COMS3261: Computer Science Theory

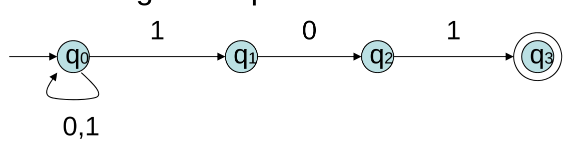
Spring 2013

Mihalis Yannakakis

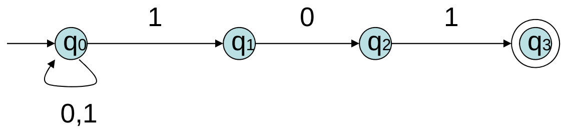
Lecture 3, 9/11/13

Nondeterministic Finite Automaton

- A = (Q, Σ, δ, q_0, F)
- Only difference that transition function $\delta : Q \times \Sigma \to 2^Q = \mathcal{P}(Q)$
  i.e., for each q in Q, a in Σ, $\delta(q,a) \subseteq Q$ is a set of 0, 1 or more states
- Alternatively (equivalently), can represent it as a *transition relation* R ={(q,a,p) | $p \in \delta(q,a)$ }
- Transition Diagram:
  Nodes = States
  Labeled edges = tuples of transition relation



Transition table representation



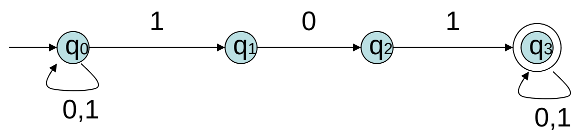
	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\emptyset$
q_2	$\emptyset$	$\{q_3\}$
$* q_3$	$\emptyset$	$\emptyset$

Computations, Acceptance

- Computation (run) of NFA on input $x = a_1 \ldots a_n$:
  sequence of states (not necessarily distinct) starting with initial state: $q_0\ q_1 \ldots q_n$ such that $q_i \in \delta(q_{i-1}, a_i)$, for all $i=1,\ldots,n$
  = path in transition diagram starting from q_0 with label x
  (label of path = sequence of labels of the edges)
- Accepting computation (run, path) = path from start state q_0 to a state in F.
- Input string x is accepted by A iff there is an accepting computation on input x, i.e. there is a path from q_0 to a node in F labeled by x.
- Language of A, L(A) = $\{ x \in \Sigma^* \mid x \text{ is accepted by A} \}$
  = set of labels of all accepting paths

Example NFA: Substring

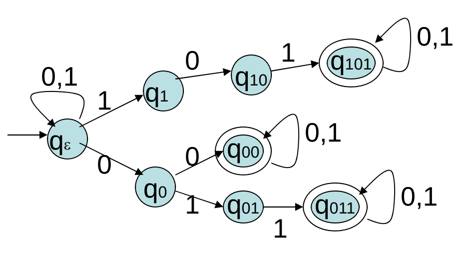
- L = set of inputs that contain 101 as a substring



Example NFA: Substrings

- L = set of inputs that contain as a substring

  00 or 011 or 101



Extension of transition function to strings

- Can extend δ to a function $\delta\text{^}$ from $Q\times\Sigma^*$ to 2^Q :

  Inductive definition:

  Basis: $\delta\text{^}(q,\varepsilon) = \{q\}$

  Induction: $\delta\text{^}(q,xa) = \bigcup_{p\in\delta\text{^}(q,x)} \delta(p,a)$, for $x\in\Sigma^*$, $a\in\Sigma$

  i.e., if $\delta\text{^}(q,x)=\{p_1,\ldots,p_k\}$ then $\delta\text{^}(q,xa)=\delta(p_1,a)\cup\ldots\cup\delta(p_k,a)$

Note: $\delta\text{^}(q,x)$= set of states p such that there is a path from q to p labeled x

(Can prove formally by induction from the definitions)

Language $L(A) = \{ x \in\Sigma^* \mid \delta\text{^}(q_0,x) \cap F \neq \emptyset \}$

Implementation of NFA

- Cannot implement nondeterminism directly.

- To test if an input x is in the language of NFA A

  Scan x keeping track of *set of states S* the NFA is in.

  Initialize: $S=\{q_0\}$

  while input is not finished

  { read next symbol, say a;

  update set S of states: $S = \bigcup_{q\in S} \delta(q,a)$ }

  if final set $S \cap F \neq \emptyset$ then accept else reject

Complexity: time $O(|x|\ |Q|^2)$, space $O(|Q|)$

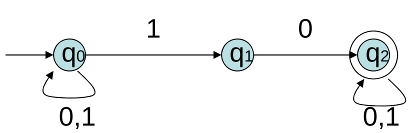
Equivalence of NFA and DFA

- Theorem: For every NFA N there is an equivalent DFA D, i.e., one that accepts the same language: L(N) = L(D)
- Constructive proof: Subset Construction
- Given NFA N = $(Q_N, \Sigma, \delta_N, q_0, F_N)$,
  Construct DFA D = $(Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$

- $Q_D = 2^{Q_N}$ = set of subsets of Q_N
- $\delta_D(S,a) = \bigcup_{q \in S} \delta_N(q,a)$ [extension of δ to *sets* of states]
- $F_D = \{ S \in Q_D \mid S \cap F_N \neq \varnothing \}$

Example

NFA N



DFA D

	0	1
$\varnothing$	$\varnothing$	$\varnothing$
$\rightarrow\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_1\}$	$\{q_2\}$	$\varnothing$
* $\{q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_0,q_1\}$	$\{q_0,q_2\}$	$\{q_0,q_1\}$
* $\{q_0,q_2\}$	$\{q_0,q_2\}$	$\{q_0, q_1,q_2\}$
* $\{q_1,q_2\}$	$\{q_2\}$	$\{q_2\}$
* $\{q_0,q_1,q_2\}$	$\{q_0,q_2\}$	$\{q_0,q_1,q_2\}$

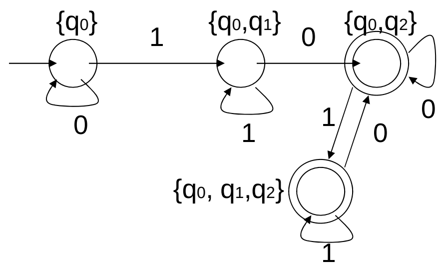
Notation: We can give the states any names we want,

eg. S1,…, S8 or A,..,H, or anything else

Reachable NFA

- Some states may be useless: We only need the states that are reachable from the start state $\{q_0\}$.
- Can construct the DFA incrementally (as in graph search) from start state $\{q_0\}$.

	0	1
$\rightarrow\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0,q_1\}$	$\{q_0,q_2\}$	$\{q_0,q_1\}$
$*\ \{q_0,q_2\}$	$\{q_0,q_2\}$	$\{q_0, q_1,q_2\}$
$*\ \{q_0,q_1,q_2\}$	$\{q_0,q_2\}$	$\{q_0,q_1,q_2\}$



Proof of NFA $\rightarrow$ DFA translation

Show by induction on the length of an input string *w* the following

- Claim: $\hat{\delta}_N(q_0, w) = \hat{\delta}_D(\{q_0\}, w)$

The claim implies the correctness of the translation:

For every string *w*, *w* is accepted by the NFA N iff

$$\hat{\delta}_N(q_0, w) \cap F_N \neq \emptyset$$

$$\Leftrightarrow \hat{\delta}_D(\{q_0\}, w) \cap F_N \neq \emptyset$$

$$\Leftrightarrow \hat{\delta}_D(\{q_0\}, w) \in F_D$$

$\Leftrightarrow$ *w* is accepted by the DFA D

Proof of Claim $\hat{\delta}_N(q_0, w) = \hat{\delta}_D(\{q_0\}, w)$

- Basis: $w = \varepsilon$. By definition of extension of δ functions to strings: both sides = $\{q_0\}$.

- Induction step: w =xa for some $x \in \Sigma^*$, $a \in \Sigma$

Induction hypothesis says: $\hat{\delta}_N(q_0, x) = \hat{\delta}_D(\{q_0\}, x)$

$$\hat{\delta}_N(q_0, xa) = \bigcup_{p \in \hat{\delta}_N(q_0, x)} \delta_N(p, a) \quad \text{extension of } \delta_N \text{ to strings}$$

$$= \delta_D(\hat{\delta}_N(q_0, x), a) \quad \text{definition of } \delta_D$$

$$= \delta_D(\hat{\delta}_D(\{q_0\}, x), a) \quad \text{Induction hypothesis}$$

$$= \hat{\delta}_D(\{q_0\}, xa) \quad \text{extension of } \delta_D \text{ to strings}$$

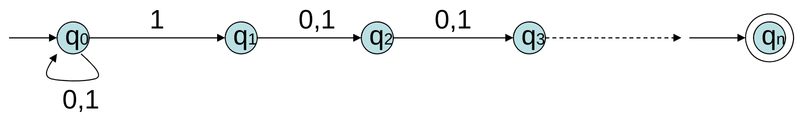
Complexity of construction

- #states of DFA $\leq 2^{(\text{\#states of NFA})}$
- Exponential: in general it could be that D could include all the subsets of states of N, even if we do the incremental construction.

- Could it be that another construction avoids exponential blowup always?
- No:
- Theorem: There are NFA such that the smallest equivalent DFA has exponentially larger size (#states) than the NFA

Complexity of NFA to DFA conversion

- Theorem: There are NFA such that the smallest equivalent DFA has exponentially larger size (#states)
- Example: Language Ln: set of binary strings such that the n-th symbol from the end is 1.
- Accepted by NFA with n+1 states



Fact: Any DFA that accepts the same language Ln must have at least 2^n states

(Try the incremental subset construction for n=3 and verify that this is the case; This does not prove it in itself: must argue this holds for *every equivalent DFA.*)

Proof of exponential blowup

Any DFA that accepts the same language Ln must have at least 2^n states

Intuition: must remember the last n symbols $\Rightarrow$ 2^n states

- Formally: Let DFA A accept Ln , consider all 2^n strings of length n.
- We will show that A is in different states after any two such strings,

  which implies that A has at least 2^n states by
- Pidgeonhole principle: If p pidgeons have less than p holes, then some pidgeons must share a hole.

Proof of exponential blowup

DFA A is in different states after any two strings of length n.

Proof by contradiction: Suppose that A is in same state after reading two (different) such strings x , y.

We' ll show A makes an error on some input.

- Let i be a position in which x, y differ, say x has 1, y has 0
- Append i-1 symbols, say 0's, at the end of x and y to get strings $x' = x0^{i-1}$, $y' = y0^{i-1}$
- The DFA A is in the same state after reading x and y, and the suffix from then on same $\Rightarrow$ same final state $\Rightarrow$ A either accepts both x',y' or rejects both x',y'
- But $x' \in Ln$ while $y' \notin Ln \Rightarrow$ A makes incorrect decision on either x' or y'