CSC 4890. Deterministic Finite Automata

$\boxed{\text{string}} \longrightarrow \boxed{\text{Finite Automaton}} \longrightarrow \boxed{\text{Reject or Accept}}$

Input Tape | output.

a,b in the same transition ? a or b.

For every state, there is a transition for every symbol in the alphabet.

Last state determines the outcome.

To accept a string: all the input string is scanned, and the last state is accepting.

To Reject a string: all the input string is scanned and the last state is non-accepting.

Deterministic Finite Automaton (DFA):

λ can be in languages, but not in alphabet.

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

Q: set of states, Σ: input alphabet $\boxed{\lambda \notin \Sigma}$

δ: transition function; q_0 = initial state; F: set of accepting states.

$\delta(q, x) = q'$: $q \xrightarrow{x} q'$, only a transition.

Extended Transition Function: $\delta^*(q, w) = q' \Rightarrow q \overset{w}{\rightsquigarrow} q'$

the resulting state after scanning string w from state q.

$\delta^*(q, \lambda) = q$. for any state q.

Language accepted by M: $L(M) = \{ w \in \Sigma^* : \delta^*(q_0, w) \in F \}$.

Language rejected by M: $\overline{L(M)} = \{ w \in \Sigma^* : \delta^*(q_0, w) \notin F \}$

Regular Language: A language is a regular language if some finite automaton recognizes it.

Nondeterministic Finite Automaton (NFA)

An NFA accepts a string: if there is a computation path of the NFA that accepts the string, i.e., all the input string is processed and the automaton is in an accepting state.

An NFA rejects a string: if there is no computation of the NFA that accepts the string.

For each computation path: i). All the input is consumed and the automaton is in a non-accepting state. OR

ii). The input cannot be consumed.

λ transition: automaton changes state, but input tape head does not move.

$$p_1 \xrightarrow{\lambda} p_2$$

Formal Definition of NFAs: $M = (Q, \Sigma, \delta, q_0, F)$

Q: set of states; Σ input alphabet, $\lambda \notin \Sigma$. δ: transition function.

q_0: initial state; F: accepting states.

$\delta(q, x)$: resulting states reached by following one transition with input symbol x

Extended Transition Function: δ^*.

For any state q, $q \in \delta^*(q, \lambda)$

The Language of an NFA is $L(M) = \{w_1, w_2, \cdots, w_n\}$.

where for each w_m: $\delta^*(q_0, w_m) = \{q_i, \cdots, q_k, \cdots, q_j\}$.

and there is some $q_k \in F$.

NFAs accept the Regular Languages:

Equivalence of Machines: Machine M_1 is equivalent to Machine M_2

if $L(M_1) = L(M_2)$

Theorem:

Languages accepted by NFAs are Regular Languages.

NFAs and DFAs have the same computation power, they accept the same set of languages.

Proof: we need to show:

$$\left\{\begin{array}{c}\text{Languages accepted} \\ \text{by NFAs}\end{array}\right\} \supseteq \{\text{Regular languages}\}.$$

and
$$\left\{\begin{array}{c}\text{Languages accepted} \\ \text{by NFAs}\end{array}\right\} \subseteq \{\text{Regular Languages}\}.$$

step 1: to prove {Languages accepted by NFAs} $\supseteq$ {regular languages}.

Since every DFA is trivially a NFA, any language L accepted by a DFA is also accepted by a NFA.

step 2: to prove {Languages accepted by NFAs} $\subseteq$ {Regular Languages}

Since Any NFA can be converted to an equivalent DFA, any language L accepted by a NFA is also accepted by a DFA.

Conversion of NFA to DFA.

General Conversion Procedure: $\left\{\begin{array}{l}\text{input: an NFA } M. \\ \text{output: an equivalent DFA } M' \text{ with } L(M)=L(M'\end{array}\right.$

step 1: Initial state of NFA: q_0, $\delta^*(q_0, \lambda) \Rightarrow$ Initial state of DFA. $\{q_0, \dots\}$

step 2: For every DFA's state $\{q_i, q_j, \cdots, q_m\}$, compute in the NFA:

$$\left.\begin{array}{l}\delta^*(q_i, a) \\ \cup\, \delta^*(q_j, a) \\ \cdots\end{array}\right\} = \{q_k, q_l, \cdots, q_n\}$$

Then, add transition to DFA:

$$\delta(\{q_i, q_j, \cdots, q_m\}, a) = \{q_k, q_l, \cdots, q_n'\}$$

Step 3: Repeat step 2 for every state in DFA and symbols in alphabet until no more states can be added in the DFA.

Step 4: For any DFA state $\{q_i, q_j, \cdots, q_m\}$, if some q_j is accepting state in NFA. Then $\{q_i, q_j, \cdots, q_m\}$ is accepting in DFA.

Lemma: if we convert NFA M to DFA M', then the two automata are equivalent: $L(M) = L(M')$.

Proof: We need to show $L(M) \subseteq L(M')$ and $L(M) \supseteq L(M')$.

i). To show $L(M) \subseteq L(M')$, we just need to prove:

$$w \in L(M) \Rightarrow w \in L(M')$$

$L(M) \supseteq L(M')$ can be proved in a similar way.

Proof by induction on $|v|$.

Induction Basis: $|v| = 1$, $v = a_1$.

NFA: $M: \rightarrow q_0 \xrightarrow{a_1} q_i$

DFA: $M': \rightarrow \underset{\{q_0, \cdots\}}{\bigcirc} \xrightarrow{a_1} \underset{\{q_i, \cdots\}}{\bigcirc}$.

is true by construction of M'.

Induction hypothesis: $1 \le |v| \le k$.
$v = a_1 a_2 \cdots a_k$.

Suppose that the following hold:

NFA $M: \rightarrow q_0 \xrightarrow{a_1} q_i \xrightarrow{a_2} q_j \rightsquigarrow \xrightarrow{a_k} q_d$

DFA $M': \rightarrow \underset{\{q_0, \cdots\}}{\bigcirc} \xrightarrow{a_1} \{q_i, \cdots\} \xrightarrow{a_2} \{q_j, \cdots\} \rightsquigarrow \xrightarrow{a_k} \{q_d, \cdots\}$

Induction Step: For $|v| = k+1$, $v = a_1 a_2 \cdots a_k a_{k+1} = v' \cdot a_{k+1}$

This is true by by construction of $\begin{cases} \text{NFA } M: v' \xrightarrow{a_{k+1}} q_e \\ \text{DFA } M': v' \xrightarrow{a_{k+1}} \{q_e, \cdots\} \end{cases}$