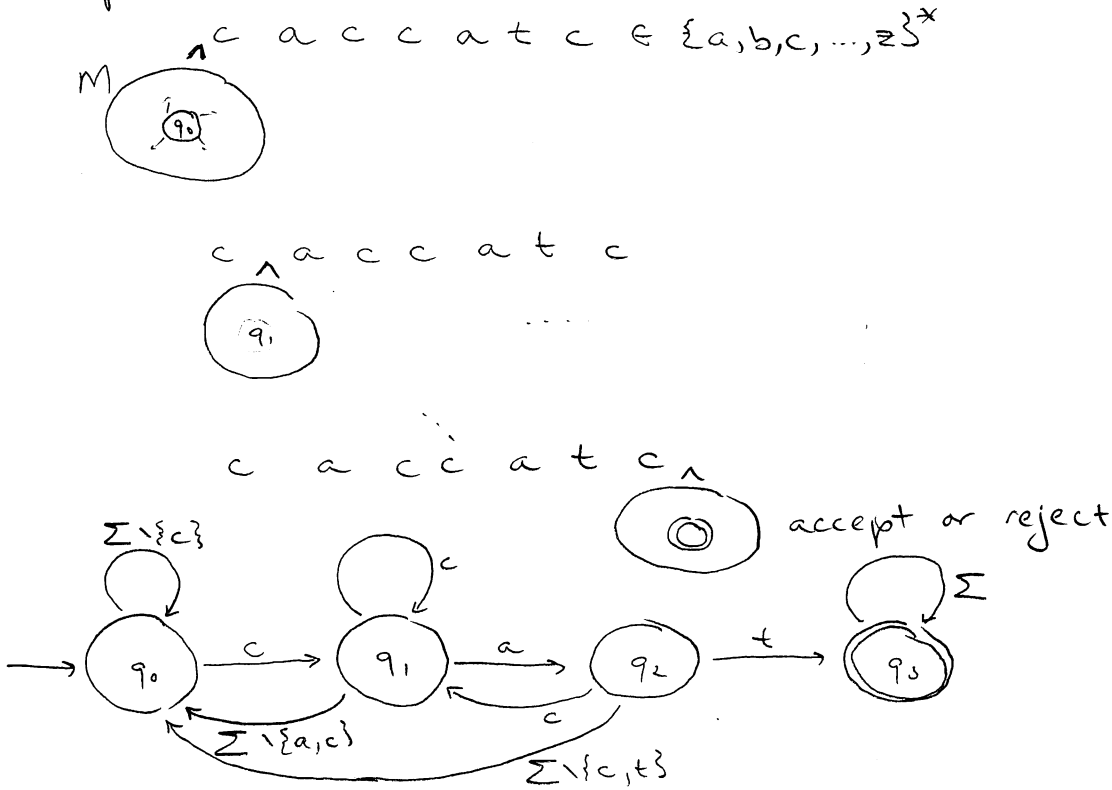
1/6/11 CS 360 Lecture 2: Deterministic & Nondeterministic finite automata

Admin note: Webpage has course info, proof methodology & homework 1

Recall: alphabet Σ, Σ^j = strings of length j, $\Sigma^0 = \{\varepsilon\}$,
$\Sigma^* = \{\text{set of all strings over } \Sigma\} = \bigcup_{j \geq 0} \Sigma^j$.

Deterministic finite automaton (DFA) = "deterministic machine with a finite amount of memory"

Example:



State	Remaining input	
q_0	cacccatc	
q_1	accatc	
q_2	ccatc	
q_1	catc	
q_1	atc	
q_2	tc	
q_3	c	
q_3	ε	→ accept

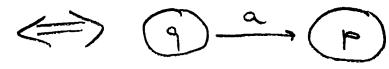
Formally,

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

- Q = finite set of states
- Σ = input alphabet
- q_0 = initial state $\in Q$
- F = set of final (accepting) states $\subseteq Q$
- δ = transition function: $Q \times \Sigma \to Q$ (input symbols)

eg.

state \ input	a	c	t	$\Sigma - \{a,c,t\}$
q_0	q_0	q_1	q_0	q_0
q_1	q_2	q_1	q_0	q_0
q_2	q_0	q_1	q_3	q_0
q_3	q_3	q_3	q_3	q_3

$\delta(q,a) = p \iff$ 

Extended transition function

δ : single step $Q \times \Sigma \to Q$

$\hat{\delta}$ sequence of steps $Q \times \Sigma^* \to Q$

$\hat{\delta}(q,x)$ means that the machine starting at state q, will go to state $\hat{\delta}(q,x)$ after reading x

eg. $\hat{\delta}(q_0, caccatc) = q_3$

- defined recursively, ie. by induction in the input length

Base case: $\hat{\delta}(q,\varepsilon) = q \quad \forall q$

Induction: $\hat{\delta}(q, ax) = \hat{\delta}(\delta(q,a), x) \quad \forall q \in Q, a \in \Sigma, x \in \Sigma^*$

Claim: $\forall q \in Q, a \in \Sigma, \hat{\delta}(q,a) = \delta(q,a)$

pf: $\hat{\delta}(q,a) = \hat{\delta}(q,a\varepsilon) = \hat{\delta}(\delta(q,a),\varepsilon) = \delta(q,a)$. □

Exercise: $\forall q \in Q, x,y \in \Sigma^*, \hat{\delta}(q,xy) = \hat{\delta}(\hat{\delta}(q,x),y)$.

Definition: Language of a DFA

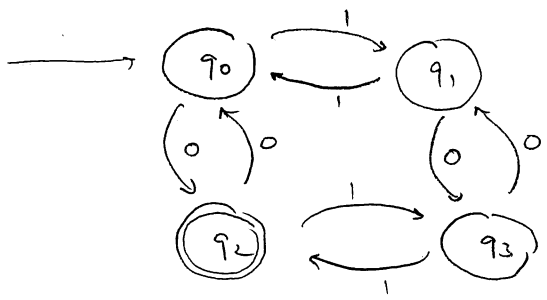
$L(M) = \{x \in \Sigma^* : \hat{\delta}(q_0,x) \in F\}$

$= \{$set of all input strings accepted by $M\}$

Example:

L(our first example) = {all strings containing cat as a substring}

Question: What is L(M) for:



L(M) = {all binary strings with an odd number of 0s and an even number of 1s}

Why?

Let $n_a(x)$ = # of occurrences of letter a in string x

Claim:

$$\hat{\delta}(q_0, x) = \begin{cases} q_0 & \text{if } (n_0(x) \bmod 2,\ n_1(x) \bmod 2) = (0,0) \\ q_1 & \text{" } (0,1) \\ q_2 & \text{" } (1,0) \\ q_3 & \text{" } (1,1) \end{cases}$$

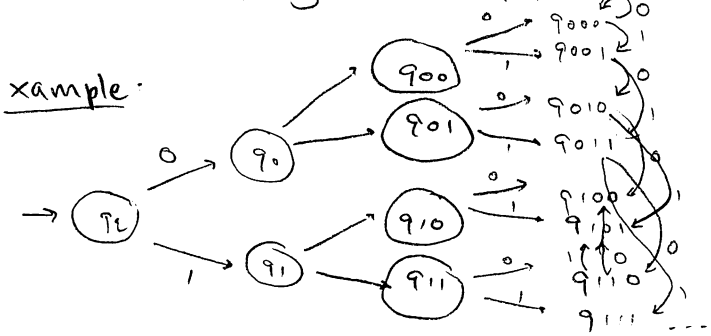
Proof: Holds for $|x| = 0$, i.e. $x = \varepsilon$.

Assume by induction it holds for $x \in \Sigma^n$.

Consider $y = xa \in \Sigma^{n+1}$

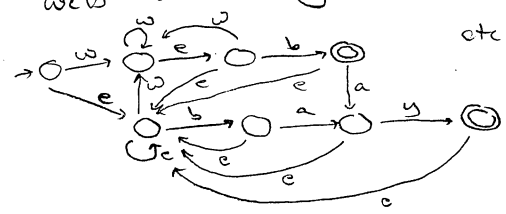
$\hat{\delta}(q_0, y) = \delta(\hat{\delta}(q_0, x), a)$ ← check eight cases... □

Example:



DFA with memory it remembers last 3 chars.

Example: Give a DFA that accepts any string ending with "web" or "ebay" as a substring.



etc.

-or just remember the last four characters

$Q = \{q_x : x \in \Sigma^*, |x| \le 4\}$

$\delta(q_x, a) = \begin{cases} q_{xa} & \text{if } |x| \le 3 \\ q_{x_2 \cdots x_n a} & \text{if } |x| = n \ge 4 \end{cases}$

DFA: Since $\delta(q,a)$ is unique,
for any input, the execution is predictable & repeatable

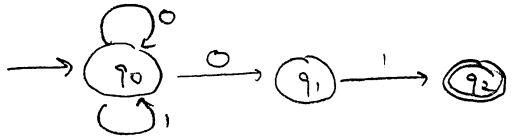
Nondeterministic finite automaton (NFA)

$\delta(q,a)$ is a set of states
- could have no choices, or several

$\Rightarrow$ (q) can have multiple arrows leaving it with same label
- allows for "guessing" the right action

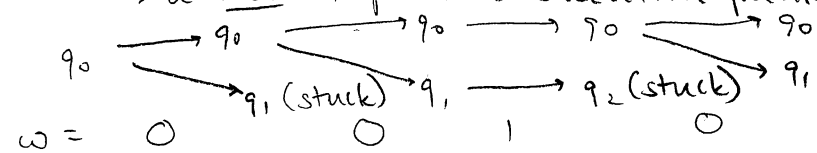
Example:

$L_{01} = \{w \mid w \text{ ends in } 01\}$

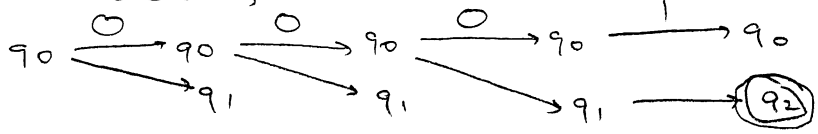


- it guesses when the end is coming, then looks for 01

$\Rightarrow$ a tree of possible execution paths



for $w = 0001$,



Acceptance: when at least one of the many execution paths completes in a final states

Rejection: when all paths either end at a non-final state or get stuck

Formally: $N = (Q, \Sigma, \delta, q_0, F)$

where now $\delta: Q \times \Sigma \to 2^Q$ = {set of all subsets of Q}

	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\emptyset$

$\hat{\delta}(q, w)$ = { states reachable from q on input w }

eg. $\hat{\delta}(q_0, 001) = \{q_0, q_2\}$

$\hat{\delta}(q_0, 000) = \{q_0, q_1\}$

defined by, For all $q \in Q$, $x \in \Sigma^*$, $a \in \Sigma$,

$\hat{\delta}(q, \varepsilon) = \{q\}$

$\hat{\delta}(q, xa) = \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a)$

eg. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 00) = \bigcup_{p \in \hat{\delta}(q_0,0)} \delta(p, 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$

$= \{q_0, q_1\} \cup \emptyset$

$= \{q_0, q_1\}$

$\hat{\delta}(q_0, 001) = \bigcup_{p \in \hat{\delta}(q_0,00)} \delta(p, 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$

$= \{q_0, q_2\}$

Language $N = (Q, \Sigma, \delta, q_0, F)$

$L(N) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset \}$.

Theorem: For any DFA M with $L(M)$ language, there exists an NFA N with $L(N) = L(M)$.

Proof: Obvious, since a DFA is a special case of an NFA with $|\delta(q, a)| = 1$ always. □

Theorem: For any NFA N, there exists a DFA M with $L(M) = L(N)$.