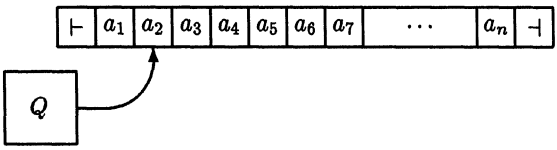
Lecture 17

Two-Way Finite Automata

Two-way finite automata are similar to the machines we have been studying, except that they can read the input string in either direction. We think of them as having a *read head*, which can move left or right over the input string. Like ordinary finite automata, they have a finite set Q of *states* and can be either deterministic (2DFA) or nondeterministic (2NFA).

Although these automata appear much more powerful than one-way finite automata, in reality they are equivalent in the sense that they only accept regular sets. We will prove this result using the Myhill–Nerode theorem.

We think of the symbols of the input string as occupying cells of a finite tape, one symbol per cell. The input string is enclosed in left and right endmarkers $\vdash$ and $\dashv$, which are not elements of the input alphabet Σ. The read head may not move outside of the endmarkers.



Informally, the machine starts in its start state s with its read head pointing to the left endmarker. At any point in time, the machine is in some state q with its read head scanning some tape cell containing an input symbol a_i or

one of the endmarkers. Based on its current state and the symbol occupying the tape cell it is currently scanning, it moves its read head either left or right one cell and enters a new state. It *accepts* by entering a special accept state t and *rejects* by entering a special reject state r. The machine's action on a particular state and symbol is determined by a transition function δ that is part of the specification of the machine.

Example 17.1 Here is an informal description of a 2DFA accepting the set

$$A = \{x \in \{a,b\}^* \mid \#a(x) \text{ is a multiple of 3 and } \#b(x) \text{ is even}\}.$$

The machine starts in its start state scanning the left endmarker. It scans left to right over the input, counting the number of a's mod 3 and ignoring the b's. When it reaches the right endmarker $\dashv$, if the number of a's it has seen is not a multiple of 3, it enters its reject state, thereby rejecting the input—the input string x is not in the set A, since the first condition is not satisfied. Otherwise it scans right to left over the input, counting the number of b's mod 2 and ignoring the a's. When it reaches the left endmarker $\vdash$ again, if the number of b's it has seen is odd, it enters its reject state; otherwise, it enters its accept state. □

Unlike ordinary finite automata, a 2DFA needs only a single accept state and a single reject state. We can think of it as halting immediately when it enters one of these two states, although formally it keeps running but remains in the accept or reject state. The machine need not read the entire input before accepting or rejecting. Indeed, it need not ever accept or reject at all, but may loop infinitely without ever entering its accept or reject state.

Formal Definition of 2DFA

Formally, a 2DFA is an octuple

$$M = (Q, \Sigma, \vdash, \dashv, \delta, s, t, r),$$

where

- Q is a finite set (the *states*),
- Σ is a finite set (the *input alphabet*),
- $\vdash$ is the *left endmarker*, $\vdash \notin \Sigma$,
- $\dashv$ is the *right endmarker*, $\dashv \notin \Sigma$,
- $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \to (Q \times \{L, R\})$ is the *transition function* (L, R stand for left and right, respectively),
- $s \in Q$ is the *start state*,

- $t \in Q$ is the *accept state*, and
- $r \in Q$ is the *reject state*, $r \neq t$,

such that for all states q,

$$\begin{array}{lcll} \delta(q, \vdash) & = & (u, R) & \text{for some } u \in Q, \\ \delta(q, \dashv) & = & (v, L) & \text{for some } v \in Q, \end{array} \tag{17.1}$$

and for all symbols $b \in \Sigma \cup \{\vdash\}$,

$$\begin{array}{ll} \delta(t, b) = (t, R), & \delta(r, b) = (r, R), \\ \delta(t, \dashv) = (t, L), & \delta(r, \dashv) = (r, L). \end{array} \tag{17.2}$$

Intuitively, the function δ takes a state and a symbol as arguments and returns a new state and a direction to move the head. If $\delta(p, b) = (q, d)$, then whenever the machine is in state p and scanning a tape cell containing symbol b, it moves its head one cell in the direction d and enters state q. The restrictions (17.1) prevent the machine from ever moving outside the input area. The restrictions (17.2) say that once the machine enters its accept or reject state, it stays in that state and moves its head all the way to the right of the tape. The octuple is not a legal 2DFA if its transition function δ does not satisfy these conditions.

Example 17.2 Here is a formal description of the 2DFA described informally in Example 17.1 above.

$$Q = \{q_0, q_1, q_2, p_0, p_1, t, r\},$$
$$\Sigma = \{a, b\}.$$

The start, accept, and reject states are q_0, t, and r, respectively. The transition function δ is given by the following table:

	$\vdash$	a	b	$\dashv$
q_0	(q_0, R)	(q_1, R)	(q_0, R)	(p_0, L)
q_1	–	(q_2, R)	(q_1, R)	(r, L)
q_2	–	(q_0, R)	(q_2, R)	(r, L)
p_0	(t, R)	(p_0, L)	(p_1, L)	–
p_1	(r, R)	(p_1, L)	(p_0, L)	–
t	(t, R)	(t, R)	(t, R)	(t, L)
r	(r, R)	(r, R)	(r, R)	(r, L)

The entries marked – will never occur in any computation, so it doesn't matter what we put here. The machine is in states q_0, q_1, or q_2 on the first pass over the input from left to right; it is in state q_i if the number of a's it has seen so far is i mod 3. The machine is in state p_0 or p_1 on the second pass over the input from right to left, the index indicating the parity of the number of b's it has seen so far. □

Configurations and Acceptance

Fix an input $x \in \Sigma^*$, say $x = a_1 a_2 \cdots a_n$. Let $a_0 = \vdash$ and $a_{n+1} = \dashv$. Then

$$a_0 a_1 a_2 \cdots a_n a_{n+1} = \vdash x \dashv .$$

A *configuration* of the machine on input x is a pair (q, i) such that $q \in Q$ and $0 \leq i \leq n+1$. Informally, the pair (q, i) gives a current state and current position of the read head. The *start configuration* is $(s, 0)$, meaning that the machine is in its start state s and scanning the left endmarker.

A binary relation $\xrightarrow[x]{1}$, the *next configuration relation*, is defined on configurations as follows:

$$\delta(p, a_i) = (q, L) \Rightarrow (p, i) \xrightarrow[x]{1} (q, i-1),$$
$$\delta(p, a_i) = (q, R) \Rightarrow (p, i) \xrightarrow[x]{1} (q, i+1).$$

The relation $\xrightarrow[x]{1}$ describes one step of the machine on input x. We define the relations $\xrightarrow[x]{n}$ inductively, $n \geq 0$:

- $(p, i) \xrightarrow[x]{0} (p, i)$; and
- if $(p, i) \xrightarrow[x]{n} (q, j)$ and $(q, j) \xrightarrow[x]{1} (u, k)$, then $(p, i) \xrightarrow[x]{n+1} (u, k)$.

The relation $\xrightarrow[x]{n}$ is just the n-fold composition of $\xrightarrow[x]{1}$. The relations $\xrightarrow[x]{n}$ are functions; that is, for any configuration (p, i), there is exactly one configuration (q, j) such that $(p, i) \xrightarrow[x]{n} (q, j)$. Now define

$$(p, i) \xrightarrow[x]{*} (q, j) \overset{\text{def}}{\iff} \exists n \geq 0 \; (p, i) \xrightarrow[x]{n} (q, j).$$

Note that the definitions of these relations depend on the input x. The machine is said to *accept* the input x if

$$(s, 0) \xrightarrow[x]{*} (t, i) \quad \text{for some } i.$$

In other words, the machine enters its accept state at some point. The machine is said to *reject* the input x if

$$(s, 0) \xrightarrow[x]{*} (r, i) \quad \text{for some } i.$$

In other words, the machine enters its reject state at some point. It cannot both accept and reject input x by our assumption that $t \neq r$ and by properties (17.2). The machine is said to *halt* on input x if it either accepts x or rejects x. Note that this is a purely mathematical definition—the machine doesn't really grind to a halt! It is possible that the machine neither accepts nor rejects x, in which case it is said to *loop* on x. The set $L(M)$ is defined to be the set of strings accepted by M.

Example 17.3 The 2DFA described in Example 17.2 goes through the following sequence of configurations on input *aababbb*, leading to acceptance:

$$(q_0,0),\ (q_0,1),\ (q_1,2),\ (q_2,3),\ (q_2,4),\ (q_0,5),\ (q_0,6),\ (q_0,7),\ (q_0,8),$$
$$(p_0,7),\ (p_1,6),\ (p_0,5),\ (p_1,4),\ (p_1,3),\ (p_0,2),\ (p_0,1),\ (p_0,0),\ (t,1).$$

It goes through the following sequence of configurations on input *aababa*, leading to rejection:

$$(q_0,0),\ (q_0,1),\ (q_1,2),\ (q_2,3),\ (q_2,4),\ (q_0,5),\ (q_0,6),\ (q_1,7),\ (r,6).$$

It goes through the following sequence of configurations on input *aababb*, leading to rejection:

$$(q_0,0),\ (q_0,1),\ (q_1,2),\ (q_2,3),\ (q_2,4),\ (q_0,5),\ (q_0,6),\ (q_0,7),$$
$$(p_0,6),\ (p_1,5),\ (p_0,4),\ (p_0,3),\ (p_1,2),\ (p_1,1),\ (p_1,0),\ (r,1).$$ □