# FORMAL LANGUAGES AND AUTOMATA 22AIE302

Ayswarya R Kurup

# Equivalence of NFA and DFA

- Two machines / automata are equal if they recognize the same language
- Converting an NFA to an equivalent DFA
- Two finite accepters, $M_1$ and $M_2$, are equivalent, if they both accept the same language

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

Every NFA has an equivalent DFA

# Equivalence of NFA and DFA..

Theorem

Let $L$ be the language accepted by a nondeterministic finite accepter
$M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$
Then there exists a deterministic finite accepter
$M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that $L(M_N) = L(M_D)$

Two steps in proof

Step 1. Define the DFA $M_D$
Step 2. Prove $L(M_N) = L(M_D)$

Proof

Procedure: Use the procedure **nfa-to-dfa** to construct the transition graph $G_D$ for $M_D$.
$G_D$ should have the following

- Every vertex must have exactly $|\Sigma|$ outgoing edges, each labeled with a different element of $\Sigma$.

# Proof..

**procedure: nfa-to-dfa**

1. Create a graph $G_D$ with vertex $\{q_0\}$. Identify this vertex as the initial vertex.
2. Repeat the following steps until no more edges are missing.
   1. Take any vertex $\{q_i, q_j, ..., q_k\}$ of $G_D$ that has no outgoing edge for some $a \in \Sigma$
      Compute $\delta_N^*(q_i, a), \delta_N^*(q_j, a), ..., \delta_N^*(q_k, a)$
      - If $\delta_N^*(q_i, a) \cup\ \delta_N^*(q_j, a) \cup ..\ \delta_N^*(q_k, a) = \{q_l, q_m, ..q_n\}$ create a vertex for $G_D$ labeled $q_l, q_m, ..., q_n$ if it does not already exist. Add to $G_D$ an edge from $\{q_i, q_j, ..., q_k\}$ and label it with $a$.
3. Every state of $G_D$ whose label contains any $q_f \in F_N$ is identified as a final vertex.
4. If $M_N$ accepts $\lambda$, the vertex $\{q_0\}$ in $G_D$ is also made a final vertex.

# Proof..

1. $L(M_N) \subseteq L(M_D)$
   - We need to prove that any string accepted by $M_N$ is also accepted by $M_D$.

# Proof..

1. $L(M_N) \subseteq L(M_D)$
   - We need to prove that any string accepted by $M_N$ is also accepted by $M_D$.
   - Let $w$ be a string accepted by $M_N$.

# Proof..

1. $L(M_N) \subseteq L(M_D)$
   - We need to prove that any string accepted by $M_N$ is also accepted by $M_D$.
   - Let $w$ be a string accepted by $M_N$.
   - That is there is a sequence of transitions in $M_N$ starting from the initial state $q_0$, reading the symbols of $w$, and ending in a state in $F_N$.

# Proof..

1. $L(M_N) \subseteq L(M_D)$
    - We need to prove that any string accepted by $M_N$ is also accepted by $M_D$.
    - Let $w$ be a string accepted by $M_N$.
    - That is there is a sequence of transitions in $M_N$ starting from the initial state $q_0$, reading the symbols of $w$, and ending in a state in $F_N$.
    - By the construction of $M_D$, there exists a corresponding sequence of transitions in $M_D$ that starts from $\{q_0\}$, reads the same symbols of $w$, and ends in a state in $F_D$.

# Proof..

1. $L(M_N) \subseteq L(M_D)$
   - We need to prove that any string accepted by $M_N$ is also accepted by $M_D$.
   - Let $w$ be a string accepted by $M_N$.
   - That is there is a sequence of transitions in $M_N$ starting from the initial state $q_0$, reading the symbols of $w$, and ending in a state in $F_N$.
   - By the construction of $M_D$, there exists a corresponding sequence of transitions in $M_D$ that starts from $\{q_0\}$, reads the same symbols of $w$, and ends in a state in $F_D$.
   - Therefore, $w$ is also accepted by $M_D$, and $L(M_N) \subseteq L(M_D)$.

# Proof..

2. $L(M_D) \subseteq L(M_N)$
   - We need to prove that any string accepted by $M_D$ is also accepted by $M_N$.

# Proof..

2. $L(M_D) \subseteq L(M_N)$
   - We need to prove that any string accepted by $M_D$ is also accepted by $M_N$.
   - Let $w$ be a string accepted by $M_D$.

# Proof..

2. $L(M_D) \subseteq L(M_N)$
   - We need to prove that any string accepted by $M_D$ is also accepted by $M_N$.
   - Let $w$ be a string accepted by $M_D$.
   - That is there is a sequence of transitions in $M_D$ starting from $\{q_0\}$, reading the symbols of $w$, and ending in a state in $F_D$.

# Proof..

2. $L(M_D) \subseteq L(M_N)$
   - We need to prove that any string accepted by $M_D$ is also accepted by $M_N$.
   - Let $w$ be a string accepted by $M_D$.
   - That is there is a sequence of transitions in $M_D$ starting from $\{q_0\}$, reading the symbols of $w$, and ending in a state in $F_D$.
   - By the construction of $M_D$, this state in $F_D$ corresponds to a subset of states in $Q_N$, denoted as $S$.
   - $S \cap F_N \neq \emptyset$, there exists at least one state in $S$ that is in $F_N$.

2. $L(M_D) \subseteq L(M_N)$
   - We need to prove that any string accepted by $M_D$ is also accepted by $M_N$.
   - Let $w$ be a string accepted by $M_D$.
   - That is there is a sequence of transitions in $M_D$ starting from $\{q_0\}$, reading the symbols of $w$, and ending in a state in $F_D$.
   - By the construction of $M_D$, this state in $F_D$ corresponds to a subset of states in $Q_N$, denoted as $S$.
   - $S \cap F_N \neq \emptyset$, there exists at least one state in $S$ that is in $F_N$.
   - Therefore, there exists a sequence of transitions in $M_N$ that starts from $q_0$, reads the symbols of $w$, and ends in a state in $F_N$.

# Proof..

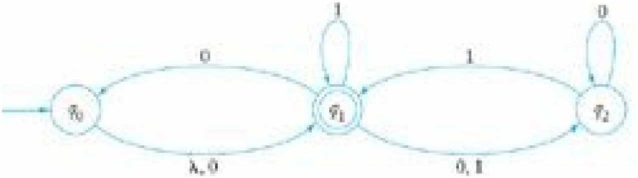
2. $L(M_D) \subseteq L(M_N)$
   - We need to prove that any string accepted by $M_D$ is also accepted by $M_N$.
   - Let $w$ be a string accepted by $M_D$.
   - That is there is a sequence of transitions in $M_D$ starting from $\{q_0\}$, reading the symbols of $w$, and ending in a state in $F_D$.
   - By the construction of $M_D$, this state in $F_D$ corresponds to a subset of states in $Q_N$, denoted as $S$.
   - $S \cap F_N \neq \emptyset$, there exists at least one state in $S$ that is in $F_N$.
   - Therefore, there exists a sequence of transitions in $M_N$ that starts from $q_0$, reads the symbols of $w$, and ends in a state in $F_N$.
   - Thus, $w$ is also accepted by $M_N$, and $L(M_D) \subseteq L(M_N)$.

# Examples

Convert the given nfa into an equivalent deterministic machine.

# Examples

Convert the given nfa into an equivalent deterministic machine.



# Reduction of the Number of States in Finite Automata

Process of simplifying the number of states in an automaton without affecting the language accepted by the automaton

- Unreachable States
- Equivalent States
- $\epsilon$ - Closure Reduction