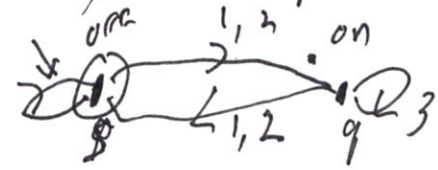
CSE 396 Lecture Tue Feb 9 Spr 2016 (1)

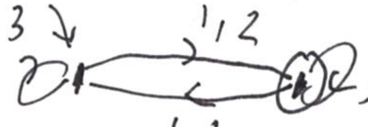
Demo of "Dragonslayer" DFA] using the "Turing Kit"

<u>Theorem</u>: If a DFA $M = (Q, \Sigma, \delta, s, F)$ accepts a language $L$, then the DFA $M' = (Q, \Sigma, \delta, s, \underline{Q \setminus F})$ accepts $\sim L$.

<u>Example</u>: $L = L_A = \{x \in \{1,2,3\}^*$ : x leaves Light A off$\}$.


$M =$ 


$Q = \{s, q\}$ $F = \{s\}$

$\sim L_A = \{x \in \{1,2,3\}^*$ : x leaves light A <u>on</u>$\}$. $\underline{M'_A} =$ 


$F' = Q \setminus F = \{q\}$.

By the Cartesian Product theorem in recitations, if $L_1$ is accepted by a DFA $M_1$ and $L_2$ by $M_2$, then we can build a DFA $M_3$ s.t. $L(M_3) = L_1 \cap L_2$.

~~$M_3 = (Q_1 \times Q_2, \Sigma, (s_1, s_2), \delta_3$~~

$M_3 = (Q_1 \times Q_2, \Sigma, \delta_3, \overbrace{(s_1, s_2)}^{s_3}, F_3)$ $F_3 = \{(q_1, q_2) : q_1 \in F_1 \text{ and } q_2 \in F_2\}$ $= F_1 \times F_2$.

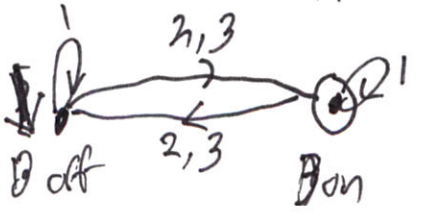
We could build $M_4 = ((Q_1 \times Q_2), \Sigma, \text{"}, (s_1, s_2), \{(q_1, q_2) : q_1 \in F_1 \ \underline{\text{or}}\ q_2 \in F_2\})$

Then $L(M_4) = L(M_1) \ \underline{\cup}\ L(M_2)$.

$M_5 = (-$ditto$ \text{———} (s_1, s_2), F_5)$ $F_5 = \{(q_1, q_2) : q_1 \in F_1 \ \underline{\text{XOR}}\ q_2 \in F_2\}$.

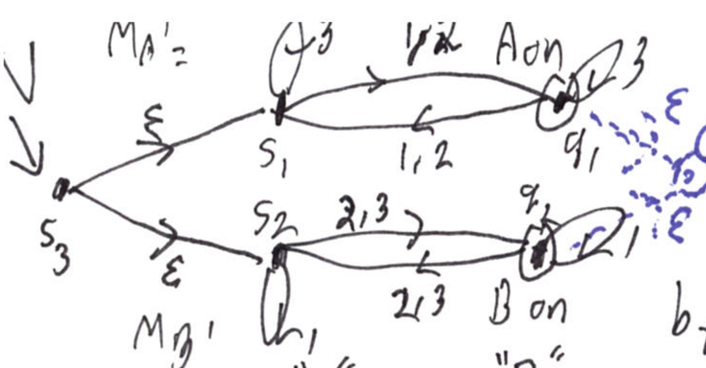
Then $L(M_5) = \{x : M_1$ accepts $x$ <u>XOR</u> $M_2$ accepts $x\}$

$= L(M_1) \triangle L(M_2)$ i.e. $L(M_1) \oplus L(M_2)$.

Do $M_B' =$ 


$\tilde{L} = \{x \in \{1,2,3\}^*$ : x leaves one or both lights on$\}$

$\tilde{L} = \tilde{L}_A \cup \tilde{L}_B$ Third idea: For $\cup$, try an <u>NFA</u>.




Build an NFA $N_3 = (Q_3, \Sigma, \delta_3, s_3, F_3)$ ②

such that $L(N_3) = L(M_A') \cup L(M_B')$:

by: $Q_3 = \{s_3\} \cup Q_1 \cup Q_2$

$F_3 = F_1 \cup F_2 = \{q_1, q_2\}$

$\delta_3 = \delta_1 \cup \delta_2 \cup \{(s_3, \varepsilon, s_1), (s_3, \varepsilon, s_2)\}$

FA: $\delta : \overbrace{(Q \times \Sigma)}^{\text{"A"}} \to \overbrace{Q}^{\text{"B"}}$

relation: $\delta \subseteq A \times B \equiv \underline{Q \times \Sigma \times Q}$

FA: relation is a function.

FA: general relation.

Formal Definition of NFA (with ε-arcs)

Defn: A nondeterministic finite automaton (NFA) is a 5-tuple $N = (Q, \Sigma, \delta, s, F)$ where:

- $Q$ is a finite set of states
- $\Sigma$ is the input alphabet
- $s$, a member of $Q$, is the start state
- $F$, a subset of $Q$, comprise the final states

and $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$.

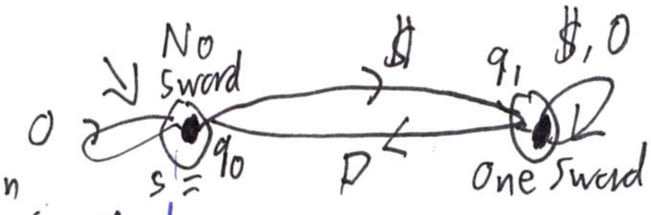
A DFA "Is-A" NFA in which
- No instructions have ε, and
- The relation $\delta \subseteq Q \times \Sigma \times Q$ defines a function with domain all of $Q \times \Sigma$ and range $\subseteq Q$.

Typical instruction: $(p, c, q)$ $c \in \Sigma$

or $(p, \varepsilon, q)$ $p = q$ allowed!


- An NFA is a DFA if for all $q \in Q$ and $c \in \Sigma$ there is exactly one instruction $(q, c, r) \in \delta$ where $r \in Q$. (And no instructions have ε.)

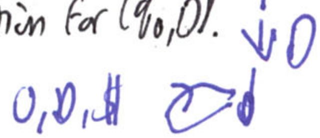
Is this a DFA?




$\Sigma = \{0, \$, D\}$

$L = \{x \in \Sigma^* :$ Between any two D's there is at least one \$ and at least one \$ before a D$\}$

No: lacks an instruction for $(q_0, 0)$.




Formally this diagram needs to be "completed" by adding a dead state. Then we can complement the machine.

Defn: A computation path that processes a string $x$ is a sequence $(q_0, w_1, q_1, w_2, q_2, \dots, q_{m-1}, w_m, q_m)$ such that

- for all $j$, $1 \le j \le m$, $(q_{j-1}, w_j, q_j) \in \delta$
- the string $w_1 \cdot w_2 \cdots w_m$ equals $x$.

Then we also say that $N$ can process $x$ from state $q_0$ to state $q_m$

Formally, $L(N) = \{x \in \Sigma^* : N$ can process $x$ from $s$ to a state in $F\}$.

Example: $x = 13$ (leaves both lights on)

Path: $(s_3, \epsilon, s_1, 1, q_1, 3, q_1)$ $|x| = n = 2$ but $m = 3$

$x = \epsilon \cdot 1 \cdot 3$

Another Acc Path: $(s_3, \epsilon, s_2, 1, s_2, 3, q_2)$

With a DFA, always $m = n = |x|$, and every step has just one option

[Note Added: We will use the concept in cases where "$q_0$" is not the start state. Indeed for any $p, q \in Q$ we can define

$$L_{pq} = \{x \in \Sigma^* : N \text{ can process } x \text{ from } p \text{ to } q\}.$$

So $L(N) = \bigcup_{q \in F} L_{sq}$. This again is why I like to use separate notation "s" from "$q_0$" for the start state.]