Empty string has zero 0's. Start in state q_0.
- Step 3: Accepting state(s)?
  - Want at least three 0's. So accept state q_{3+}.
- Step 4: Transitions?

Example M_5

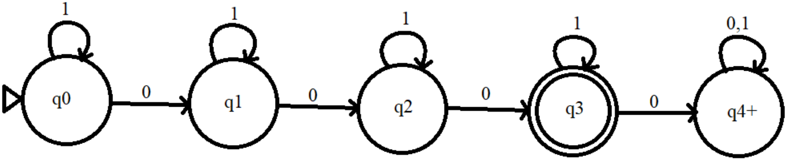


Example M_6

What if we change the language to:

$$L(M_5) = \{w \mid w \text{ contains exactly three } 0\text{'s}\}?$$

Need a new state for more than three 0's seen: q_{4+}.



Example M_7

Consider the alphabet:

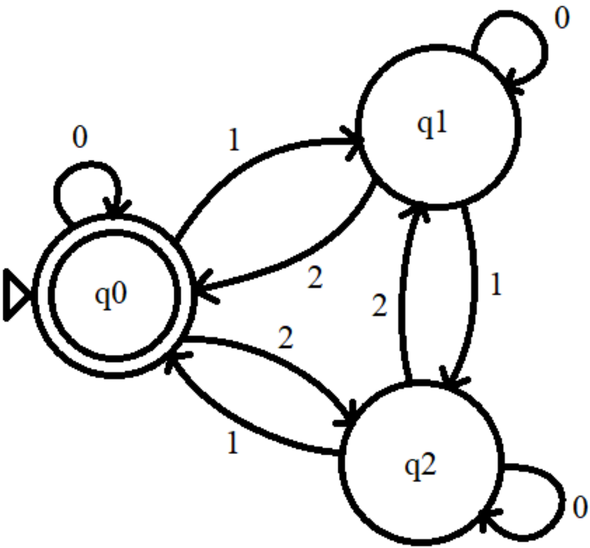
$$\Sigma = \{0, 1, 2\}.$$

Create a DFA that recognizes the language:

$$L(M_7) = \{w \mid \text{the sum of the digits of } w \text{ is divisible by } 3\}.$$

- Step 1: What info?
  - Modulo 3: q_0, q_1, q_2
- Step 2: Starting state?
  - Empty string's digits sum to 0, so start in q_0.
- Step 3: Accepting state(s)?
  - Multiples of 3 are 0 mod 3. So accept q_0.
- Step 4: Transitions?

Example M_7



Regular Operations

Let A and B be languages.

- Union: $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$
- Concatenation: $A \circ B = \{w_1 w_2 \mid w_1 \in A \text{ and } w_2 \in B\}$
- Star: $A^* = \{w_1 w_2 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in A\}$
  - What if k=0?
  - $\varepsilon \in A^*$

Regular Operations - Examples

- Let $A = \{good,\ bad\}$
- Let $B = \{dog,\ cat\}$

$$A \cup B = \{good,\ bad,\ dog,\ cat\}$$

$$A \circ B = \{gooddog,\ goodcat,\ baddog,\ badcat\}$$

$$\begin{aligned} A^* \quad = \quad & \{\varepsilon,\ good,\ bad,\ goodgood,\ badbad,\ goodbad, \\ & badgood,\ goodgoodgood,\ goodgoodbad,\ \ldots\} \end{aligned}$$

Union

Can we prove that regular languages are closed under the union operation?

In other words, can we prove that if A and B are regular languages, then $A \cup B$ is also a regular language?

Idea: Recall that a regular language is one that can be recognized by a finite automaton. So, let's build an automaton.

Union

Can we prove that if A and B are regular languages, then $A \cup B$ is also a regular language?

Since A and B are regular languages, we know that there must exist a machine M_1 that recognizes A and a machine M_2 that recognizes B.

Our goal is to create a machine M that recognizes $A \cup B$ using machines M_1 and M_2.

Can we just run M_1, see if that works, then run M_2?

Nope. We can't rewind the input!