# DFA

Noman Amin
Lecturer, Dept. of CSE, Southeast University

# DFA

A *deterministic finite automaton* consists of:

1. A finite set of *states*, often denoted $Q$.
2. A finite set of *input symbols*, often denoted $\Sigma$.
3. A *transition function* that takes as arguments a state and an input symbol and returns a state. The transition function will commonly be denoted $\delta$. In our informal graph representation of automata, $\delta$ was represented by arcs between states and the labels on the arcs. If $q$ is a state, and $a$ is an input symbol, then $\delta(q, a)$ is that state $p$ such that there is an arc labeled $a$ from $q$ to $p$.[2]
4. A *start state*, one of the states in $Q$.
5. A set of *final* or *accepting* states $F$. The set $F$ is a subset of $Q$.

A deterministic finite automaton will often be referred to by its acronym: *DFA*. The most succinct representation of a DFA is a listing of the five components above. In proofs we often talk about a DFA in "five-tuple" notation:

$$A = (Q, \Sigma, \delta, q_0, F)$$

where $A$ is the name of the DFA, $Q$ is its set of states, $\Sigma$ its input symbols, $\delta$ its transition function, $q_0$ its start state, and $F$ its set of accepting states.

# Example

**Example 2.1 :** Let us formally specify a DFA that accepts all and only the strings of 0's and 1's that have the sequence 01 somewhere in the string. We can write this language $L$ as:

$$\{w \mid w \text{ is of the form } x01y \text{ for some strings } x \text{ and } y \text{ consisting of 0's and 1's only}\}$$

Another equivalent description, using parameters $x$ and $y$ to the left of the vertical bar, is:

$$\{x01y \mid x \text{ and } y \text{ are any strings of 0's and 1's}\}$$

Examples of strings in the language include 01, 11010, and 100011. Examples of strings *not* in the language include $\epsilon$, 0, and 111000.

# Solution

1. Has it already seen 01? If so, then it accepts every sequence of further inputs; i.e., it will only be in accepting states from now on.

2. Has it never seen 01, but its most recent input was 0, so if it now sees a 1, it will have seen 01 and can accept everything it sees from here on?

3. Has it never seen 01, but its last input was either nonexistent (it just started) or it last saw a 1? In this case, $A$ cannot accept until it first sees a 0 and then sees a 1 immediately after.

# Solution Cont'd

Thus, $Q = \{q_0, q_1, q_2\}$. $q_0$ is the start state, and the only accepting state is $q_1$; that is, $F = \{q_1\}$. The complete specification of the automaton $A$ that accepts the language $L$ of strings that have a 01 substring, is

$$A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

where $\delta$ is the transition function described above. □

# Simpler Notations

Specifying a DFA as a five-tuple with a detailed description of the $\delta$ transition function is both tedious and hard to read. There are two preferred notations for describing automata:

1. A *transition diagram*, which is a graph such as the ones we saw in Section 2.1.

2. A *transition table*, which is a tabular listing of the $\delta$ function, which by implication tells us the set of states and the input alphabet.

# Transition Diagrams

**Transition Diagrams**

A *transition diagram* for a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is a graph defined as follows:

a) For each state in $Q$ there is a node.

b) For each state $q$ in $Q$ and each input symbol $a$ in $\Sigma$, let $\delta(q, a) = p$. Then the transition diagram has an arc from node $q$ to node $p$, labeled $a$. If there are several input symbols that cause transitions from $q$ to $p$, then the transition diagram can have one arc, labeled by the list of these symbols.

c) There is an arrow into the start state $q_0$, labeled *Start*. This arrow does not originate at any node.

d) Nodes corresponding to accepting states (those in $F$) are marked by a double circle. States not in $F$ have a single circle.

# Example

**Example 2.2:** Figure 2.4 shows the transition diagram for the DFA that we designed in Example 2.1. We see in that diagram the three nodes that correspond to the three states. There is a *Start* arrow entering the start state, $q_0$, and the one accepting state, $q_1$, is represented by a double circle. Out of each state is one arc labeled 0 and one arc labeled 1 (although the two arcs are combined into one with a double label in the case of $q_1$). The arcs each correspond to one of the $\delta$ facts developed in Example 2.1. □

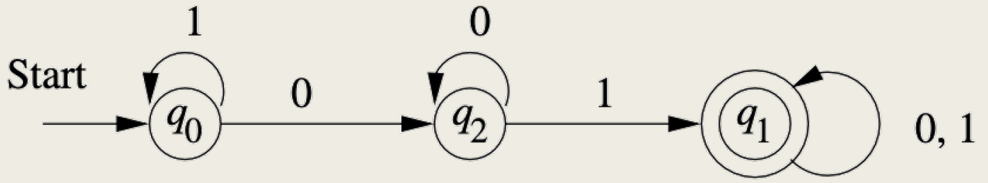



Figure 2.4: The transition diagram for the DFA accepting all strings with a substring 01

# Transition Tables

**Transition Tables**

A *transition table* is a conventional, tabular representation of a function like $\delta$ that takes two arguments and returns a value. The rows of the table correspond to the states, and the columns correspond to the inputs. The entry for the row corresponding to state $q$ and the column corresponding to input $a$ is the state $\delta(q, a)$.

# Example

| | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $q_2$ | $q_0$ |
| $*q_1$ | $q_1$ | $q_1$ |
| $q_2$ | $q_2$ | $q_1$ |

Figure 2.5: Transition table for the DFA of Example 2.1

# Extending the Transition Function to Strings

Now, we need to make the notion of the language of a DFA precise. To do so, we define an *extended transition function* that describes what happens when we start in any state and follow any sequence of inputs. If $\delta$ is our transition function, then the extended transition function constructed from $\delta$ will be called $\hat{\delta}$. The extended transition function is a function that takes a state $q$ and a string $w$ and returns a state $p$ — the state that the automaton reaches when starting in state $q$ and processing the sequence of inputs $w$. We define $\hat{\delta}$ by induction on the length of the input string, as follows:

**BASIS**: $\hat{\delta}(q, \epsilon) = q$. That is, if we are in state $q$ and read no inputs, then we are still in state $q$.

**INDUCTION**: Suppose $w$ is a string of the form $xa$; that is, $a$ is the last symbol of $w$, and $x$ is the string consisting of all but the last symbol.[3] For example, $w = 1101$ is broken into $x = 110$ and $a = 1$. Then

$$\hat{\delta}(q, w) = \delta\big(\hat{\delta}(q, x), a\big) \tag{2.1}$$

Now (2.1) may seem like a lot to take in, but the idea is simple. To compute $\hat{\delta}(q, w)$, first compute $\hat{\delta}(q, x)$, the state that the automaton is in after processing all but the last symbol of $w$. Suppose this state is $p$; that is, $\hat{\delta}(q, x) = p$. Then $\hat{\delta}(q, w)$ is what we get by making a transition from state $p$ on input $a$, the last symbol of $w$. That is, $\hat{\delta}(q, w) = \delta(p, a)$.

# Example

**Example 2.4:** design a DFA to accept the language

$$L = \{w \mid w \text{ has both an even number of 0's and an even number of 1's}\}$$

# Solution

It should not be surprising that the job of the states of this DFA is to count both the number of 0's and the number of 1's, but count them modulo 2. That is, the state is used to remember whether the number of 0's seen so far is even or odd, and also to remember whether the number of 1's seen so far is even or odd. There are thus four states, which can be given the following interpretations:

# Solution Cont'd

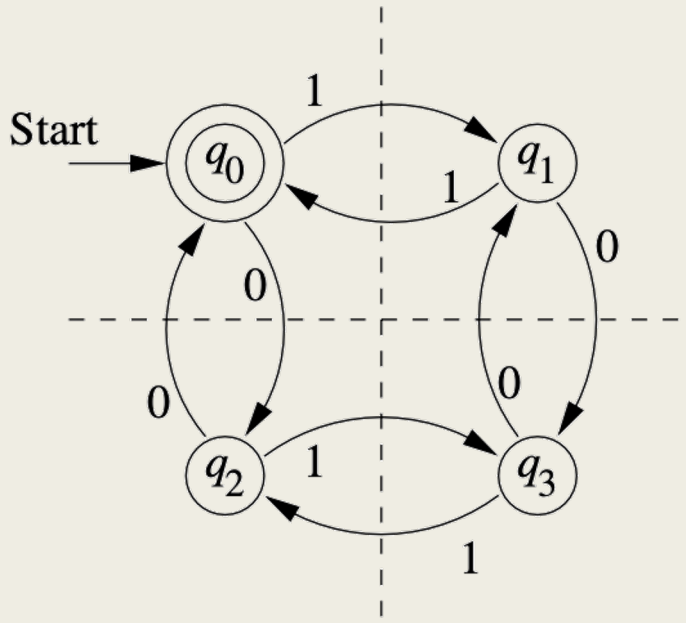
$q_0$: Both the number of 0's seen so far and the number of 1's seen so far are even.

$q_1$: The number of 0's seen so far is even, but the number of 1's seen so far is odd.

$q_2$: The number of 1's seen so far is even, but the number of 0's seen so far is odd.

$q_3$: Both the number of 0's seen so far and the number of 1's seen so far are odd.

# Solution Cont'd




We now know almost how to specify the DFA for language $L$. It is

$$A = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_0\})$$

# Solution Cont'd

| | 0 | 1 |
|---|---|---|
| $* \rightarrow q_0$ | $q_2$ | $q_1$ |
| $q_1$ | $q_3$ | $q_0$ |
| $q_2$ | $q_0$ | $q_3$ |
| $q_3$ | $q_1$ | $q_2$ |

Figure 2.7: Transition table for the DFA of Example 2.4

# Example String for the Previous DFA

The check involves computing $\hat{\delta}(q_0, w)$ for each prefix $w$ of 110101, starting at $\epsilon$ and going in increasing size. The summary of this calculation is:

- $\hat{\delta}(q_0, \epsilon) = q_0$.
- $\hat{\delta}(q_0, 1) = \delta\big(\hat{\delta}(q_0, \epsilon), 1\big) = \delta(q_0, 1) = q_1$.
- $\hat{\delta}(q_0, 11) = \delta\big(\hat{\delta}(q_0, 1), 1\big) = \delta(q_1, 1) = q_0$.
- $\hat{\delta}(q_0, 110) = \delta\big(\hat{\delta}(q_0, 11), 0\big) = \delta(q_0, 0) = q_2$.
- $\hat{\delta}(q_0, 1101) = \delta\big(\hat{\delta}(q_0, 110), 1\big) = \delta(q_2, 1) = q_3$.
- $\hat{\delta}(q_0, 11010) = \delta\big(\hat{\delta}(q_0, 1101), 0\big) = \delta(q_3, 0) = q_1$.
- $\hat{\delta}(q_0, 110101) = \delta\big(\hat{\delta}(q_0, 11010), 1\big) = \delta(q_1, 1) = q_0$.

# Exercise

- $Q = \{a, b, c\}$,
- $\Sigma = \{0, 1\}$,
- $q_0 = \{a\}$,
- $F = \{c\}$, and

| Present State | Next State for Input 0 | Next State for Input 1 |
|---|---|---|
| a | a | b |
| b | c | a |
| c | b | c |

# Exercise

**Exercise** : Give DFA's accepting the following languages over the alphabet $\{0, 1\}$:

a) The set of all strings ending in 00.

b) The set of all strings with three consecutive 0's (not necessarily at the end).

c) The set of strings with 011 as a substring.

# Exercise

- The set of all strings over the alphabet {0, 1} that end with a 1.
- The set of all strings over the alphabet {0, 1} that contain an even number of 1s.
- The set of all strings over the alphabet {0, 1} that contain at least one 1.

# Exercise

**Exercise** : Consider the DFA with the following transition table:

| | 0 | 1 |
|---|---|---|
| $\rightarrow A$ | $A$ | $B$ |
| $*B$ | $B$ | $A$ |

Informally describe the language accepted by this DFA.

# Exercise

**! Exercise** : Repeat for the following transition table

| | 0 | 1 |
|---|---|---|
| $\rightarrow *A$ | $B$ | $A$ |
| $*B$ | $C$ | $A$ |
| $C$ | $C$ | $C$ |