Lecture 1

Math Review

Relations:

Let A and B two sets. The **cartesian product** $A \times B$ is defined as

$$A \times B = \{ (a,b) \mid a \in A \text{ and } b \in B \}$$

e.g. $A = \{1, 2\}$ $B = \{a, b\}$

$$A \times B = \{ (1,a), (1,b), (2,a), (2,b) \}$$

$B \times A = ?$

e.g. $A = \{1, 2\}$ $B = \{a, b, c\}$

$A \times B = ?$

A **relation** R between two sets A and B is a **subset** of $A \times B$.

e.g. $A = \{1, 2\}$ $B = \{a, b\}$

$$A \times B = \{ (1,a), (1,b), (2,a), (2,b) \}$$

"binary relations"

$R_1 = \emptyset \subseteq A \times B$

$R_2 = \{(1,a)\}$ $R_3 = A \times B$

Exercise: $A \times B = B \times A$?

$A \times A = ?$

$A \times A \times A = ?$ $\quad A \times B \times C = ?$

$A \times A \times A \times A = ?$

- Find relations between B and A?
- How many relations are there between A and B?

A function f from a set A to a set B, $f: A \to B$, is a relation with the property that every element of A is mapped to exactly one element of B.

e.g. $A = \{1, 2\}$ $\quad B = \{a, b\}$

$A \times B = \{(1,a), (1,b), (2,a), (2,b)\}$

$R = \phi$ .. not a function

$R = A \times B$ .. "

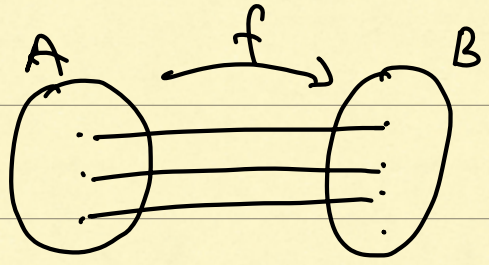
not 1-1 $\quad R = \{(1,b), (2,b)\}$ ✓

1-1 $\quad R = \{(1,b), (2,a)\}$ ✓ onto

Exercise: How many functions from A to B?

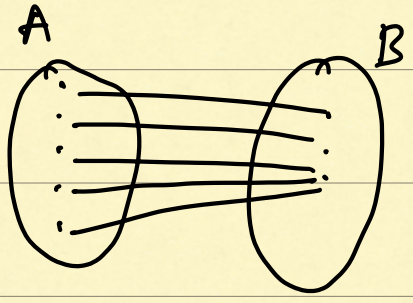
1-1 functions

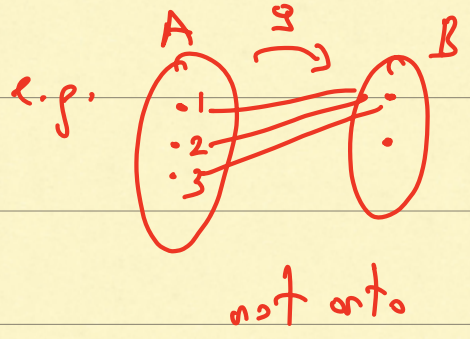
$f: A \to B$ is 1-1 if for every pair $a, b \in A$ we have $a \neq b \implies f(a) \neq f(b)$



Onto functions:

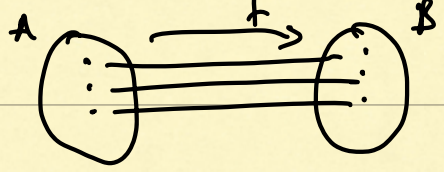
$f: A \to B$ is onto if for every $b \in B$ there exists $a \in A$ s.t. $f(a) = b$.

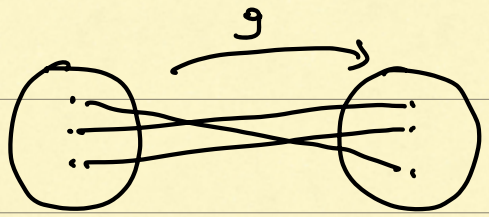




Bijections:

$f: A \to B$ is a bijection if it is both 1-1 and onto.





Popular Sets: $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$

$\mathbb{N}\times\mathbb{N}$, $\mathbb{Z}\times\mathbb{Z}$

$f: A \to B$ ... unary $f(x)$

$f: A \times B \to C$ ... binary $f(x,y)$

$f: A \times B \times C \to D$ ... ternary $f(x,y,z)$

$f: A_1 \times A_2 \times \dots \times A_n \to B$ n-ary

Lecture 2

Math Review cont.

Remark: If the range of a function f is $\{0,1\}$ (or, in general, any binary set), then f is called a predicate / property.

e.g. $\text{even} : \underbrace{\mathbb{Z}}_{\text{domain}} \longrightarrow \underbrace{\{T, F\}}_{\text{range}}$

$\text{even}(1) = F$ ... not even

$\text{even}(4) = T$

$$\text{equals} : \mathbb{Z} \times \mathbb{Z} \longrightarrow \{T, F\}$$

$$\text{equals}(1, 2) = F$$
$$\text{equals}(1, 1) = T$$

Exercise: Define the signature of equality function between three integers.

e.g. $eq(1,1,1) = T$

$eq(1,2,2) = F$

$$<= : \mathbb{Z} \times \mathbb{Z} \longrightarrow \{T, F\}$$

property ... predicate ... relation

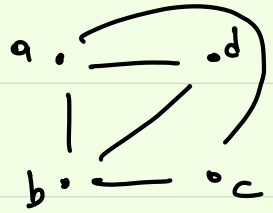
Review: reflexive, symmetric, transitive, antisymmetric

Graphs

A graph G is a pair (V, E) where

- V is the set of nodes
- $E \subseteq V \times V$ is the set of edges.

e.g.

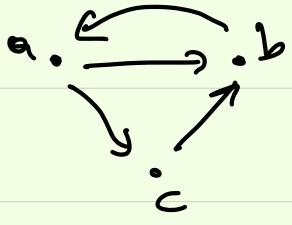


undirected graph

- adjacency matrices
- adjacency lists
- list of edges

- Check: degree of a node, path, circuit/cycle

Directed graphs:



- indegree of a node.
- outdegree of a node
- directed path
- connectedness

Strings (Words) and Formal Languages

Definition: An **alphabet** Σ is a **finite set of symbols**

e.g. $\Sigma = \{\diamondsuit, \square\}$

$\Sigma = \{0, 1\}$

$\Sigma = \{a, b, c, \ldots, x, y, z\}$

$\Sigma = ASCII$

$\Sigma = \{0, 1, 2, \ldots, 9\}$

$\Sigma = \{C, D, E, F, G, A, B\}$

$\Sigma = \{1\}$

Definition: A string (word) over an alphabet Σ is a finite sequence of symbols of Σ.

e.g. $\Sigma = \{0, 1\}$ $\quad w = 010$

$w = 1$

empty string $\quad w = \lambda \ldots (\epsilon)$

* $|w|$ ... length of w

... # of symbols in w

e.g. $w = 010 \Rightarrow |w| = 3$

$w = 1 \Rightarrow |w| = 1$

$w = \lambda \Rightarrow |w| = 0$

$|_| : \text{Word} \longrightarrow \mathbb{N}$

* Given a string $w = w_1 w_2 \cdots w_n$ (i.e $|w| = n$)

the reverse of w is denoted by w^R

and $w^R = w_n w_{n-1} \cdots w_3 w_2 w_1$

e.g. $w = 011 \Rightarrow w^R = 110$

$w = \lambda \Rightarrow w^R = \lambda$

$w = 1 \Rightarrow w^R = 1$

$w = 010 \Rightarrow w^R = 010$

Remark: If $w = w^R$, then w is called a "palindrome".

Exercise: Let Σ be an alphabet and P is the set of all palindromes over Σ. ($\Sigma = \{a, b\}$). Then $|P| = ?$

* Concatenation of strings.

$x = x_1 \cdots x_m$ and $y = y_1 \cdots y_n$

$\Rightarrow$ $x \cdot y = x_1 \cdots x_m y_1 \cdots y_n$

$_\cdot_ : Word \times Word \to Word$

$|x \cdot y| = |x| + |y|$

Remark:

- $w = x \cdot y \cdot z$ "y is a substring of w"
- $w = x \cdot y$ "x is a prefix of w"

e.g. $w = 0011$ $w = 0 \cdot 011$

$w = 00 \cdot 11$

- $w = x \cdot y$ "y is a suffix of w"

* Lexicographic Order

- Shorter strings come before

$\Sigma = \{a, b\}$ and $a < b$

e.g. $x = ab$ $y = a$ $\Rightarrow y <_{lex} x$

- If same length, then use dictionary order

e.g. $x = bab$ $y = bba$ $\Rightarrow x <_{lex} y$

Definition: A language L over an alphabet Σ is a set of strings over Σ.

e.g. $\Sigma = \{0, 1\}$

$L_1 = \{\}$ ... the empty language over Σ.

$L_2 = \{0\}$

$L_3 = \{\lambda\}$

Note: $L_1 \neq L_3$ because $|L_1| = 0$ but $|L_3| = 1$.

Σ^* = the set of all strings over Σ.

$= \{\lambda, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

$L_4 = \{w \in \Sigma^* \mid |w| = 2k, k \in \mathbb{N}\}$

$\Rightarrow L_4 \subseteq \Sigma^*$

Remark: For every language L over Σ, we have $L \subseteq \Sigma^*$

Recursive Definitions

Example: Set of palindroms P over $\Sigma = \{0, 1\}$

Basis: $\lambda, 0, 1 \in P$

Rec. step: If $w \in P$, so are $0w0 \in P$ and $1w1 \in P$

No other rule is needed.

Example: Dyck language D ... set of well-formed parentheses. $\Sigma = \{(,)\}$

e.g. $() \in D$ $((\notin D$

Basis: $\lambda \in D$

Rec. step: If $x, y \in D$, then so are (x) (or (y)) and $x \cdot y$ (or $y \cdot x$) $\in D$.

No other rule is needed.

e.g. $x = ()()$ $y = (())$

$(x) = (()()) \in D$

$x \cdot y = ()() \cdot (()) \in D$.

Lecture 3

Proof by Induction

Example: Show that for every palindrome w, we have $w = w^R$. ($\Sigma = \{a, b\}$)

Proof: by induction on the form of w.

Basis: $w = \lambda \Rightarrow w^R = \lambda$ ✓

$w = a \Rightarrow w^R = a$ ✓

$w = b \Rightarrow w^R = b$ ✓

Ind. Hyp. Assume a palindrome w satisfies $w = w^R$.

Show: For $m = awa$, we have $m = m^R$

$$m^R = (awa)^R$$
$$= aw^Ra$$
$$= awa \quad \text{by ind. hyp we have } w = w^R$$
$$= m.$$

(Similarly, for $m = bwb$ we have $m = m^R$)

Example: Prove that $3^n > 3n$ for $n \geq 2$.

Proof: by induction

Basis: $n = 2 \Rightarrow 3^2 > 3 \cdot 2$ ✓

Ind. Hyp: Assume $3^k > 3k$ for $k < n$.

Show: $3^n > 3n$ is true.

$$3^n = 3 \cdot 3^{n-1}$$

$$> 3 \cdot 3(n-1) \quad \text{by ind. hyp.}$$

$$= 9n - 9$$

$$= 3n + 6n - 9$$

$$> 3n \quad \text{if } n \geq 2.$$

$\Rightarrow 3^n > 3n$ //

Proof by Contradiction

Example: Let $a, b, c > 0$. Prove: if $\underbrace{ab = c}_{p}$, then $\underbrace{a \leq \sqrt{c} \text{ or } b \leq \sqrt{c}}_{q}$. $(p \Rightarrow q)$

Proof: by contradiction

Assume the opposite.

- Assume p is true
- Assume q is false

$\Rightarrow a > \sqrt{c}$ and $b > \sqrt{c}$

$\Rightarrow a \cdot b > \sqrt{c} \cdot \sqrt{c} = c$

Contradiction.

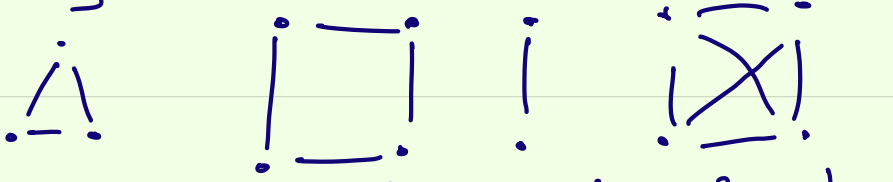
Exercise: - Prove that $\sqrt{2}$ is irrational.

- Prove that there are infinitely many primes.
- Prove that there exists no set of all sets. (Bertrand Russell)

Proof by Construction

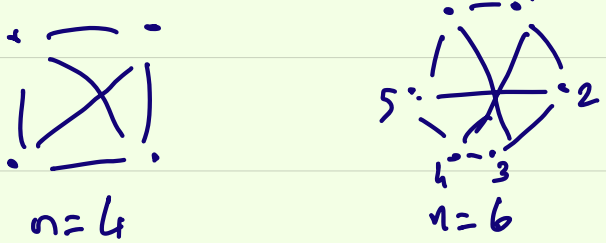
Example: A k-regular graph is a graph in which every node has degree k.

e.g.



2-regular 2-regular 1-regular 3-regular

Claim: For every even number $n \geq 4$, there exist 3-regular graphs with n nodes.



$n=4$ $n=6$ $n=8$

In general we place every node into a cycle and connect the opposite nodes.

nodes .... $0, 1, \dots, n-1$

$G = (V, E)$ where

$V = \{0, 1, \dots, n-1\}$

$E = \{(0,1), (1,2), (2,3), \dots, (n-1, 0)\}$

$\cup \{(0, \frac{n}{2}), (1, \frac{n}{2}+1), \dots, (\frac{n}{2}-1, n-1)\}$

Some more defs. on languages.

- Let Σ be an alphabet
- Let $L_1, L_2 \subseteq \Sigma^*$ be two languages.

Concatenation of L_1 and L_2:

$L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1 \text{ and } v \in L_2\}$

e.g. $\Sigma = \{a, b\}$ $L_1 = \{a, b\}$, $L_2 = \{aa, bb\}$

$L_1 \cdot L_2 = \{aaa, abb, baa, bbb\}$

$L_2 \cdot L_1 = ?$

- For a string w and $n \geq 0$

$$w^n = \begin{cases} \lambda & \text{if } n = 0 \\ w^{n-1} \cdot w & \text{if } n \geq 1 \end{cases}$$

e.g. $w = 01 \Rightarrow w^3 = w^2 \cdot w = w^2 \cdot 01$
$= w \cdot w \cdot 01$
$= 01 \cdot 01 \cdot 01$

- For a language L and $n \geq 0$

$$L^n = \begin{cases} \{\lambda\} & \text{if } n = 0 \\ L^{n-1} \cdot L & \text{if } n \geq 1. \end{cases}$$

- Kleene closure of a language L:

$$L^* = L^0 \cup L \cup L^2 \cup \ldots = \bigcup_{i \geq 0} L^i$$

- Positive closure of a language L:

$$L^+ = L \cup L^2 \cup L^3 \cup \ldots = \bigcup_{i > 0} L^i$$

Note: $L^* = L^+ \cup \{\lambda\}$

Lecture 4

Finite Automata and Regular Languages

- Simplest computational model
- Limited computational power
- Applications: Compiler construction, program analysis. Systems biology.

Definition: A finite automaton (FA) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

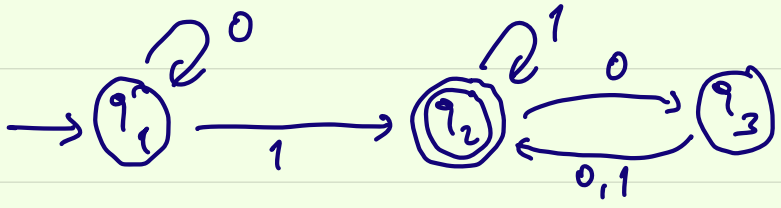
- Q is a set of states
- Σ is the input alphabet
- δ is a transition function:

$$\delta : Q \times \Sigma \longrightarrow Q \qquad \text{e.g. } \delta(q, a) = q'$$

- q_0 is the initial state (i.e. $q_0 \in Q$)
- $F \subseteq Q$ is the set of final/accepting states.

Example: $\Sigma = \{0, 1\}$

M:



q_1 : the initial state

q_2 : the final state

$Q = \{q_1, q_2, q_3\}$

$F = \{q_2\}$

$\delta(q_1, 0) = q_1$

$\delta(q_1, 1) = q_2$

$\delta(q_3, 0) = q_2$

$\delta(q_3, 1) = q_2$

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

"deterministic"

e.g. $w = 001$ ... accepted

$q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2$

$w_2 = 010$ ... rejected

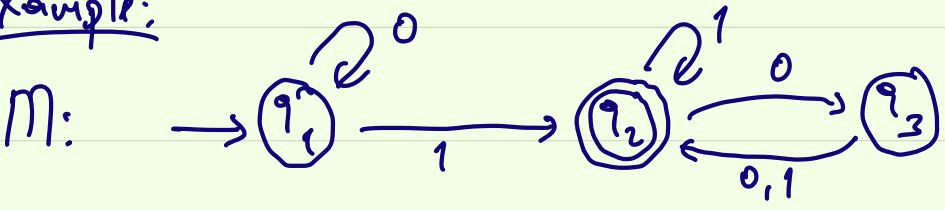
$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3$

$w_3 = \lambda$ ... rejected.

Definition: A string $w \in \Sigma^*$ is **accepted** by a FA M if M starts with the initial state and enters a final state $q_f \in F$ after processing w.

Example:

M:



Q: Set of all strings accepted by M?

called $L(M)$

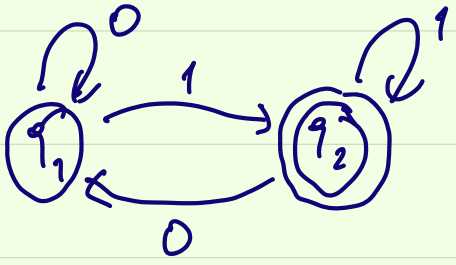
Rejected	Accepted	
λ	1	0100
0	01	01000
00	0101	010000
		0111000
		1110000

$L(M)$ = "set of all strings that contain even number of 0s after the last occurrence of 1"

Definition: For an FA M, $L(M)$ denotes the language accepted by M.

$$L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$$

Example:



M_2

$L(M_2) = ?$

$\lambda \in L(M_2)$? No

$1 \in L(M_2)$? Yes

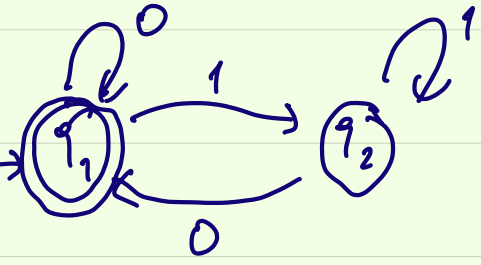
$0 \in L(M_2)$? No

$0 \ldots 0 \in L(M_2)$? No

$000\ldots01 \in L(M_2)$? Yes.

$$L(M_2) = \{ w \in \Sigma^* \mid w = w'1 \text{ , } w' \in \Sigma^* \}$$

Example:



M_3

$$L(M_3) = \{\lambda\} \cup \{ w \in \Sigma^* \mid w = w'0 \text{ , } w' \in \Sigma^* \}$$

Exercise:

- FA for only $\{w \in \Sigma^* \mid w = w'0, w' \in \Sigma^*\}$?
- FA for the set of strings that start and end with the same string. e.g. $w = 0 \ldots 0$ or $w = 1 \ldots 0$ or

Formal definition of Computation of an FA:

Given $M = (Q, \Sigma, \delta, q_0, F)$ and an input string $w = w_1 w_2 \ldots w_n$

"M accepts w if there exists a sequence of states $r_0, r_1, \ldots, r_n \in Q$ s.t.

- r_0 is the initial state $q_0 \in Q$
- $r_n \in F$
- $\delta(r_i, w_{i+1}) = r_{i+1}$ for all $i = 0, \ldots, n-1$

$$r_0 \xrightarrow{w_1} r_1 \xrightarrow{w_2} r_2 \cdots \to r_i \xrightarrow{w_{i+1}} r_{i+1} \cdots$$

Def: A language $L \subseteq \Sigma^*$ is "regular" if $L = L(M)$ for some FA M.

Lecture 5

Example:

1) M: $L(M) = \emptyset$

2) M: $L(M) = \Sigma^*$

3) M: $L(M) = \Sigma^* - \{\lambda\}$

4) M: $L(M) = \{\lambda\}$

Designing Finite Automata

Given a language $L \subseteq \Sigma^*$, design an FA that accepts it.

Example: Design an FA that accepts all the strings containing an even number of 1s. ($\Sigma = \{0,1\}$).

i.e. $L = \{ w \in \Sigma^* \mid |w|_1 = 2k, \ k \in \mathbb{N} \}$

Need to construct an FA $M = (Q, \Sigma, \delta, q_0, F)$ s.t. $L = L(M)$.

? ? ? ? ?

Idea: Count modulo 2.

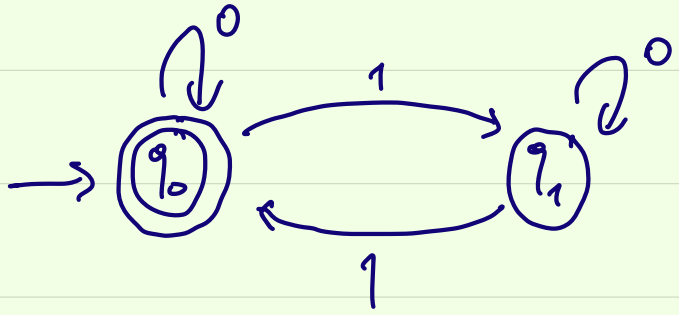
q_0 ... "even number 1's" final, initial

q_1 ... "odd number of 1"

$Q = \{q_0, q_1\}$ $\Sigma = \{0, 1\}$

$F = \{q_0\}$ $\delta = ?$

q_0



$\lambda \in L$

$1 \notin L$

$01101 \notin L$.

0110011

Exercise: FA to accept all strings with odd number of 1s?

Example: FA to accept all strings that have "aba" as a substring.

i.e. $L = \{w \in \Sigma^* \mid w = w'aba\,w'', \; w', w'' \in \Sigma^*\}$

$\lambda \notin L$ $a \notin L$, $b \notin L$, $ab \notin L$, $abbba \notin L$

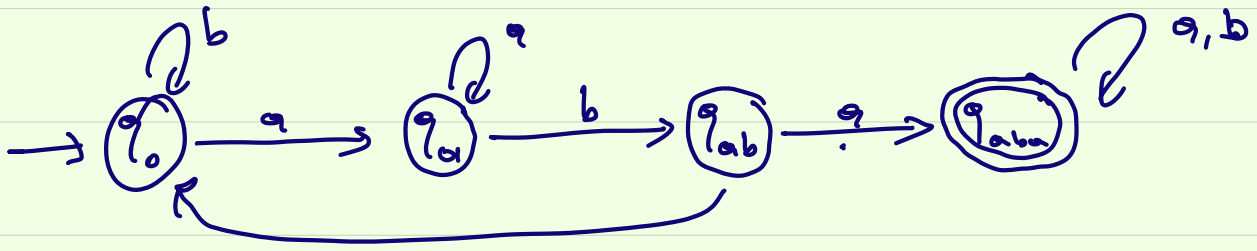
$aba \in L$, $aabab \in L$, ... $abbaba$

q_a ... "an a is seen"

q_{ab} ... "an ab is seen"

final ... q_{aba} ... "an aba is seen."

q_0 ... initial state.



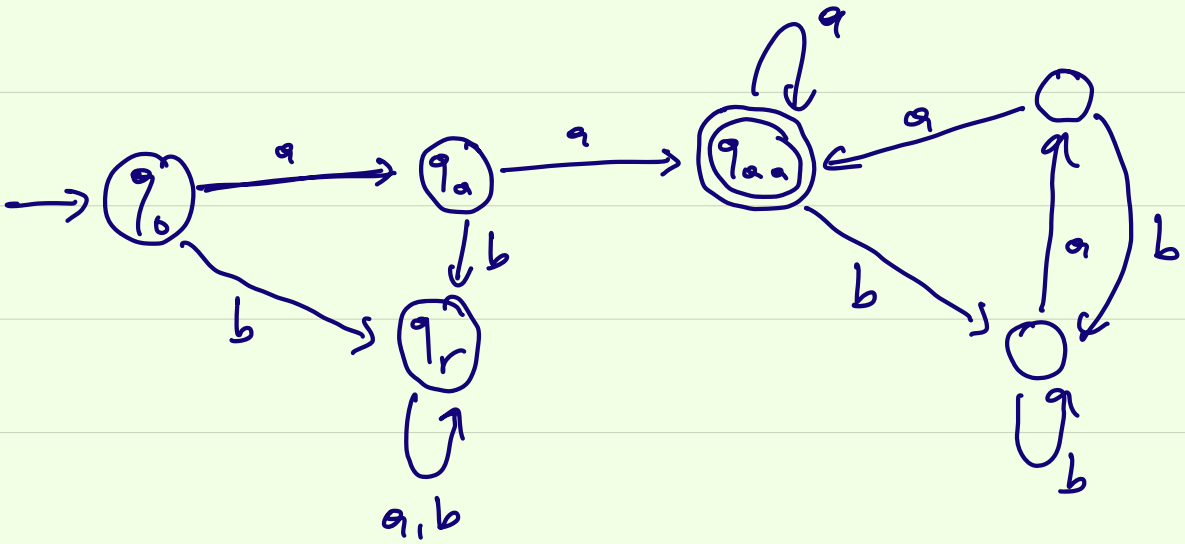
Exercise FA to accept all strings that have "bba" as a substring.

Example: FA to accept all strings that begin and end with "aa".

λ, a, b, bb, bw', w'b, w'ab, X

aa, aaa, aa w'aa s.t. $w' \in \Sigma$ ✓

$L = \{ w \in \Sigma^* \mid w = aa\, w'aa,\ w' \in \Sigma^* \} \cup \{aa, aaa\}$



aabbaa ✓

aababaa ✓

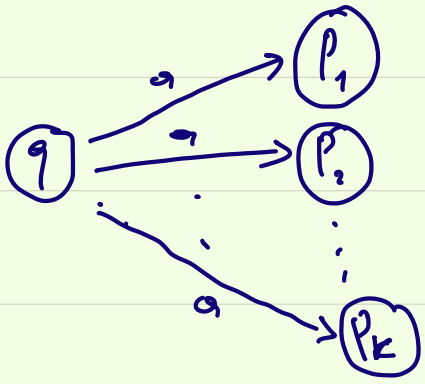
Nondeterministic Finite Automata (NFA)

So far, at any given state q, an FA jumps to the next state deterministically;

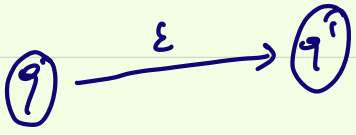
$$\delta(q, a) = q' \quad \text{-- unique.}$$

Hence, we call such FA "deterministic FA" (DFA).

In an NFA, given a state q and an input symbol a, there is a set of possible next states. i.e. $\delta(q, a) = \{p_1, p_2, \ldots, p_k\} \subseteq Q$

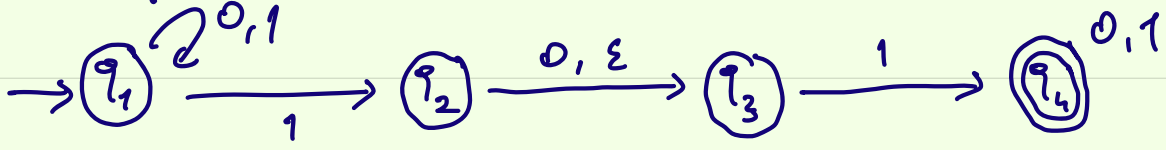


Also, in an NFA, there may be ε-transitions:



That is, N can jump from q to q' without consuming any symbol.

Example:

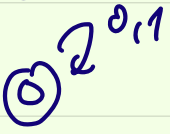
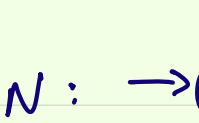


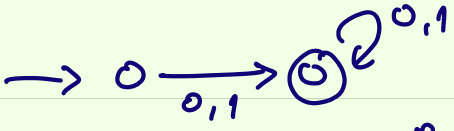
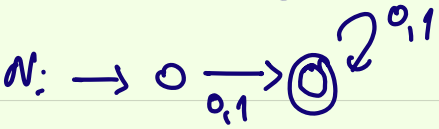
$$\delta(q_1, 1) = \{q_1, q_2\}$$
$$\delta(q_1, 0) = \{q_1\}$$
$$\delta(q_3, 0) = \emptyset$$

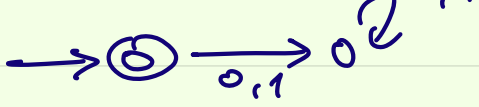
To be cont.

Example:

1) M:  N:

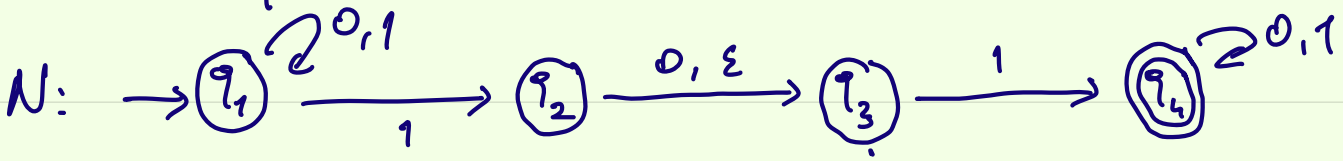
2) M:  N: 

3) M:  N: 

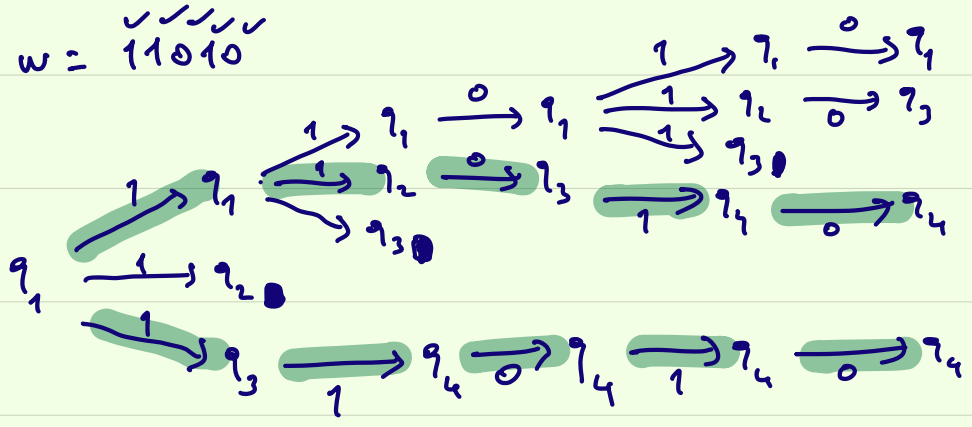
4) M:  N: 

Lecture 6

Example:



$$\delta(q_1, 1) = \{q_1, q_2\}$$
$$\delta(q_1, 0) = \{q_1\}$$
$$\delta(q_3, 0) = \emptyset$$



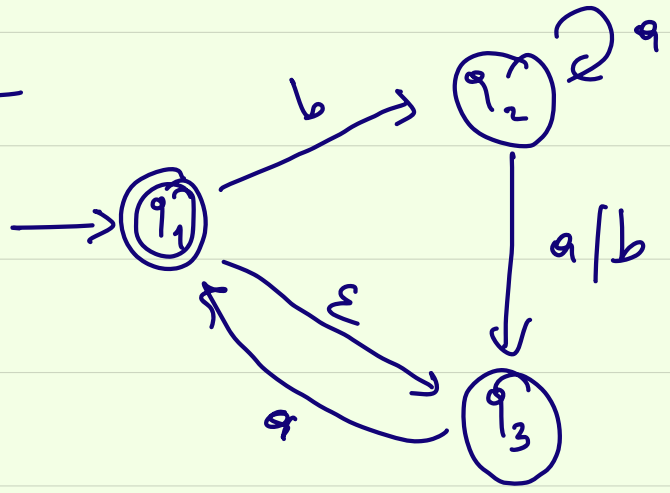
Exercise: Try 001101?

Remark:

In a DFA we have only one sequence of states (i.e. one possible set of state transitions). Yet, in an NFA, there may be more than possible sequence of transitions for a given input.

Example:

N:



$\delta(q_1, \varepsilon) = \{q_3\}$

$\delta(q_2, a) = \{q_2, q_3\}$

δ	a	b	ε
q_1	$\emptyset$	$\{q_2\}$	$\{q_3\}$
q_2	$\{q_2, q_3\}$	$\{q_3\}$	$\emptyset$
q_3	$\{q_1\}$	$\emptyset$	$\emptyset$

$\Sigma = \{a, b\}$

$\Sigma_\varepsilon = \{a, b, \varepsilon\}$

Definition: An NFA N is a 5-tuple

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

- Q: states
- Σ: input alphabet
- δ: transition function
- q_0: init. state
- F: set of final states

- $\delta : Q \times \Sigma_\varepsilon \to 2^Q$

- $F \subseteq Q$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

2^Q: set of all subsets of Q.

Definition: An input string $w = w_1 w_2 \ldots w_n \in \Sigma^*$ is accepted by an NFA N if there exists a sequence of states $r_0, r_1, \ldots, r_n$ s.t.

- $r_0 = q_0$ (i.e. r_0 should be the init. state.)
- $r_n \in F$ (i.e. r_n should be a final state)
- $r_{i+1} \in \delta(r_i, w_{i+1})$

Theorem: Let $L \subseteq \Sigma^*$ be a language.

Then $L = L(M)$ for some DFA M p

if and only if

$L = L(N)$ for some NFA N. q

To prove $p \iff q$ we need to prove

- $p \Rightarrow q$
- $q \Rightarrow p$

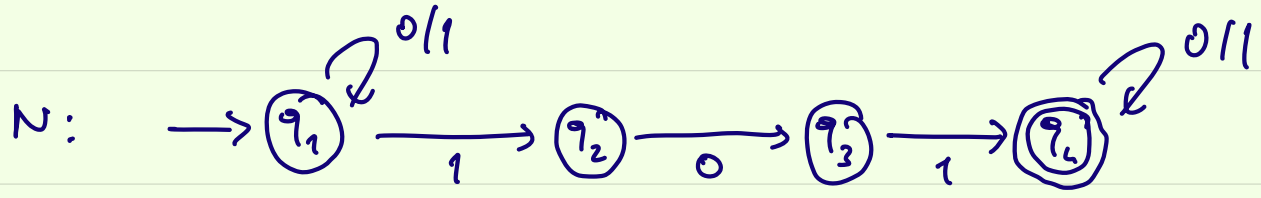
Proof of $p \Rightarrow q$: ($L = L(M) \ldots \Rightarrow L = L(N) \ldots$)

Trivial. Note that every DFA M is an NFA by defn.

Proof of $q \Rightarrow p$: ($L = L(N) \ldots \Rightarrow L = L(M) \ldots$)

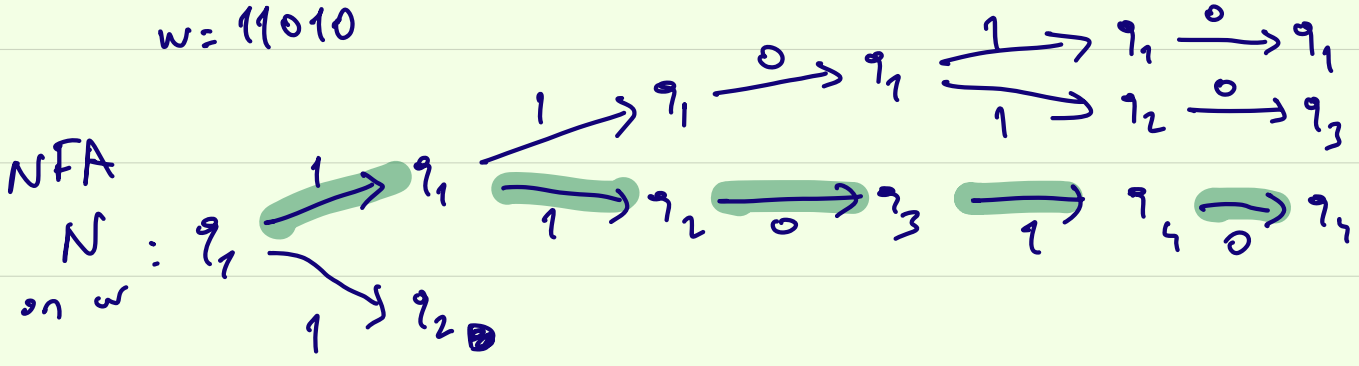
Idea: Given an NFA $N = (Q, \Sigma, \delta, q_0, F)$, construct a DFA $M = (Q', \Sigma, \delta', q_0', F')$ s.t. $L(N) = L(M)$.

For example, take the following NFA N:

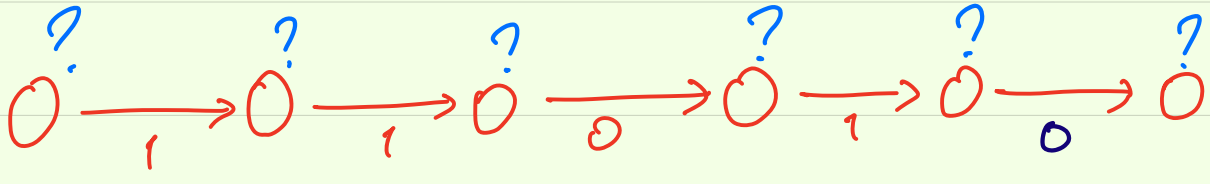


$w = 11010$

NFA N on w:

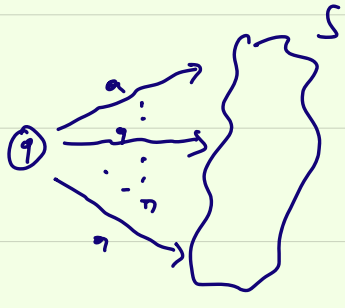


The equiv. DFA on w:

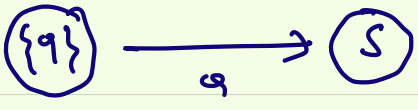


IDEA:

NFA:

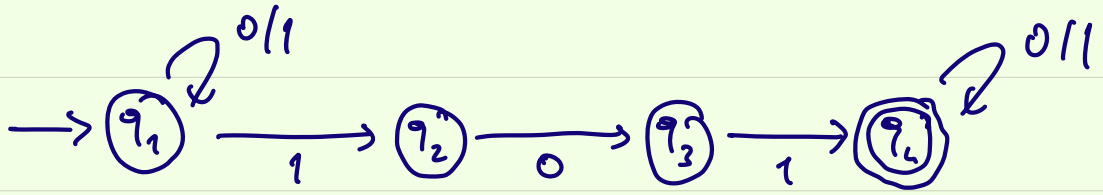


DFA:

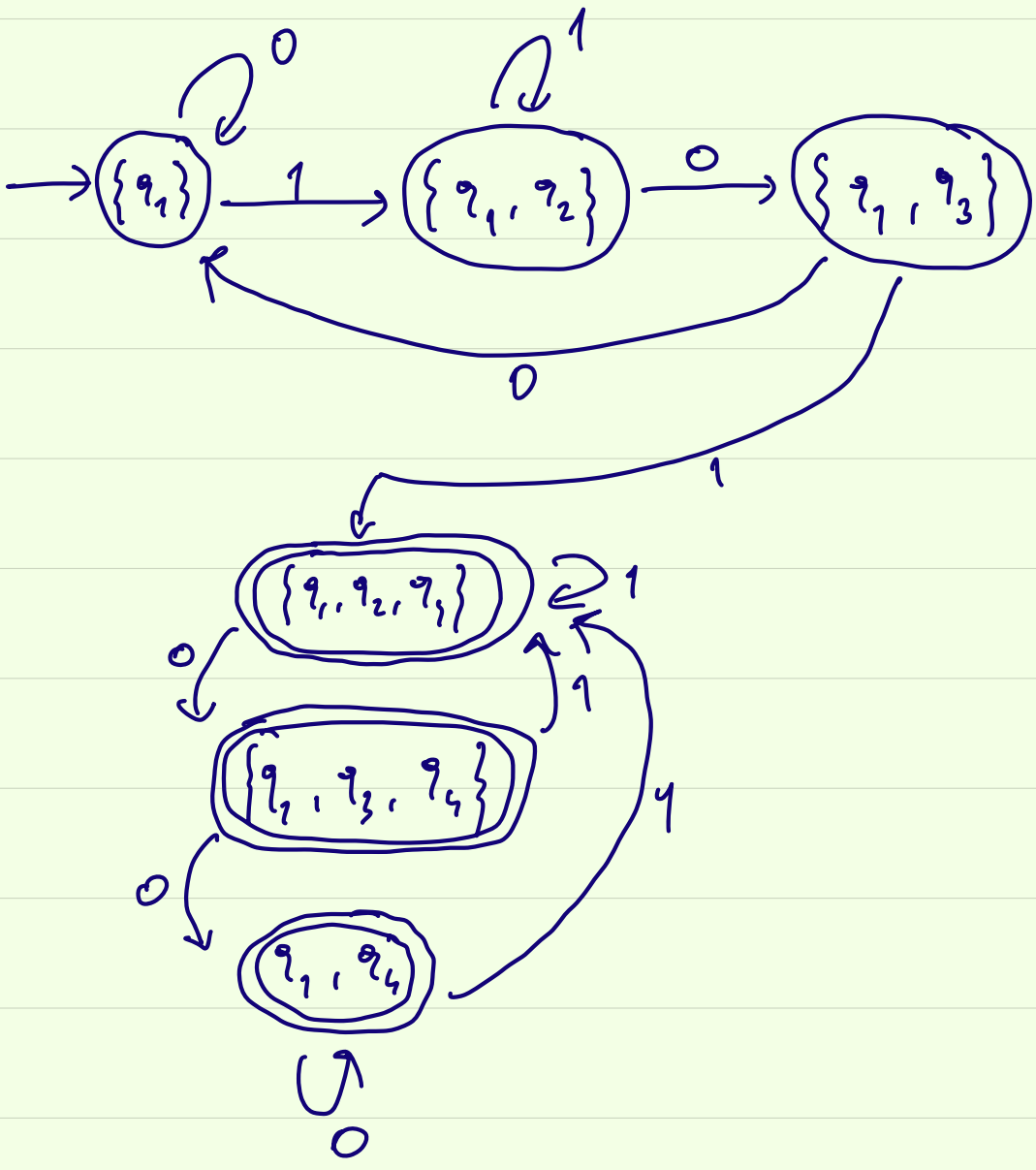


"subset construction"

NFA N:

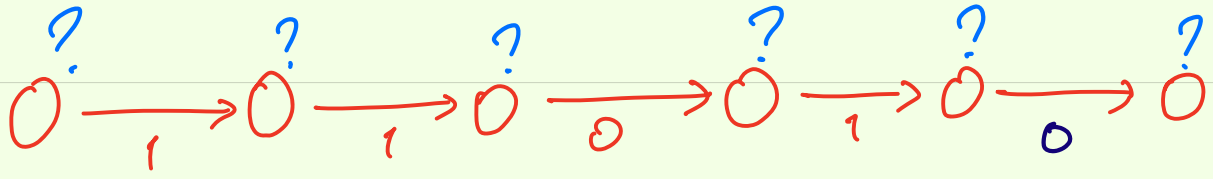


DFA M:



Exercise:

The equiv. DFA on w:



In general, the equivalent DFA $M = (Q', \Sigma, \delta', q_0', F')$ is defined as follows:

- $Q' \subseteq 2^Q$
- $q_0' = \{q_0\}$ (Note: $q_0 \neq \{q_0\}$)
- $F' = \{s \in Q' \mid s \cap F \neq \emptyset\}$
- $\delta'(s, a) = \bigcup_{q \in s} \delta(q, a)$ //

Lecture 7

Proof of equivalence of NFAs and DFAs ... contin.

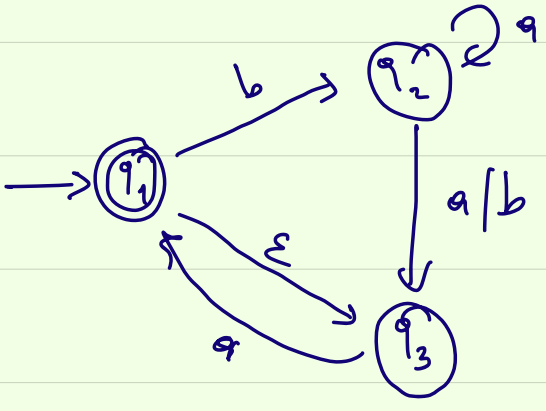
Q: What if an NFA has ε-transitions?

A: Modify the transition function δ' with ε-closures?

Definition: Let $S \in Q'$. Then the ε-closure of S is defined as

$$E(S) = \{ q \in Q \mid q \text{ is reachable from a state in } S \text{ using zero or more } \varepsilon\text{-transitions only} \}$$

e.g.



$S = \{q_1\}$

$\Rightarrow E(S) = \{q_1, q_3\}$

$S = \{q_1, q_2\}$

$\Rightarrow E(S) = \{q_1, q_3, q_2\}$

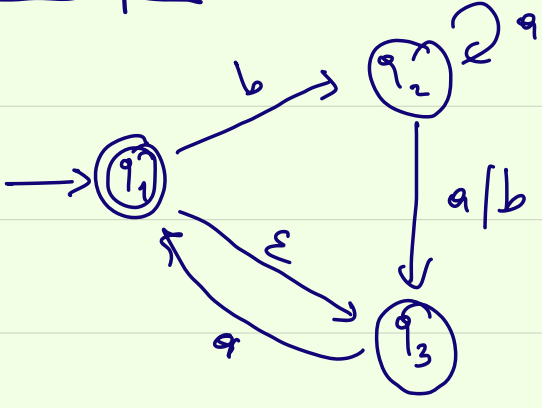
$S = \emptyset \Rightarrow E(S) = \emptyset$.

Then we modify the construction of the equiv. DFA as follows:

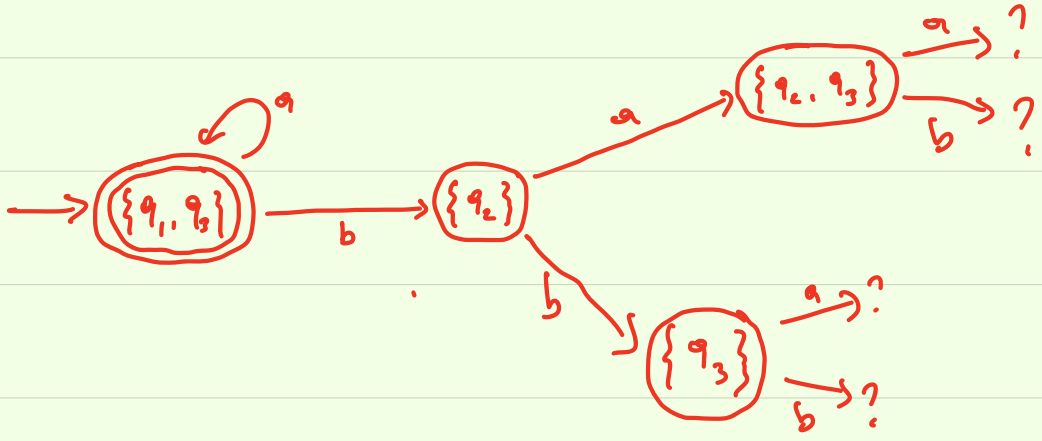
- $Q' \subseteq 2^Q$
- $q_0'' = E(\{q_0\})$
- $F' = \{s \in Q' \mid s \cap F \neq \emptyset\}$
- $\delta''(S, a) = \bigcup_{q \in S} E(\delta(q, a))$

—— ∞ End of Proof of Equivalence ——

Example:



$q_0'' = E(\{q_0\}) = \{q_1, q_3\}$



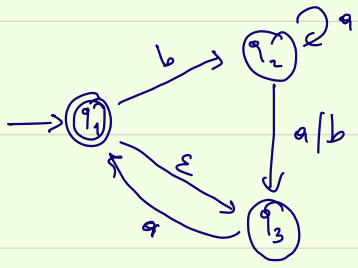
Exercise: Complete the construction.

- $\delta''(\{q_1, q_3\}, a) = E(\delta(q_1, a)) \cup E(\delta(q_3, a))$

$= E(\emptyset) \cup E(\{q_1\})$

$= \emptyset \cup \{q_1, q_3\}$

$= \{q_1, q_3\}$



- $\delta''(\{q_1, q_3\}, b) = E(\delta(q_1, b)) \cup E(\delta(q_3, b))$

$= E(\{q_2\}) \cup E(\emptyset)$

$= \{q_2\}$

- $\delta''(\{q_2\}, a) = E(\delta(q_2, a))$

$= E(\{q_2, q_3\})$

$= \{q_2, q_3\}$

- $\delta''(\{q_2\}, b) = E(\delta(q_2, b))$

$= E(\{q_3\}) = \{q_3\}$

Corollary: A language $L \subseteq \Sigma^*$ is regular if $L = L(N)$ for some NFA N.

REGULAR EXPRESSIONS

Notation: For a reg. exp. α, we write $L(\alpha)$ to denote its language.

Definition: Let Σ be an alphabet. Then the set of all regular expressions over Σ is "recursively" defined as follows:

Basis:
- $\emptyset$ is a reg. exp. $\quad L(\emptyset) = \emptyset$
- λ is a reg. exp. $\quad L(\lambda) = \{\lambda\}$
- $\forall a \in \Sigma$ is a reg. exp. $\quad L(a) = \{a\}$

Rec. Step: If α and β are reg. exp., then
- $\alpha + \beta$ is a reg. exp. $\quad L(\alpha+\beta) = L(\alpha) \cup L(\beta)$
- $\alpha \cdot \beta$ " " $\quad L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$
- α^* " " $\quad L(\alpha^*) = (L(\alpha))^*$

$$\{0\}^* = \{0\}^0 \cup \{0\} \cup \{0\}^2 \cdots$$

Example: $\Sigma = \{0,1\}$

1) $\alpha = 0^* 1 0^*$ $\quad L(\alpha) = ?$

$0 \notin L(\alpha)$ $\quad 011 \in L(\alpha)$

$1 \in L(\alpha)$

$01 \in L(\alpha)$

$10 \in L(\alpha)$

$$L(\alpha) = L(0^* \cdot 1 \cdot 0^*)$$

$$= L(0^*) \cdot L(1) \cdot L(0^*)$$

$$= (L(0))^* \cdot \{1\} \cdot (L(0))^*$$

$$= \{0\}^* \cdot \{1\} \cdot \{0\}^*$$

= the set of all strings with exactly one 1.

$$= \{ w \in \{0,1\}^* \mid |w|_1 = 1 \}$$

2) $\alpha = (0+1)^* 1 (0+1)^*$ $\quad 1 \checkmark$ $\quad 11 \checkmark$

$$L(\alpha) = L((0+1)^* \cdot 1 \cdot (0+1)^*)$$

$$= L((0+1)^*) \cdot L(1) \cdot L((0+1)^*)$$

$$= (L(0+1))^* \cdot L(1) \cdot (L(0+1))^*$$

$$= (L(0) \cup L(1))^* \cdot \{1\} \cdot (L(0) \cup L(1))^*$$

$$= (\{0\} \cup \{1\})^* \cdot \{1\} \cdot (\{0\} \cup \{1\})^*$$

$$= \{0,1\}^* \cdot \{1\} \cdot \{0,1\}^*$$

$$= \Sigma^* \cdot \{1\} \cdot \Sigma^*$$

$$= \{ w \in \{0,1\}^* \mid w = x\,1\,y \quad , x, y \in \Sigma^* \}$$

$$= \{ w \in \{0,1\}^* \mid |w|_1 \geq 1. \}$$

3) $\alpha = (0+1)^*\,010\,(0+1)^*$

$L(\alpha) = \Sigma^* \cdot \{010\} \cdot \Sigma^*$

4) $\alpha = 00\,(0+1)^*\,11$

$L(\alpha) = \{00\} \cdot \Sigma^* \cdot \{11\}$

5) $\alpha = ((0+1)(0+1))^*$

$L(\alpha) = \{ w \in \Sigma^* \mid |w| = 2k \quad , k \in \mathbb{N} \}$

Exercise: Reg. exp. for the set of all odd length strings?

Lecture 8

Equivalence of NFAs and REs

Theorem: $L = L(N)$ for some NFA N (p)

if and only if

$L = L(\alpha)$ for some RE α. (q)

Proof of (q $\Rightarrow$ p):

(i.e. $L = L(\alpha)$ ... $\Rightarrow$ $L = L(N)$ ...)

Idea: Given a reg. exp α, we construct an equiv. NFA N. The construction is defined recursively.

Basis:

- $\alpha = \phi$ ... N: 
- $\alpha = \varepsilon$ ... N: 
- $\alpha = a \in \Sigma$ ... N: 

Rec. Step:

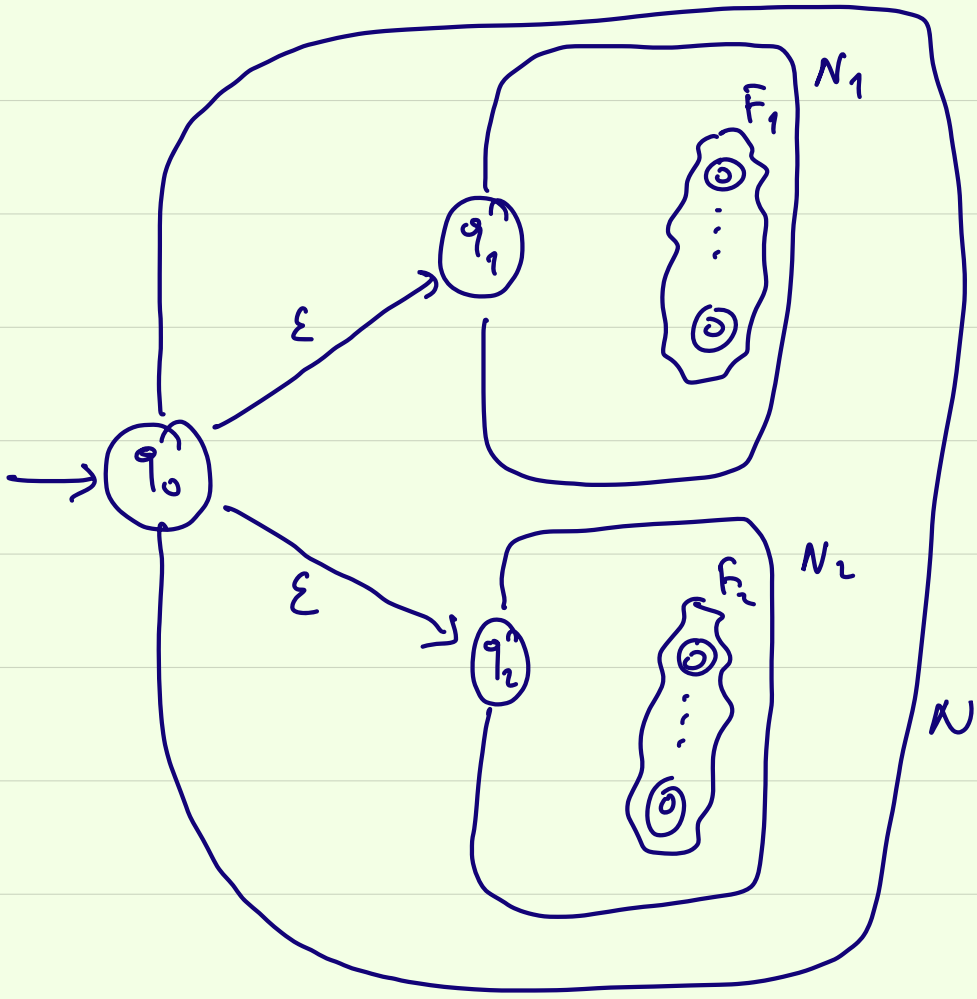
1) $\alpha = \alpha_1 + \alpha_2$ $\qquad L(\alpha) = L(\alpha_1) \cup L(\alpha_2)$

α_1 .. $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

α_2 .. $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

α N?

$w \in L(\alpha) \iff w \in L(\alpha_1) \vee w \in L(\alpha_2)$



$$N = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, \delta, q_0, F_1 \cup F_2)$$

where

$$\delta(q, a) = \begin{cases} \{q_1, q_2\} & \text{if } q = q_0,\ a = \varepsilon \\ \delta_1(q, a) & \text{if } q \in Q_1 \\ \delta_2(q, a) & \text{if } q \in Q_2 \end{cases}$$

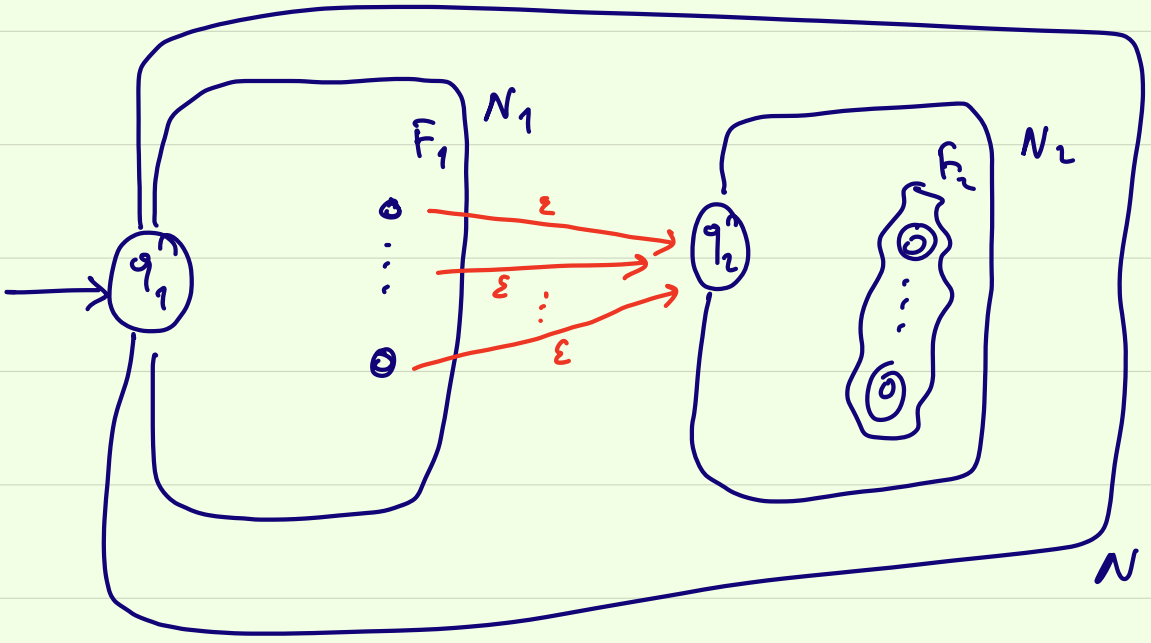
2) $\alpha = \alpha_1 \cdot \alpha_2$ $\qquad L(\alpha) = L(\alpha_1) \cdot L(\alpha_2)$

α_1 .. $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

α_2 .. $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

$$L(\alpha) = L(\alpha_1) \cdot L(\alpha_2)$$

$$= \{ x \cdot y \mid x \in L(\alpha_1) \text{ and } y \in L(\alpha_2) \}$$



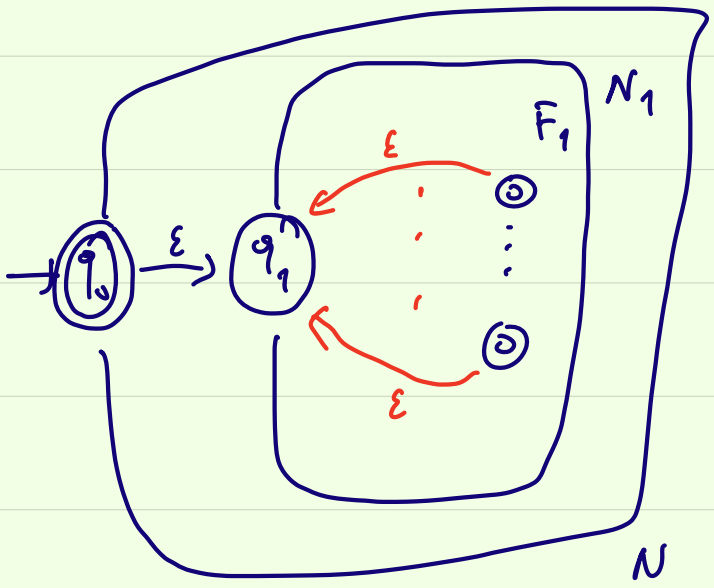
$$N = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F_2)$$

where

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & \text{if } q \in Q_1 - F \\ \delta_1(q, a) & \text{if } q \in F_1, a \neq \varepsilon \\ \delta_1(q, a) \cup \{q_2\} & \text{if } q \in F_1, a = \varepsilon \\ \delta_2(q, a) & \text{if } q \in Q_2 \end{cases}$$

3) $\alpha = \alpha_1^*$ $\qquad L(\alpha) = \left(L(\alpha_1)\right)^* = L(\alpha_1)^0 \cup L(\alpha) \cup \dots$

$\alpha_1 \dots \quad N_1 = (Q_1, \Sigma, \delta_1, q_{1_1}, F_1)$



$w \in L(\alpha)$

$\Rightarrow w \in \left(L(\alpha_1)\right)^*$

$\Rightarrow w = x^n$

$n = 0, 1, \dots$

$N = (Q_1 \cup \{q_0\}, \Sigma, \delta, q_0, F_1 \cup \{q_0\})$

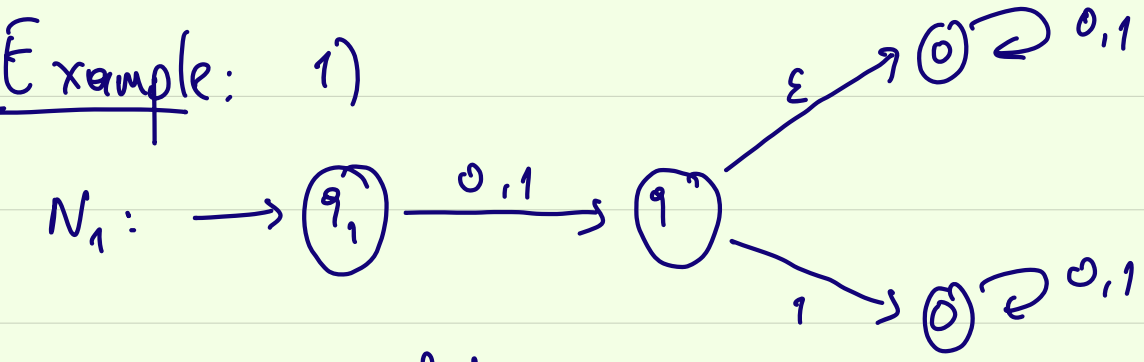
where

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 - F_1 \\ \delta_1(q,a) & \text{if } q \in F_1,\ a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & \text{if } q \in F_1,\ a = \varepsilon \\ \{q_1\} & \text{if } q = q_0,\ a = \varepsilon \\ \emptyset & \text{if } q = q_0,\ a \neq \varepsilon \end{cases}$$

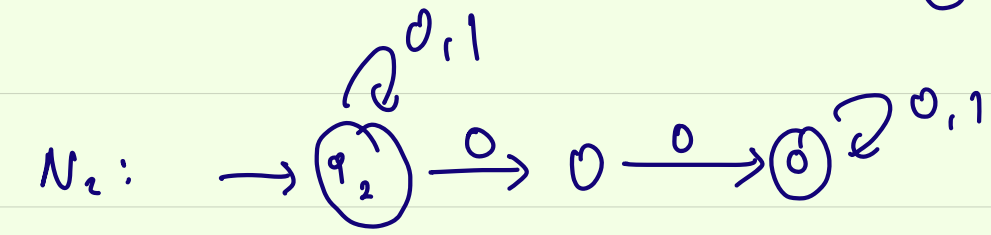
– End of proof of $q \Rightarrow p$ –

Example: 1)

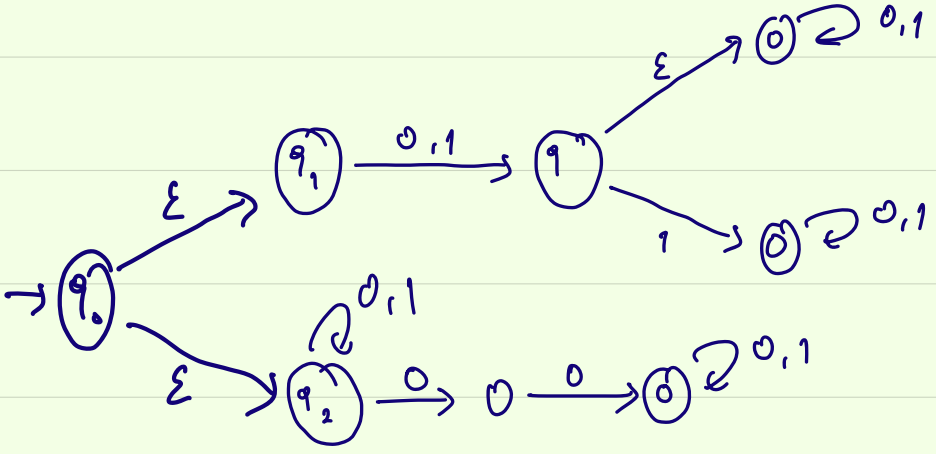
α_1 N_1:



α_2 N_2:

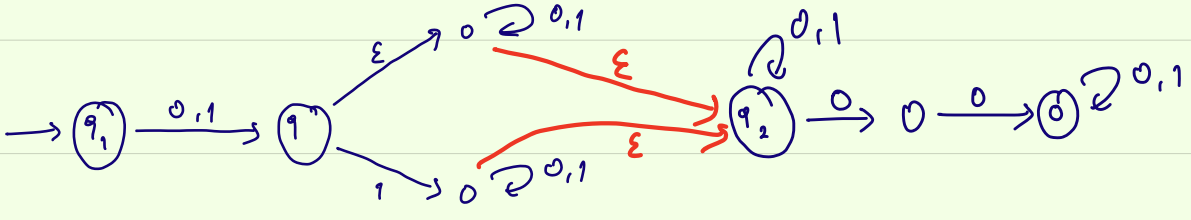


NFA for $\alpha = \alpha_1 + \alpha_2$

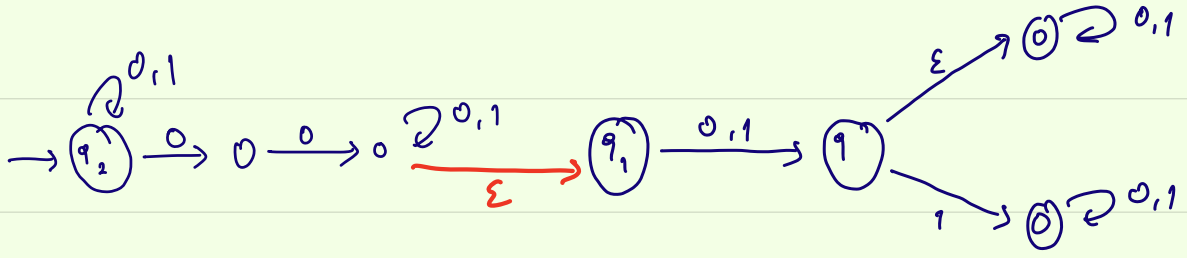


2) NFA for $\alpha = \alpha_2 + \alpha_1$? Same.

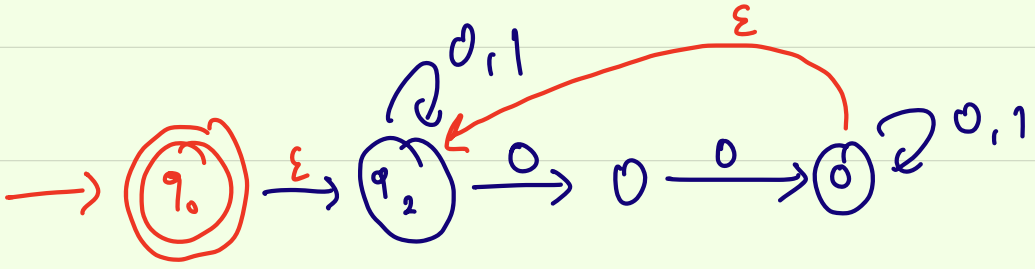
3) NFA $\alpha = \alpha_1 \cdot \alpha_2$



4) NFA $\alpha = \alpha_2 \cdot \alpha_1$



5) NFA for $\alpha = \alpha_2^*$



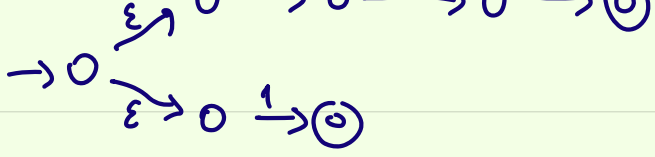
6) $\alpha = (00+1)^*0 + \varepsilon$ $\quad \Sigma = \{0,1\}$

For 0 : $\rightarrow 0 \xrightarrow{0} ((0))$

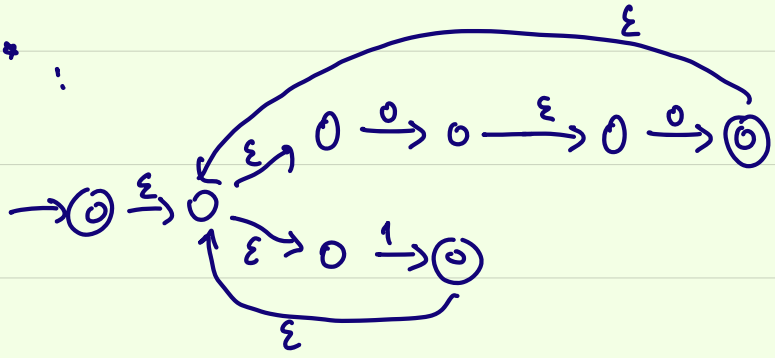
For 00: $\rightarrow 0 \xrightarrow{0} 0 \xrightarrow{\varepsilon} 0 \xrightarrow{0} ((0))$

For 1 : $\rightarrow 0 \xrightarrow{1} ((0))$

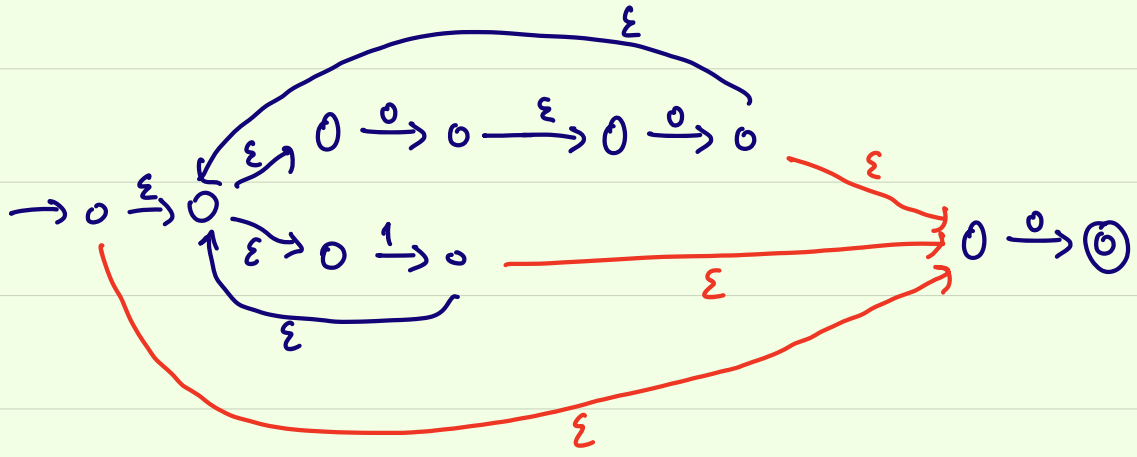
For 00+1 :



For $(00+1)^*$:

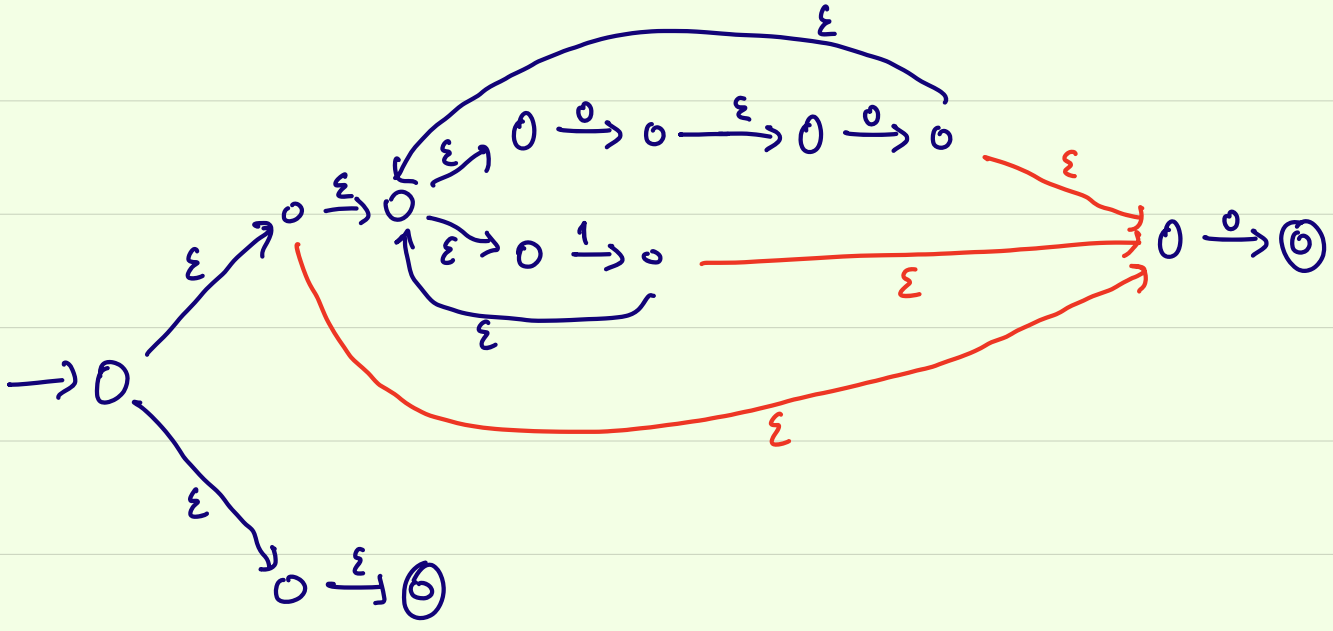


For $(00+1)^*.0$



For ε: $\rightarrow 0 \xrightarrow{\varepsilon} ◎$

For $(00+1)^*0 + \varepsilon$:



Lecture 9

Proof of $p \Rightarrow q$ part

(i.e. $L = L(N)$ ... $\Rightarrow$ $L = L(\alpha)$ ...)

NFA-to-RE?

To prove this, we need to define a new comp. model: "Generalized NFA" (GNFA)

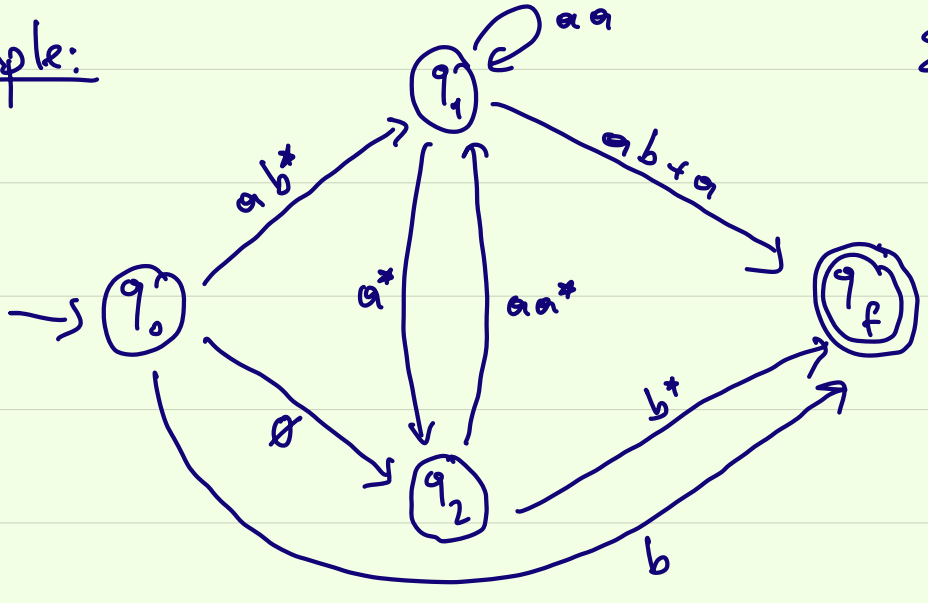
A GNFA has

- a unique init. state q_0
- a " final state q_f
- no transitions going out of q_f
- no transitions coming into q_0
- transitions labelled with reg. exp.

Example:

$\Sigma = \{a, b\}$

N



- $w = b \in L(N)$
- $w = a \in L(N)$

$$q_0 \xrightarrow{a} q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{\varepsilon} q_f$$

- $w = a^2 \in L(N)$

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{\varepsilon} q_f$$

- $w = ab^3a^4b \in L(N)$

$$q_0 \xrightarrow{ab^3} q_1 \xrightarrow{a^4} q_2 \xrightarrow{b} q_f$$

- $w = ab^3a^4b^2 \in L(N)$
- $w = ba \notin L(N)$
- $w = \varepsilon$

A GNFA N is defined as a 5-tuple

$N = (Q, \Sigma, \delta, q_0, q_f)$ where

$$\delta : Q - \{q_f\} \times Q - \{q_0\} \longrightarrow RE$$

$$(q) \xrightarrow{RE} (q')$$

(i.e. $\delta : Q - \{q_f\} \times RE \to Q - \{q_0\}$)

Definition: A string $w = x_1 \cdot x_2 \cdots x_n$

(x_i, $i = 1, \ldots, n$, are substrings of w)

is accepted by a GNFA $N = (Q, \Sigma, \delta, q_0, q_f)$

if there exists a sequence of states

$r_0, r_1, \ldots, r_n$ such that

i) r_0 is init. state, i.e. $r_0 = q_0$

ii) r_n is final state, i.e., $r_n = q_f$

iii) $\delta(r_{i-1}, r_i) = R_i$ and $x_i \in L(R_i)$

Back to proof:

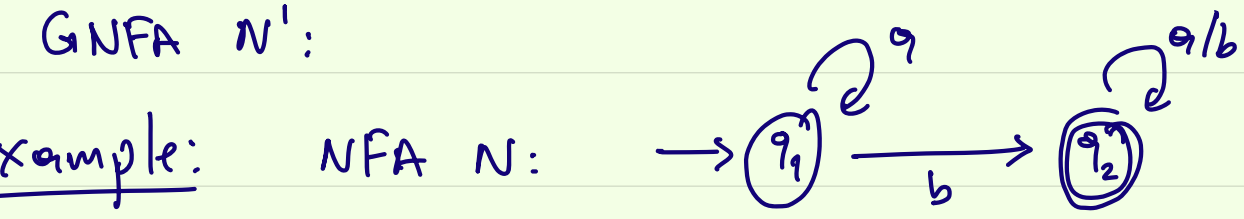
Outline:

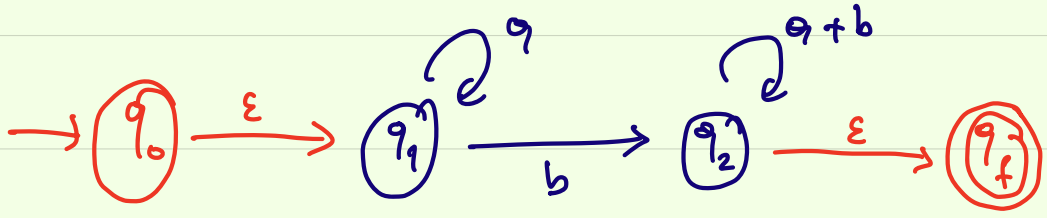
NFA N $\xrightarrow{\text{convert}}$ GNFA N' $\xrightarrow[\text{states}]{\text{Eliminate}}$ GNFA N'' $\longrightarrow$ The equiv. R.E. α

$$(L(N) = L(N') = L(N'') = L(\alpha))$$

1) Convert an NFA N into an equivalent GNFA N':

Example: NFA N:





In general, we

- add q_0 and an ε-transition from it to the old init. state.
- add q_f and ε-transitions from old final states to it.
- Convert every label into a regular expression.

e.g. $O \xrightarrow{a,b} O \longrightarrow O \xrightarrow{a+b} O$

2) Eliminate states from GNFA N' to obtain GNFA N' (with two states only).

The Algorithm:

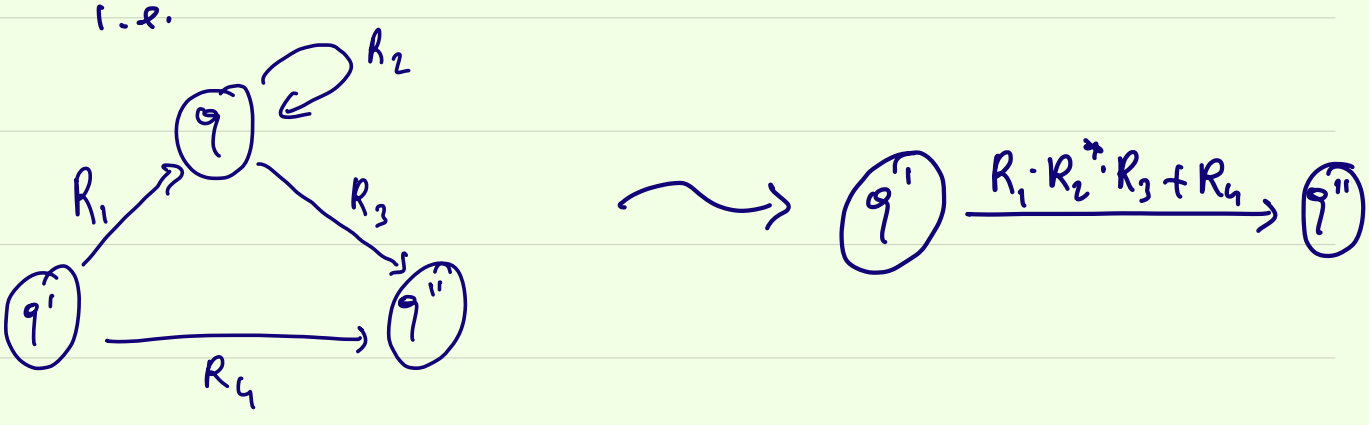
Repeat the following until there are two states left (q_0 and q_f namely).

- Select a state $q \in Q - \{q_0, q_f\}$
- For every pair of states (q', q'') s.t. $q' \in Q - \{q, q_f\}$ and $q'' \in Q - \{q, q_0\}$

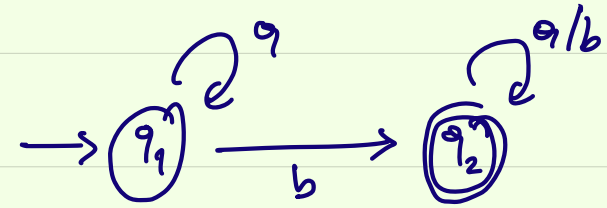
label the transition from q' to q'' with $R_1 \cdot R_2^* \cdot R_3 + R_4$ where

- R_1 is the transition of $q' \to q$
- R_2 " " " " $q \to q$
- R_3 " " " " $q \to q''$
- R_4 " " " " $q' \to q''$

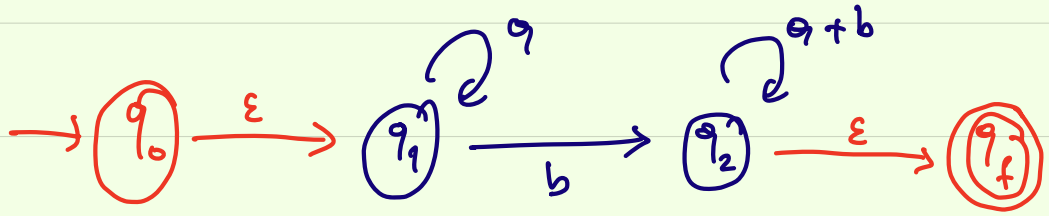
i.e.



Example: NFA N:

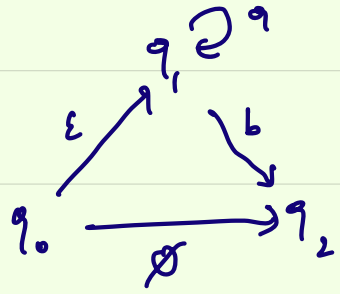


N':



Recall: $\alpha \cdot \varepsilon = \alpha$, $\varepsilon \cdot \alpha = \alpha$, $\alpha + \emptyset = \alpha$, $\alpha \cdot \emptyset \neq \alpha$, $L(\emptyset^*) = \{\varepsilon\}$, $\emptyset^* = \varepsilon$

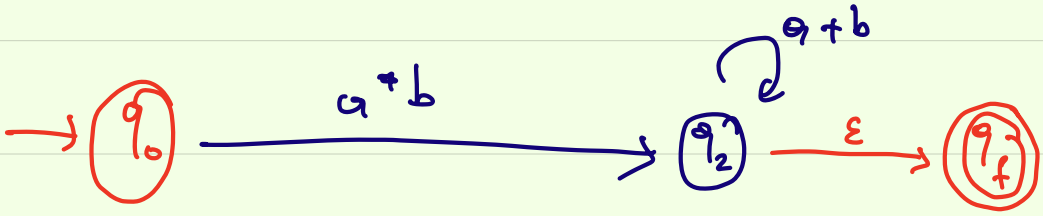
Eliminate q_1:



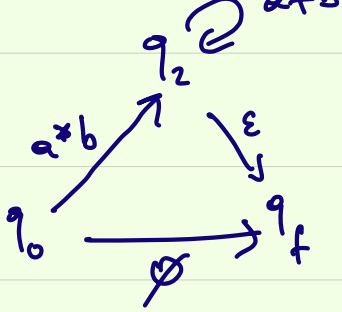
$\rightsquigarrow$ $q_0 \xrightarrow{\varepsilon \cdot a^* b + \emptyset} q_2$

i.e. $q_0 \xrightarrow{a^* b} q_2$

We get,

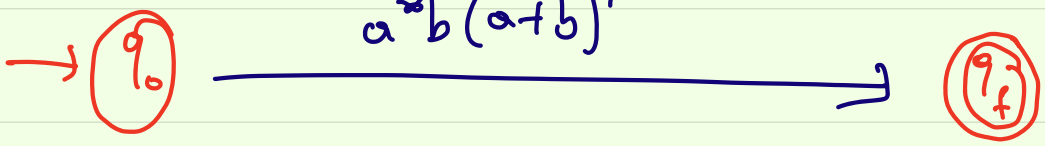


Eliminate q_2:



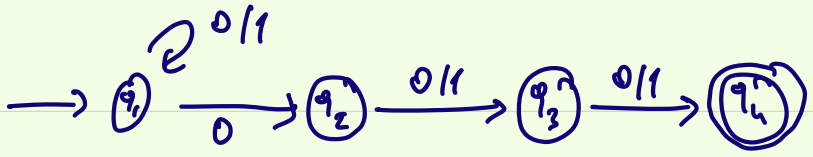
$\rightsquigarrow$ $q_0 \xrightarrow{a^* b (a+b)^* \cdot \varepsilon + \emptyset} q_f$

We get

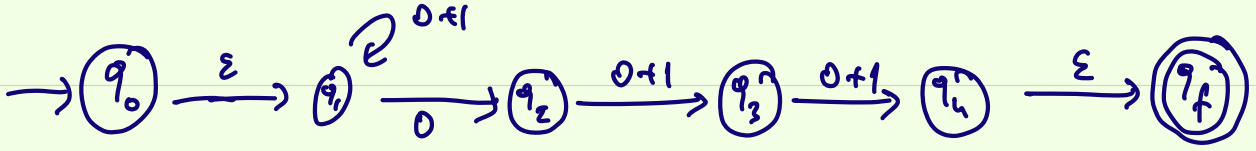


The equiv. RE is $a^*b(a+b)^*$.

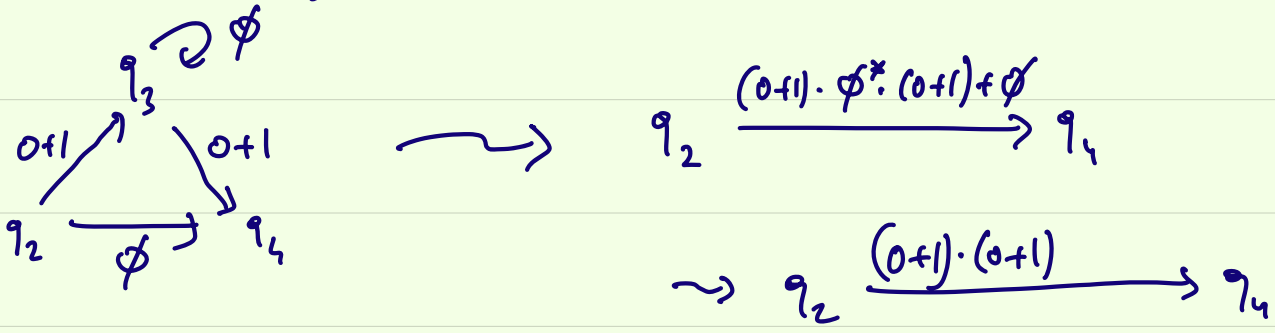
Example: N:

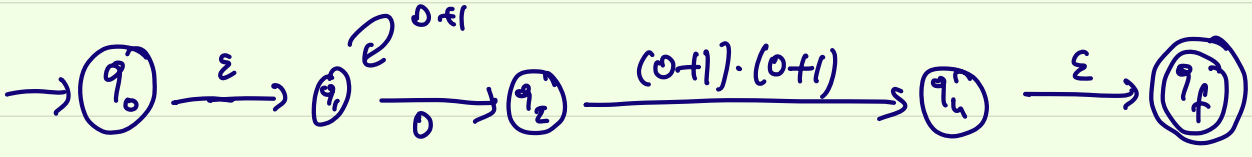


GNFA N':

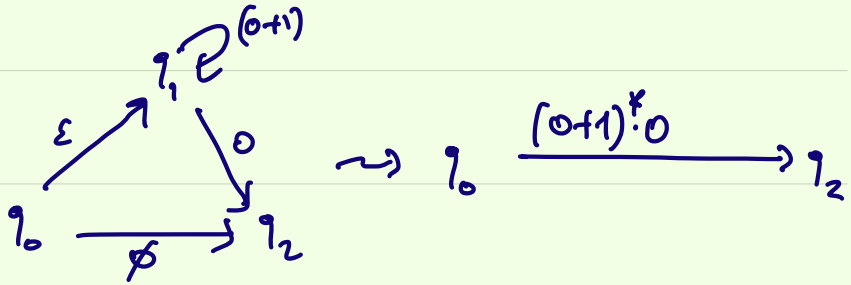


Eliminate q_3:

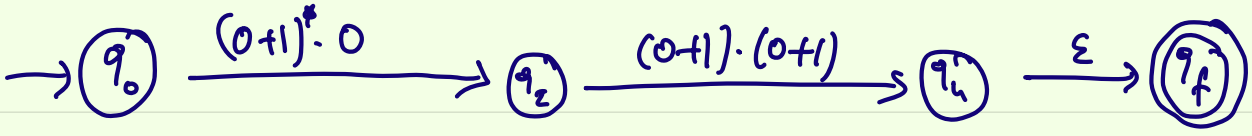




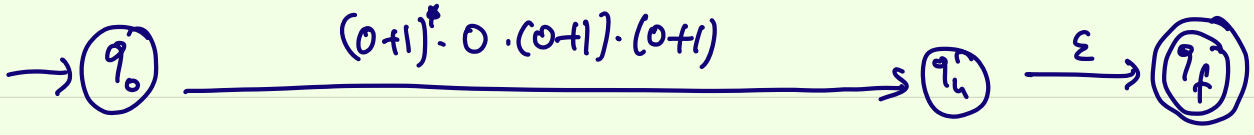
Eliminate q_1:



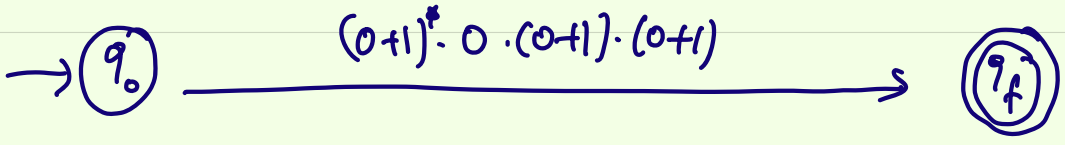
we get



Eliminate q_2:

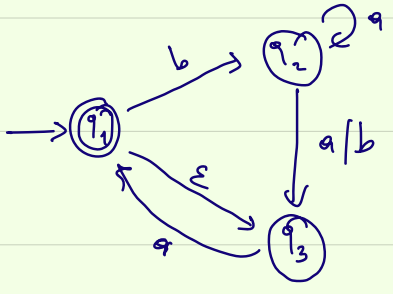


Eliminate q_4:



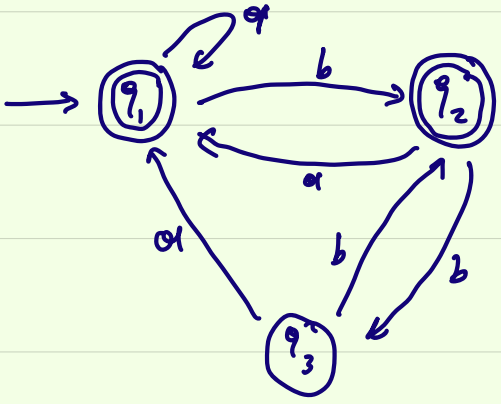
The equiv. RE $(0+1)^*\cdot 0\cdot(0+1)\cdot(0+1)$.

Exercise: The equiv. RE for the following NFA N:



Lecture 10

Example: M:

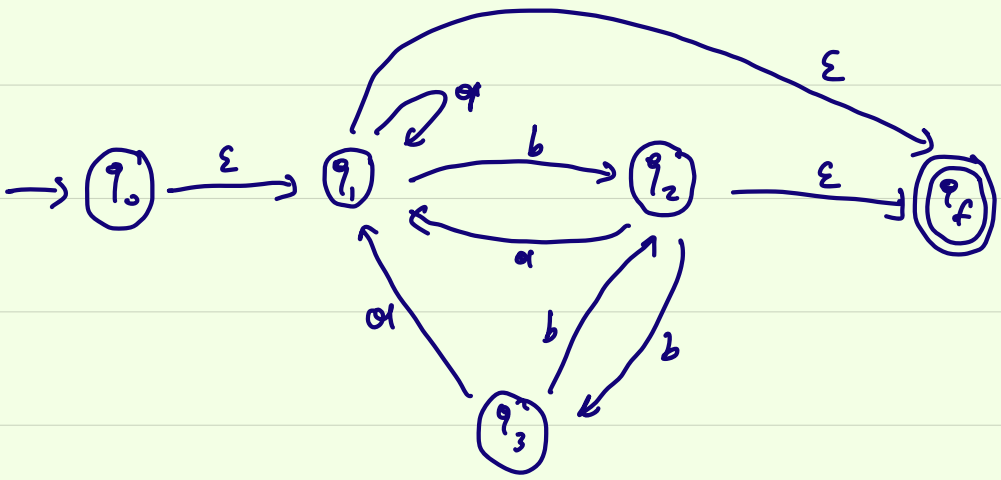


The equiv. RE?

note: $L(M) = L(M')$

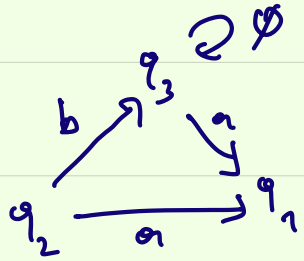
The equiv. GNFA M'

$\Sigma = \{a, b\}$



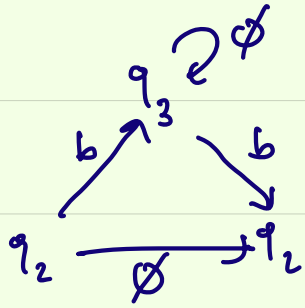
Eliminate q_3:

Recall: $\emptyset^* = \varepsilon$



$\rightsquigarrow$ $q_2 \xrightarrow{b \cdot \emptyset^* \cdot a \,+\, a} q_1$

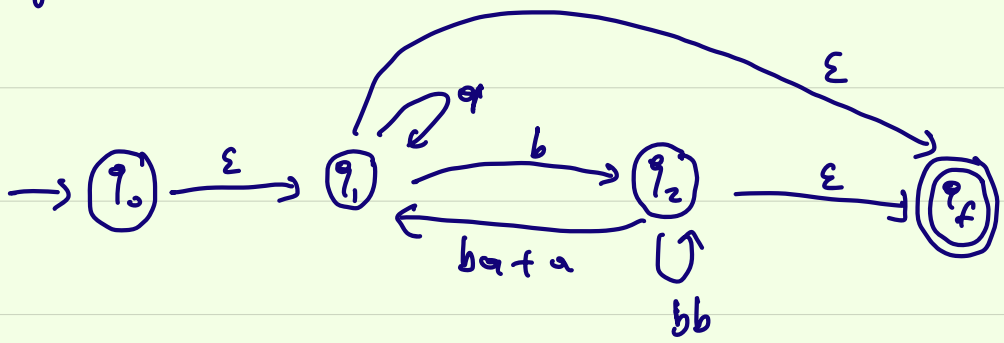
$\rightsquigarrow$ $q_2 \xrightarrow{ba \,+\, a} q_1$



$\rightsquigarrow$ $q_2 \xrightarrow{b \cdot \emptyset^* \cdot b \,+\, \emptyset} q_2$

$\rightsquigarrow$ $q_2 \xrightarrow{b \cdot b} q_2$

We get

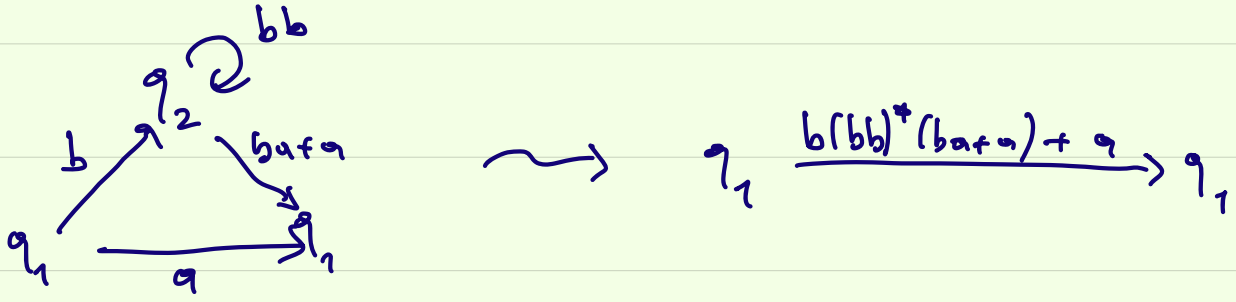


Eliminate q_2:

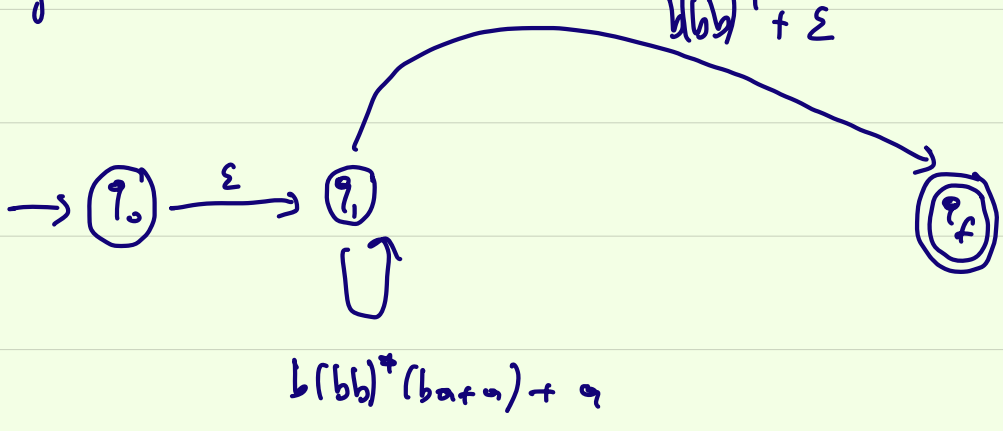
Note: $\alpha + \varepsilon \neq \alpha$

$\alpha + \phi = \alpha$



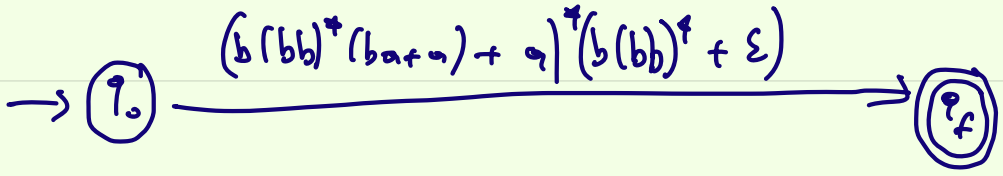


We get



Eliminate q_1:

We get



The equiv. RE is $(b(bb)^*(ba+a)+a)^*(b(bb)^*+\varepsilon)$.

The Pumping Lemma for Regular Languages

Q: How to prove that a language is regular?

A: Construct an NFA or a DFA or a RE that accepts it.

Q: How to prove that a language is not regular?

A: Use the pumping lemma.

Pumping Lemma (P.L.)

$p \Rightarrow q$

p { If a language $L \subseteq \Sigma^*$ is regular, then

q { there exists a positive constant $n > 0$ such that for any string $z \in L$ with $|z| \geq n$, z can be factorized as $z = u \cdot v \cdot w$ satisfying the following properties:

i) $|u \cdot v| \leq n$

ii) $|v| \geq 1$

iii) $\forall i \geq 0$, $u \cdot v^i \cdot w \in L$.

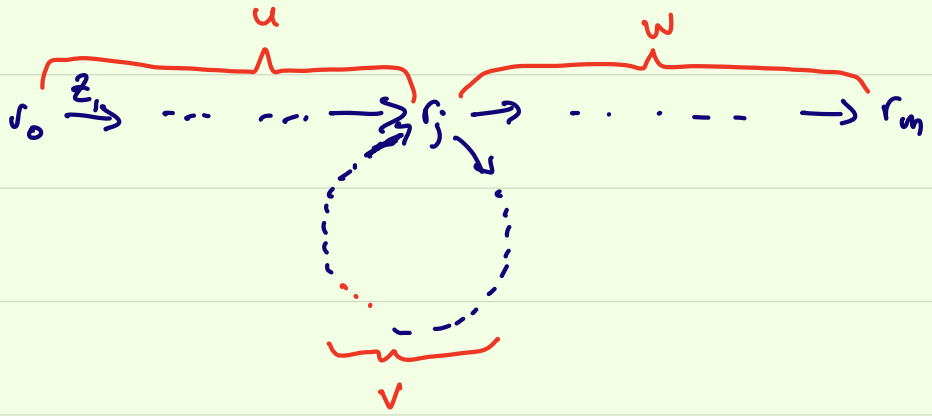
Recall: A computation of a DFA $M = (Q, \Sigma, \delta, q_0, F)$ on a string z has the following form:

$$r_0 \xrightarrow{z_1} r_1 \xrightarrow{z_2} r_2 \rightarrow \cdots\cdots\cdots \xrightarrow{z_m} r_m$$

Suppose there are n states in the DFA M.

i.e. $|Q| = n$.

If $|z| \geq n$, we can redraw the computation above as follows:



Note: P.L. is of the form $p \Rightarrow q$

i.e. it says

"If L is regular, then q holds.

(p)

We know that $p \Rightarrow q$ is logically equivalent to $\neg q \Rightarrow \neg p$

ie. "If q is not true for L, then L is not regular"

How to use Pumping Lemma to prove a language L is not regular:

1) Pick an arbitrary $n \geq 0$.
2) Choose a string $z \in L$ with $|z| \geq n$.

3) Consider a factorization of $t = u \cdot v \cdot w$ and $|u \cdot v| \leq n$ and $|v| \geq 1$.

4) Find an integer $i \geq 0$ s.t. $u \cdot v^i \cdot w \notin L$.

5) Conclude that L is not regular because "q is not true for L".

Example: Show that $L = \{0^k 1^k \mid k \geq 0\}$ is not regular.

Proof:

1) Let $n > 0$.

2) Let $z = 0^n 1^n$. Note $z \in L$ and $|z| \geq n$

3) Consider a factorization of $t = u \cdot v \cdot w$ and $|u \cdot v| \leq n$ and $|v| \geq 1$.

4) Let $i = 0$. and $|v| = \ell \geq 1$

Then $z' = u \cdot v^i \cdot w$ $\quad \ell \leq n$

$= u \cdot v^0 \cdot w$

$= u \cdot \varepsilon \cdot w$

$= u \cdot w$

$= 0^{n-\ell} 1^n$



$\Rightarrow n - \ell \neq n$ since $\ell \geq 1$

$\Rightarrow z' = 0^{n-\ell} 1^n \notin L$.

5) L is not regular because

"q is not true for L".

Example: $L = \{ 1^{k!} \mid k \geq 1 \}$ is not regular.

Proof:

1) Let $n > 0$.

2) Let $z = 1^{n!}$. Then $z \in L$ and $|z| \geq n$.

3) Consider a factorization of $z = u \cdot v \cdot w$

and $|u \cdot v| \leq n$ and $|v| \geq 1$.

4) Let $i = 2$. Then $z' = u \cdot v^2 \cdot w = u \cdot v \cdot v \cdot w$

and let $|v| = l$.

$\Rightarrow z' = 1^{n! + l}$ where $1 \leq l \leq n$

but $n! + l$ is not a factorial.

because $n! < n! + 1 \leq n! + n < (n+1)!$

$\Rightarrow z' \notin L$.

5) L is not regular.