

CSE 105: Computation

by Jy

Contents

1	Deterministic Finite Automaton (DFA)	1
1.1	Expressions of DFA's	1
1.2	Configurations of DFA's	3
2	Nondeterministic Finite Automaton (NFA)	5

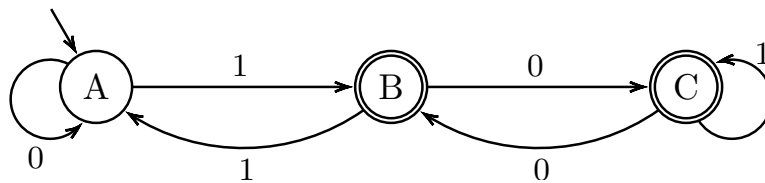
1 Deterministic Finite Automaton (DFA)

A machine consists of different states drawn in circles with names. Often a state drawn as a double circle is an “acceptive state,” and a plain circle indicates a “rejective state.” A machine receives a string consisted of ‘1’s and ‘0’s as input and the states change as the machine reads through input digits. An arrow is used to indicate which state is to start with. See [Example 1.1](#) for detailed information.

1.1 Expressions of DFA’s

Example 1.1: A DFA

Let’s first look at the DFA below which starts at state *A*.



If the string “010110” is input to the machine, will it result in true or false? Will the state be acceptive or rejective?

010110 → M $\xrightarrow{1/0 \text{ (True / False, Accept / Reject)}}$

There are two arrows leaving state *A*: one with a label reading ‘1’ which points to state *B* and one reading ‘0’ which goes back to state *A* itself. That means, if an input digit reads ‘1,’ the state changes to *B*, and if ‘0’ the state stays in *A*.

Now step through the procedure:

1. The machine starts off at state *A* with input ‘0,’ which, as explained above, changes the state to *A* itself.
2. Next, the second digit ‘1’ is read so the state is changed to *B*.
3. The next digit ‘0’ makes state *B* to switch to state *C*.
4. Then state *C* reads ‘1’ so no state change occurs.
5. The next digit is ‘1’ again so the state remains still on *C*.
6. Last, the digit ‘0’ switches the state from *C* to *B*.

Thus the input string “010110” changes the machine to state *B*, which is an acceptive state.

Definition 1.1 DFA. A DFA is a 5-tuple

$$M = (Q, \Sigma, \delta, s, F)$$

where

Q is a finite set, for states

Σ is a finite set, for input alphabet

$s \in Q$, for start states

$F \subseteq Q$, for accepting states

$\delta : Q \times \Sigma \mapsto Q$, a function that specifies the transition between states

Example 1.2: Denoting machine in [Example 1.1](#)

According to definition 1.1, the machine in [Example 1.1](#) can be denoted by

$$M = (Q, \Sigma, \delta, s, F)$$

where

- $Q = \{A, B, C\}$
- $\Sigma = \{0, 1\}$
- $s = \{A\}$
- $F = \{B, C\}$

And function δ can be described by the table below.

δ	0	1
A	A	B
B	C	A
C	B	C

Definition 1.2 f_M . For any DFA $M = (Q, \Sigma, \delta, s, F)$, let

$$f_M : \Sigma^* \mapsto \{\text{True}, \text{False}\}$$

where Σ^* is a set of strings over Σ .

$$f_M(w) = \begin{cases} \text{True}, & \delta^*(s, w) \in F \\ \text{False}, & \text{else} \end{cases}$$

Definition 1.3 δ^* .

$$\delta^* : Q \times \Sigma^* \mapsto Q$$

which is an inductive function defined as

$$\begin{cases} \delta^*(q, \varepsilon) = q \\ \delta^*(q, aw) = \delta^*(\delta(q, a), w) \end{cases}$$

where ε is an empty string and $q \in Q, a \in \Sigma, w \in \Sigma^*$.

1.2 Configurations of DFA's

Definition 1.4 Configurations.

$$\text{Conf} = Q \times \Sigma^*$$

Definition 1.5 Initial Configurations. *The initial configuration of a machine $I_M(w) \in \text{Conf}$*

$$I_M(w) = (s, w)$$

Definition 1.6 Final Configurations. *The final configuration of a machine $H_M(w) \subseteq \text{Conf}$*

$$H_M(w) = \{ (q, u) \mid q \in Q, u = \varepsilon \}$$

Definition 1.7 Output. *The output of a machine is a function that returns either “True” or “False.”*

$$O_M: H_M \mapsto \{ \text{True}, \text{False} \}$$

defined as

$$O_M(q, \varepsilon) = \begin{cases} \text{True}, & q \in F \\ \text{False}, & \text{else} \end{cases}$$

Definition 1.8 . $R_M \subseteq \text{Conf}$

$$R_M = \rightarrow_M = \{ (q, aw) \rightarrow (\delta(q, a), w) \mid q \in Q, a \in \Sigma, w \in \Sigma^* \}$$

Example 1.3: Configurations of machine in [Example 1.1](#)

With input “10010” write in mathematical language the configurations of machine in [Example 1.1](#):

$$I_M(10010) = (A, 10010) \rightarrow (B, 0010) \rightarrow (C, 010) \rightarrow (B, 10) \rightarrow (A, 0) \rightarrow (A, \varepsilon) \in H_M$$

And thus the output

$$O_F(A, \varepsilon) = \text{False}$$

.

The machine in fact will only accept integers that are *not* multiples of 3.

Definition 1.9 .

$$f'_n(w) = O_F(C_n)$$

A subset of Σ^* of a DFA that contains all inputs to which the output of the machine is True is called the *language* of the machine.

Definition 1.10 Regularity of Language. $L \subseteq \Sigma^*$ is regular if

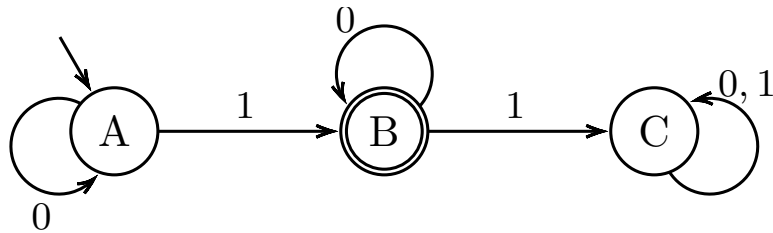
$$\exists \text{DFAM} \mid L(M) = L$$

Example 1.4

Given that $\varepsilon^* = \{\varepsilon\}$ and $\Sigma^* = \{\varepsilon, 1, 0, 10, 101, \dots\} = \{0, 1\}^*$, which of the following languages are regular?

- $L_1 = \{w \in \{0, 1\}^* \mid w \text{ is a power of } 2\}$, and
- $L_2 = \{w \in \{0, 1\}^* \mid w \text{ is a power of } 3\}$.

L_1 is regular while L_2 is not. A binary number that is a power of 2 consists of only one 1 and all other digits should be 0s. A DFA that recognizes the language would be

**Definition 1.11 Operations on Languages.**

Complement $L^C = \{w \in \Sigma^* \mid w \notin L\}$

Union $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \vee w \in L_2\}$

Intersection $L_1 \cap L_2 = \{w \mid w \in L_1 \wedge w \in L_2\}$

Concatenation $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$

Example 1.5: If L is regular, is L^C also regular?

Yes.

Proof of statement \mathbb{R} is closed under complement. Let $L \in \mathbb{R}$, prove $L^C \in \mathbb{R}$:

By definition,

$$\exists M = (Q, \Sigma, \delta, s, F) \text{ s.t. } L(M) = L.$$

Let $M' = (Q, \Sigma, \delta, s, F^C)$,

then $L(M') = L(M)^C = L^C$.

$L^C \in \mathbb{R}$ because $L^C = L(M')$. □

Example 1.6: $\forall L_1, L_2$

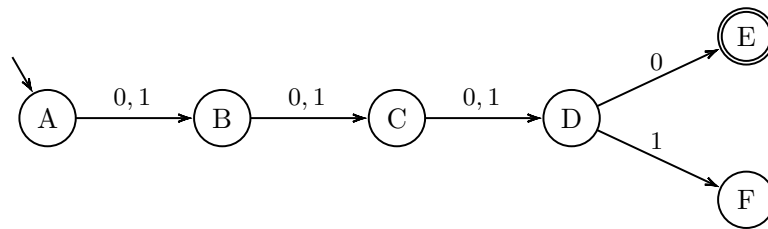
$$L_1 \in \mathbb{R} \vee L_2 \in \mathbb{R} \implies L_1 \cup L_2 \in \mathbb{R}$$

Yes, \mathbb{R} is closed under union.

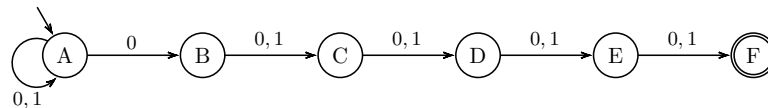
2 Nondeterministic Finite Automaton (NFA)

Example 2.1: How many states do you need?

$$L = \{ w \in \{0, 1\}^* \mid \text{the 5th digit of } w \text{ is } 0 \}$$



What if the digits are counted from the right, i.e. the last 5th digit?



Definition 2.1 NFA. An NFA is a 5-tuple

$$N = (Q, \Sigma, \delta, s, F)$$

where

- Q and Σ are finite sets
- $s \in Q, F \subseteq Q$
- $\delta: Q \times \Sigma_\epsilon^1 \mapsto \mathcal{P}(Q)$

Definition 2.2 Configurations of NFA.

$$\begin{aligned}
\text{Conf} &= Q \times \Sigma^* \\
I(w) &= (s, w) \\
H &= Q \times \{\epsilon\} \\
O(q, \epsilon) &= \begin{cases} \text{True}, & q \in F \\ \text{False}, & \text{else} \end{cases} \\
R &= \{ (q, aw) \mapsto (q', w) \mid \forall q \in Q, a \in \Sigma_\epsilon, w \in \Sigma^* \}
\end{aligned}$$

Definition 2.3 Language of NFA.

$$L(N) = \{ w \mid \exists \text{ accepting computation on input } w \}$$

Theorem 2.1 .

$$\begin{aligned}
\forall \text{DFAN} &= (Q, \Sigma, \delta, s, F), \\
\exists \text{DFAM} &= (Q', \Sigma, \delta', s', F') \text{ s.t. } L(N) = L(M)
\end{aligned}$$

Proof of Theorem 2.1.

$$\begin{aligned}
Q' &= \mathcal{P}(Q) \\
F' &= \{ A \subseteq Q \mid A \cap F \neq \emptyset \} \\
s' &= E(\{s\}) = \left\{ q \in Q \mid \exists s \xrightarrow{\epsilon} q_1 \xrightarrow{\epsilon} q_2 \xrightarrow{\epsilon} q_3 \cdots \xrightarrow{\epsilon} q \right\} \\
\delta'(A, a) &= E\left(\bigcup_{q \in A} \delta(q, a)\right)
\end{aligned}$$

□

¹ $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$