# CSE 105:
# Computation

---

*by* Liu Tan

# Contents

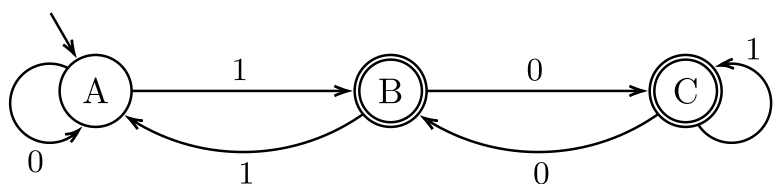
# 1 Deterministic Finite Automaton (DFA)

A machine consists different 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 read 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?

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

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 difit '0' makse the 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-turple*

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

*where*

$Q$ *is a finite set, for states*

$\Sigma$ *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: DFA table**

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 $\delta$ can be described by the table below.

| $\delta$ | 0 | 1 |
|---|---|---|
| $A$ | $A$ | $B$ |
| $B$ | $C$ | $A$ |
| $C$ | $B$ | $C$ |

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

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

*where* $\Sigma^*$ *is a set of string over* $\Sigma$*.*

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

**Definition 1.3** $\delta^*$**.**

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

*which is an inductive function defined as*

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

*where* $(q \in Q, a \in \Sigma, w \in \Sigma^*)$

## 1.2 Configurations of DFAs

**Definition 1.4 Configuration.**

$$Conf = Q \times \Sigma^*$$

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

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

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

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

**Definition 1.7 Machine Output.** *The output of a machine is a function that either "True" or "False."*

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

*defined as*

$$O_M(q, \varepsilon) = \begin{cases} \mathit{True}, & if \quad q \in F \\ \mathit{False}, & if \quad q \notin F \end{cases}$$

In summary :

- F $\subseteq$ Q
- s $\in$ Q
- $\varepsilon : Q \times \Sigma \mapsto Q$

Example 1.3: Example 1.1 as configurations

With input "10010" write in mathematical language, the confiuration of machine in Example 1.1:

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

And thus the out put

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

.

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

**Definition 1.8 .**

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

**Definition 1.9 n's State's Configuration.**

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

*e.g.*

$$I_M(w) \rightarrow_M C_1 \rightarrow_M C_2 \rightarrow_M \cdots \rightarrow_M \in H_M$$

for example

$$L(M) = \{w \in \Sigma^* \mid f_M(w) = \text{True}\}$$
$$L(M_1) \neq \Sigma^*$$
$$1001 \notin 3 \times \mathbb{Z}$$

## 1.3 Languages

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

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

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

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

Which means, a DFA could ***recognize*** L. In short, given a regular language, there always exist a DFA could be draw.

Notice that

- $\varepsilon$(small epsilon) = ***empty string***
- $\Sigma$(big Sigma) = ***alphabet set***
- $\varepsilon^* = \{\varepsilon\}$

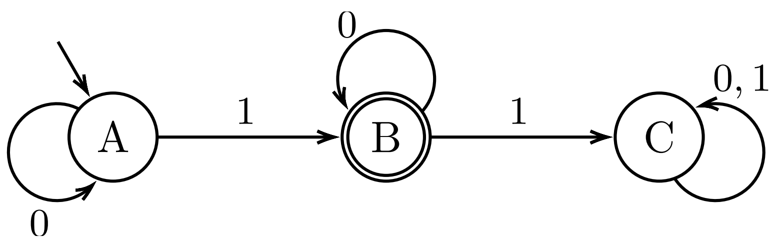
- $\Sigma^* = \{ \varepsilon, 1, 0, 10, 101, \cdots \} = \{ 0, 1 \}^*$,

**Example 1.4: Which of the follwing languages are regular?**

Given that and which of the following languages are regular?

- $L_1 = \{ w \in \{ 0, 1 \}^* \mid w$ is a power of 2 $\}$, and
- $L_2 = \{ w \in \{ 0, 1 \}^* \mid w$ 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** $L_1 \cap L_2 = \{ w \mid w \in L_1 \wedge \in L_2 \}$
**Concatenation** $L_1 \cdot L_2 = \{ w_1 \cdot w_2 \mid w \in L_1, w_2 \in L_2 \}$

**Theorem 1.1 .** *$\mathbb{R}$ is closed under complement.*

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

Yes.

*Proof of* **??**. 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 $M' = \left( Q, \Sigma, \delta, s, F^C \right)$,
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)

In a ***nondeterministic*** machine, several choices may exist for the next state at any point.

## 2.1 What is NFA

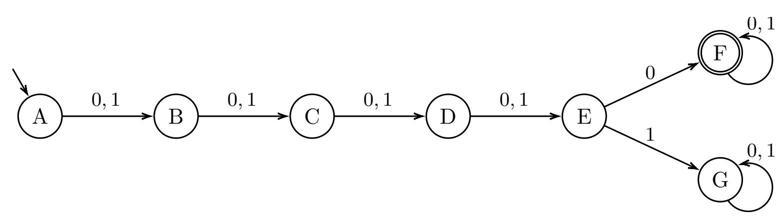
Nondeterminism is a generalization of determinism, so every deterministic finite automaton(DFA) is automatically a nondeterministic finit automaton(NFA). Notice the Difference between DFA figures and NFA's:

1. NFA may has more than one exiting arrow for symbols in the alphabet.
2. NFA may only only have arrows labeled with members of the alphabet but also $\varepsilon$. $(0, 1, \cdots, n)$ arrows may exit from each state with the label $\varepsilon$.

Example 2.1: How many states needed?

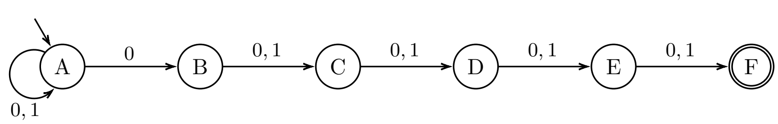
$$L = \{\, w \in \{0, 1\}^* \mid \text{ the } 5^{th} \text{ (from the left) of } w \text{ is } 0 \,\}$$

We will need 7 states




What if

$$L = \{\, w \in \{0, 1\}^* \mid \text{ the } 5^{th} \text{ from the } \textbf{\textit{r}}\text{ight) of } w \text{ is } 0 \,\}$$




Notice this is a **DFA**, state A has two exit arrows for 0.

**Definition 2.1 NFA.** *An NFA is a* $5$*-tuple*

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

*where*

- $Q$ *and* $\Sigma$ *are finite sets*
- $s \in Q$
- $F \subseteq Q$
- $\delta \colon Q \times \Sigma_\varepsilon$[1] $\mapsto \mathcal{P}(Q)$ [2]
  $\delta(A, \varepsilon) = \{ H \}$ $\delta(D, \varepsilon) = \emptyset$

As said, since a DFA ***is*** an NFA, the definition of NFA is simply a generalized version of DFA's. The difference between NFA and DFA regard as transition function $\delta$ is in NFA, $\delta$ maps to a set of states instead of exactly one state as of a DFA

## 2.2 Configurations of NFAs

**Definition 2.2 Configurations of NFA.**

$$\begin{aligned}
\text{Conf} &= Q \times \Sigma^* \\
I_M(w) &= (s, w) \\
H_M(w) &= Q \times \{ \varepsilon \} \\
O_M(q, \varepsilon) &= \begin{cases} \text{True}, & if \quad q \in F \\ \text{False}, & \textit{otherwise} \end{cases} \\
R &= \{ (q, aw) \mapsto (q', w) \mid \forall q \in Q, a \in \Sigma_\varepsilon, w \in \Sigma^* \}
\end{aligned}$$

After reading the symbol, an NFA splits into multiple copies of itself and follows ***all*** the possibilities in parallel. If the next input symbol doesn't appear on any of the arrows exiting the state occupied by a copy of the machine, that copy of the machine dies, along with the branch of the computation associated with if.Finally, if ***any one*** of these copies of the machine is in an accept state at the end of the input, the NFA accepts the input string. That is to say, if at least one of these processes accepts, then the entire computation accepts.

[1] $\Sigma_\varepsilon = \Sigma \cup \{ \varepsilon \}$
[2] $\mathcal{P}(Q)$ = powerset of $Q$

**Definition 2.3 Computation.**

$$C_0, C_1, \ldots, C_n$$

*Computation is a sequence of Configurations, such that*

$$C_0 = I(w) \quad \forall i, (C_i, C_{i+1} \in \mathbb{R}) \quad [C_i \mapsto C_{i+1}], C_n \in H$$

an NFA is Accepting if $O(C_n) =$ True , Rejecting if $O(C_n) =$ False

**Definition 2.4 Language of NFA.**

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

# 3 NFA & DFA

According to Theorem 3.1, and NFA can be translated to a DFA. The method is to fully expand the NFA and draw out every branch of it, which is explained with more details in Theorem 3.

**Theorem 3.1 .**

$$\forall N = (\, Q, \Sigma, \delta, s, F\,),\ \exists \text{DFA} \quad M = (\, Q', \Sigma', \delta', s', F'\,) s.t. L(N) = L(M)$$

*Proof of Theorem 3.1.* Let $N = (\, Q, \Sigma, \delta, s, F\,)$ be the NFA recognizing some language $A$, we construct a DFA $M = (\, Q', \Sigma, \delta', s', F'\,)$ recognizing $A$.

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

□

## 3.1 Equivalence of NFAs and DFAs

A finite automaton(DFA $\Longleftrightarrow$ NFA)

1. Defining Models of computation.
2. testifying equivalence between models.

**Definition 3.1 Reverse.**

- *reverse of a string*

$$\mathrm{rev}\big((\,Q_1, Q_2, \cdots, Q_n\,)\big) = \big(\,Q_n, Q_(n-1), \cdots, Q_1\,\big).$$

- *reverse of a language*

$$\mathrm{rev}(L) = \{\,rev(w) \mid w \in L\,\}.$$

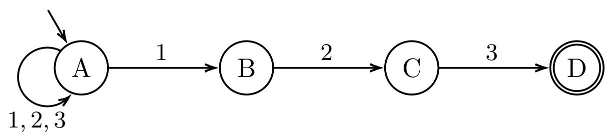



Figure 1: Example of NFA

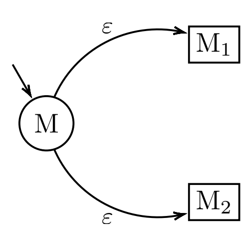
It is easy to find out that

$$\forall L \in \mathbb{R}, \mathrm{rev}(L) \in \mathbb{R}.$$[3]

**Example 3.1: Regular language is closed under union**

Recall Example 1.6, with NFA, we could proof Theorem 1.1 much more easier now.

*Proof of Theorem 1.1.* Let $M_1, M_2$ become NFA for $L_1$ and $L_2$. We build a NFA for $L_1 \cup L_2$ by simply adding a new initial state that transit to $s_1$ and $s_2$ with $\varepsilon$ arrows:




□

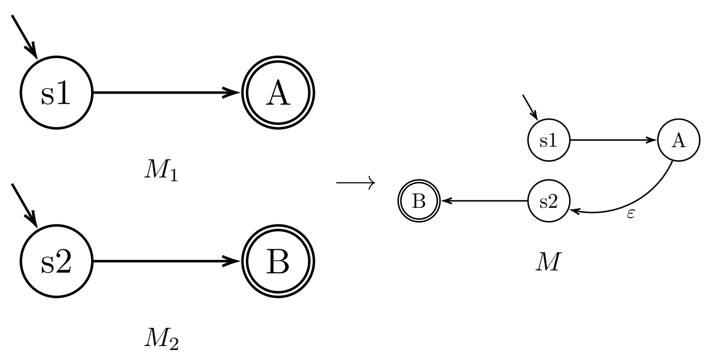
[3] $\mathbb{R}$ = Regular language

### Example 3.2: Regular language is closed under concatenation

*Proof.* Let $M_1, M_2$ become NFA for $L_1$ and $L_2$. We build a NFA for $L_1 \circ L_2$:

1. Assign M's start to be the start state of $M_1$ which is $s_1$ and
2. change the final states in $M_1$ to regular states and connect them to $s_2$ with additional $\varepsilon$ arrows that nondeterministically allow branching to $M_2$ whenevr $M_1$ is in an accept state, signifying that it has found an initial piece of the input that constitues a string in $L_1$
3. The accept states of $M$ are the accept states of $M_2$ only. only.

See the graph for visualization:




thus

$$L(M) = L(M_1) \circ L(M_2) = \{ wv \mid w \in L(M_1), v \in L(M_2) \}$$

$\square$