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

$R = A \times B$ .. "

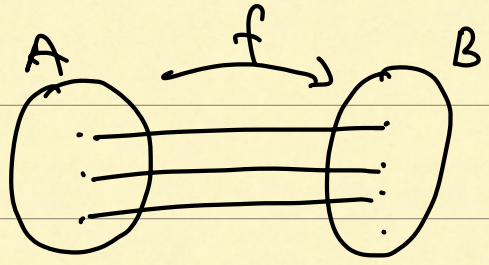
not 1-1 $\quad R = \{(1,b), (2,b)\}$ ✓

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

Exercise: How many functions from A to B?

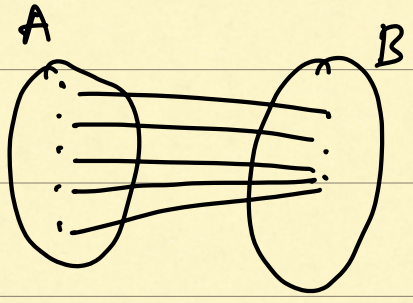
1-1 functions

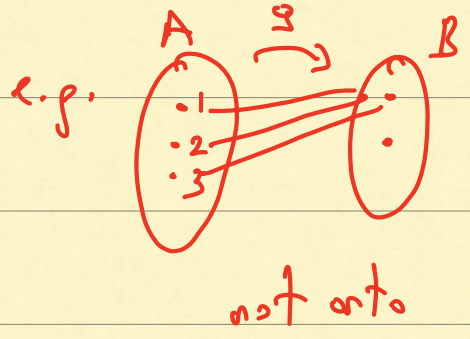
$f: A \to B$ is 1-1 if for every pair $a, b \in A$ we have $a \neq b \implies f(a) \neq f(b)$



Onto functions:

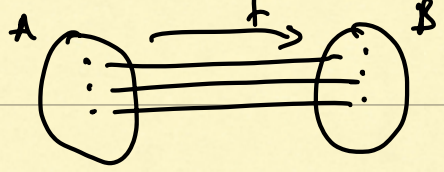
$f: A \to B$ is onto if for every $b \in B$ there exists $a \in A$ s.t. $f(a) = b$.

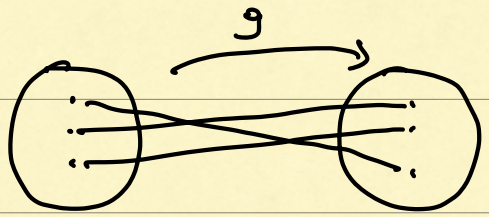




Bijections:

$f: A \to B$ 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

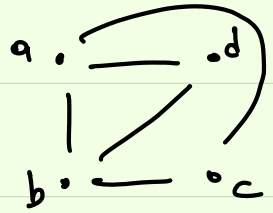
Review: reflexive, symmetric, transitive, antisymmetric

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.

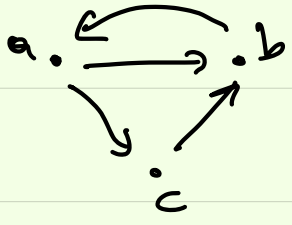


undirected graph

- 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.

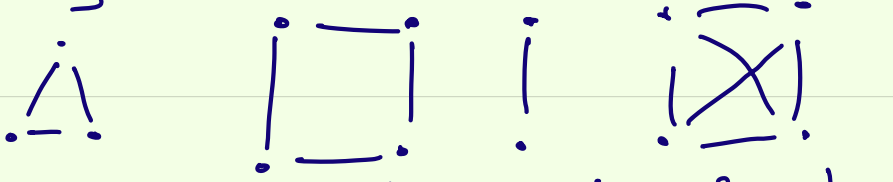
Exercise: - Prove that $\sqrt{2}$ is irrational.

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

Proof by Construction

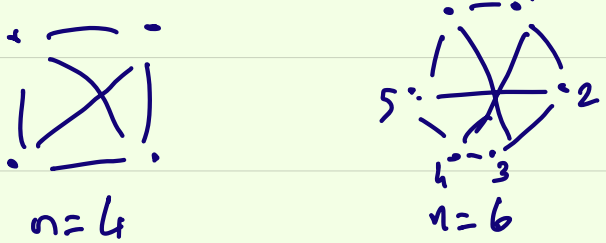
Example: A k-regular graph is a graph in which every node has degree k.

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

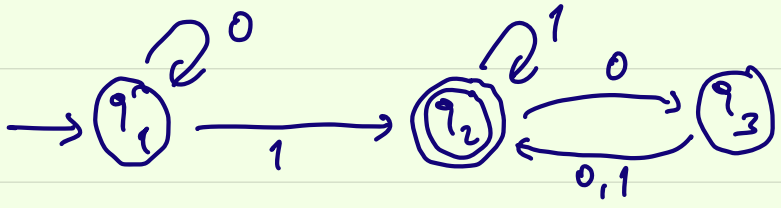
- Q is a set of states
- Σ is the input alphabet
- δ is a transition function:

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

- 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

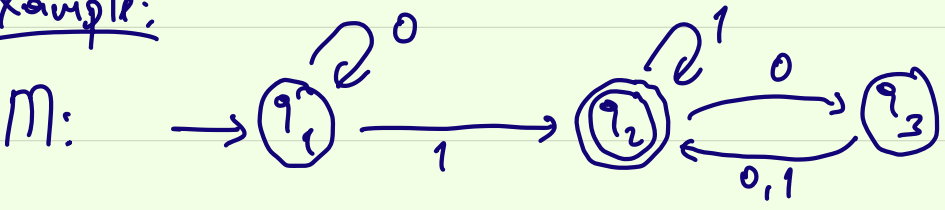
$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3$

$w_3 = \lambda$ ... 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 $L(M)$

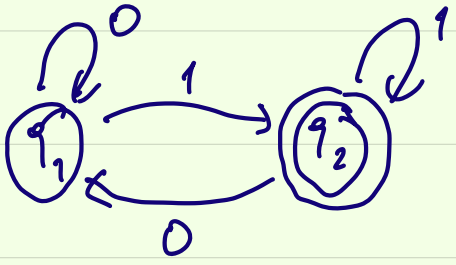
Rejected	Accepted	
λ	1	0100
0	01	01000
00	0101	010000
		0111000
		1110000

$L(M)$ = "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

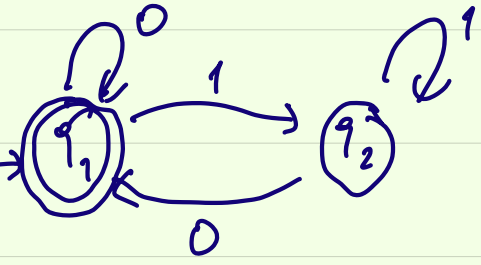
$0 \in L(M_2)$? No

$0 \ldots 0 \in L(M_2)$? 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.

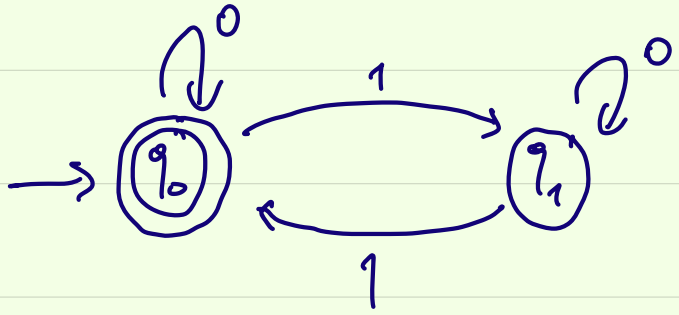
q_0 ... "even number 1's" final, initial

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

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

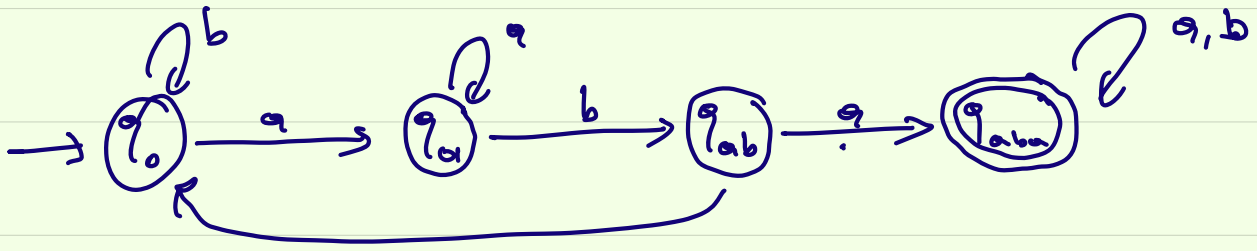
$aba \in L$, $aabab \in L$, ... $abbaba$

q_a ... "an a is seen"

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

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

q_0 ... initial state.



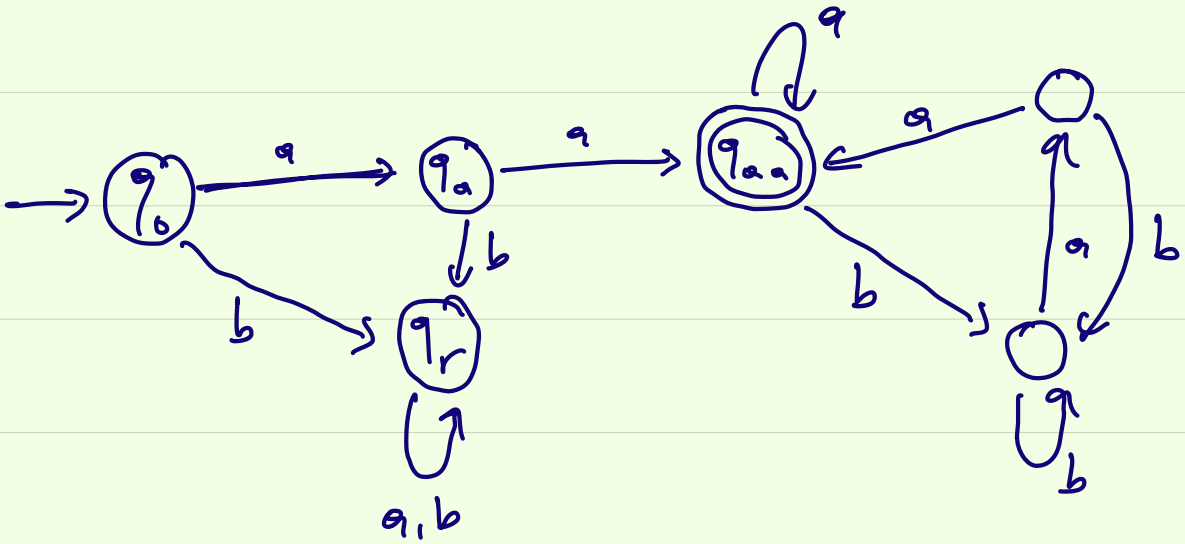
Exercise FA to accept all strings that have "bba" as a substring.

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 ✓

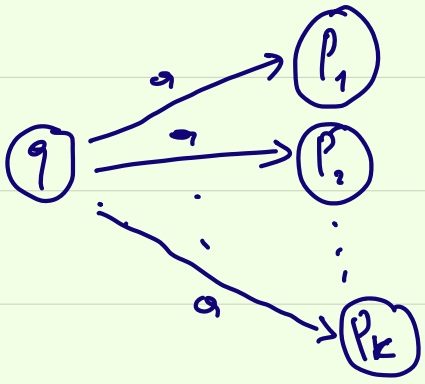
Nondeterministic Finite Automata (NFA)

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

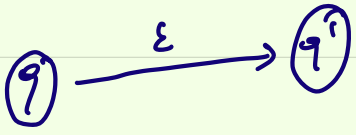
$$\delta(q, a) = q' \quad \text{-- unique.}$$

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. $\delta(q, a) = \{p_1, p_2, \ldots, p_k\} \subseteq Q$

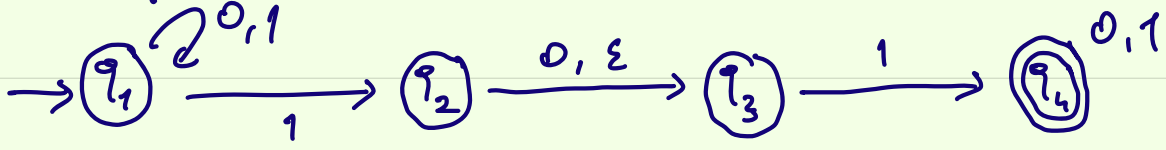


Also, in an NFA, there may be ε-transitions:



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

Example:

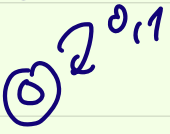
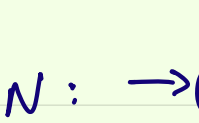


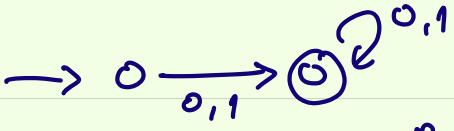
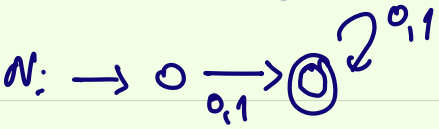
$$\delta(q_1, 1) = \{q_1, q_2\}$$
$$\delta(q_1, 0) = \{q_1\}$$
$$\delta(q_3, 0) = \emptyset$$

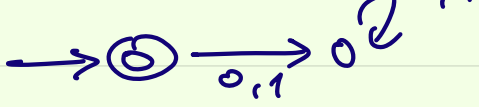
To be cont.

Example:

1) M:  N:

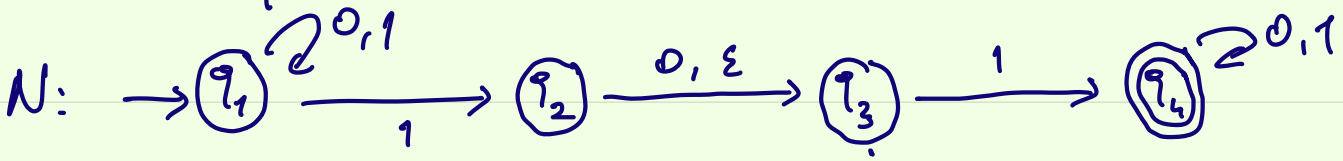
2) M:  N: 

3) M:  N: 

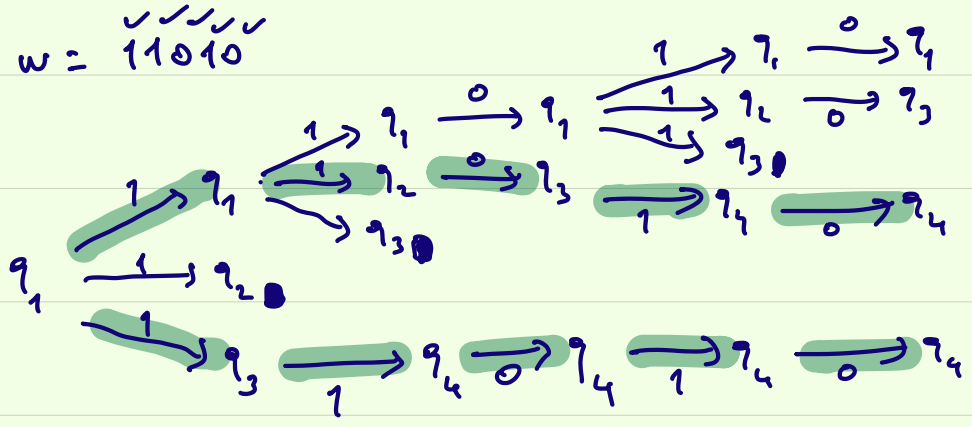
4) M:  N: 

Lecture 6

Example:



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



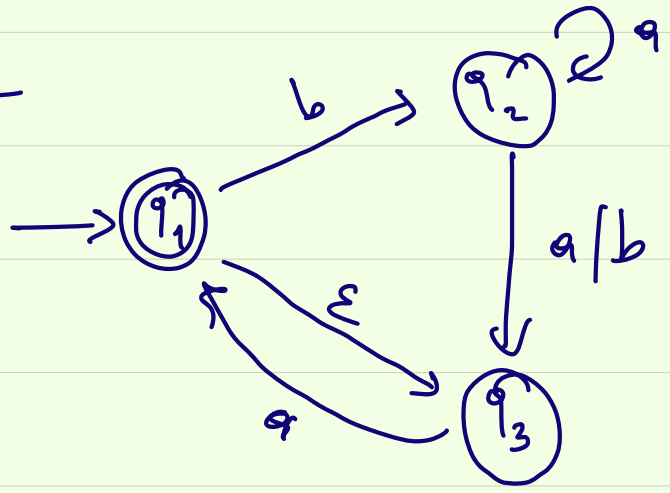
Exercise: Try 001101?

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

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

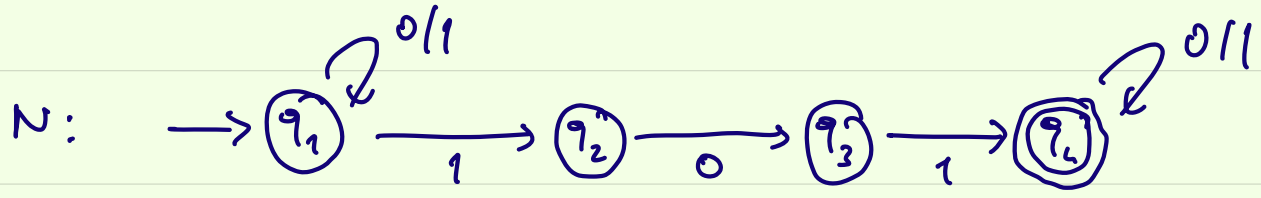
Proof of $p \Rightarrow q$: ($L = L(M) \ldots \Rightarrow L = L(N) \ldots$)

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

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

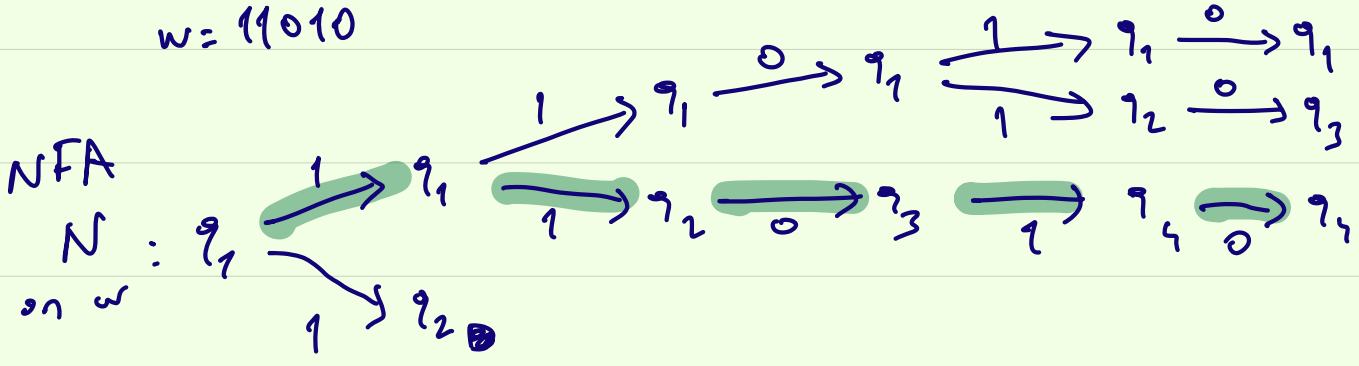
Idea: Given an NFA $N = (Q, \Sigma, \delta, q_0, F)$, construct a DFA $M = (Q', \Sigma, \delta', q_0', F')$ s.t. $L(N) = L(M)$.

For example, take the following NFA N:

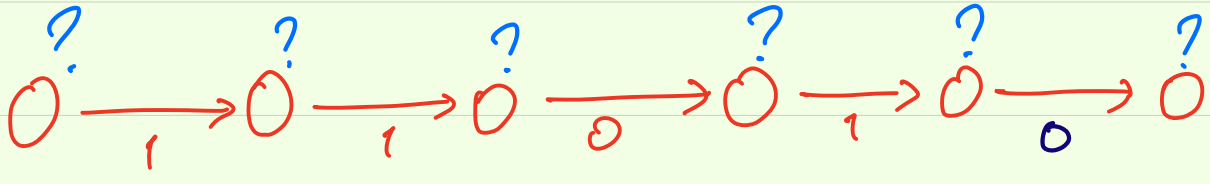


$w = 11010$

NFA N on w:

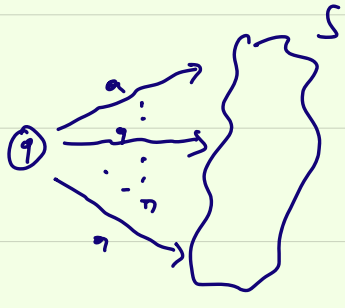


The equiv. DFA on w:

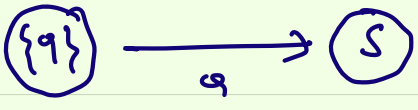


IDEA:

NFA:

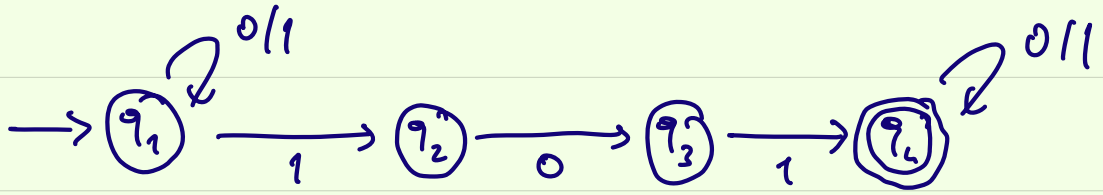


DFA:

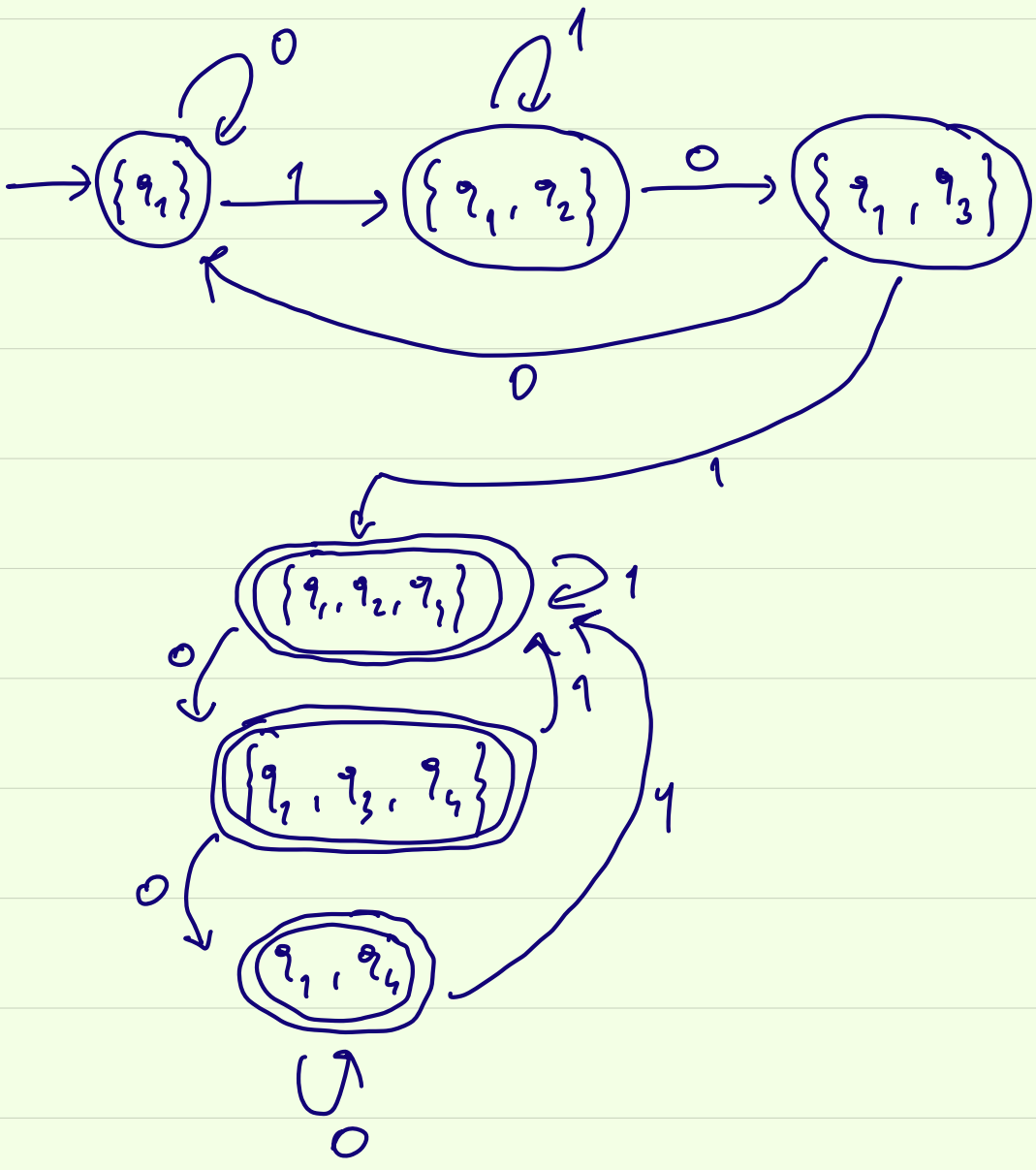


"subset construction"

NFA N:

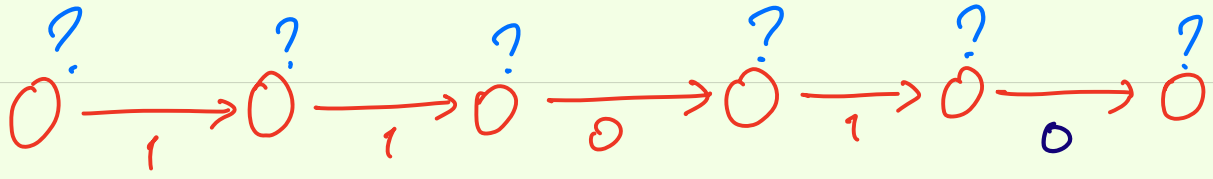


DFA M:



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.

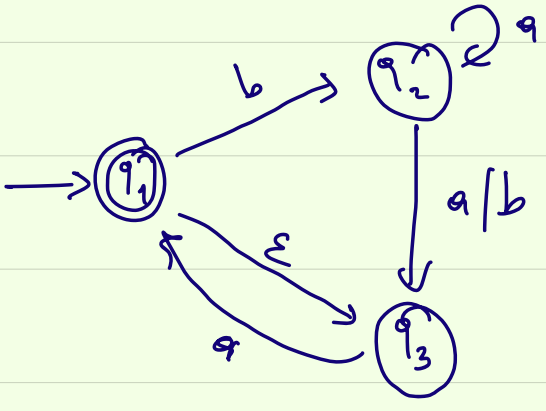
Q: What if an NFA has ε-transitions?

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.



$S = \{q_1\}$

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

$S = \{q_1, q_2\}$

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

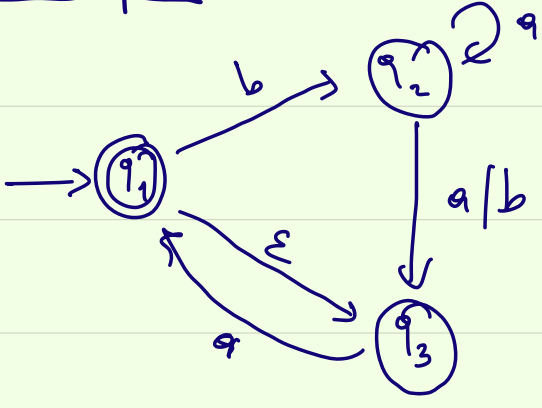
$S = \emptyset \Rightarrow E(S) = \emptyset$.

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

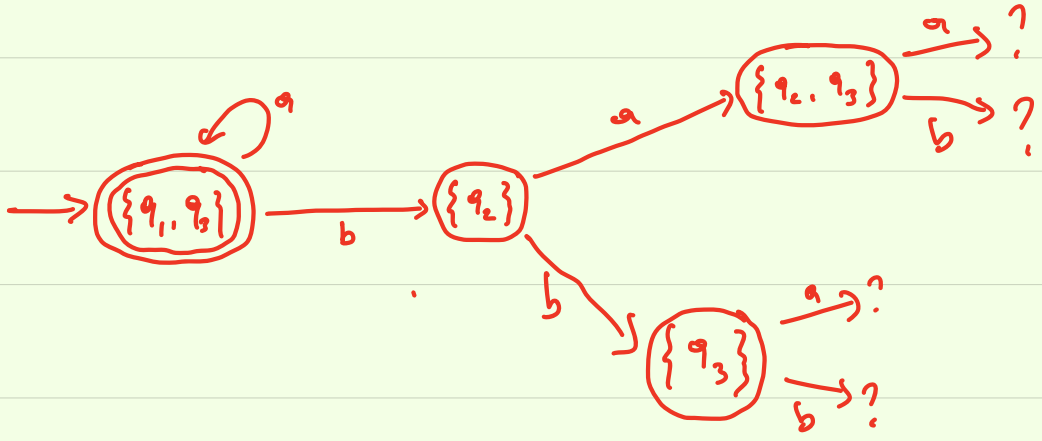
- $Q' \subseteq 2^Q$
- $q_0'' = E(\{q_0\})$
- $F' = \{s \in Q' \mid s \cap F \neq \emptyset\}$
- $\delta''(S, a) = \bigcup_{q \in S} E(\delta(q, a))$

—— ∞ End of Proof of Equivalence ——

Example:



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



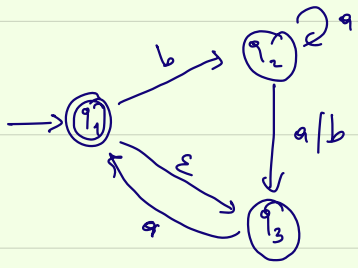
Exercise: Complete the construction.

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

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

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

$= \{q_1, q_3\}$



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

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

$= \{q_2\}$

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

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

$= \{q_2, q_3\}$

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

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

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:

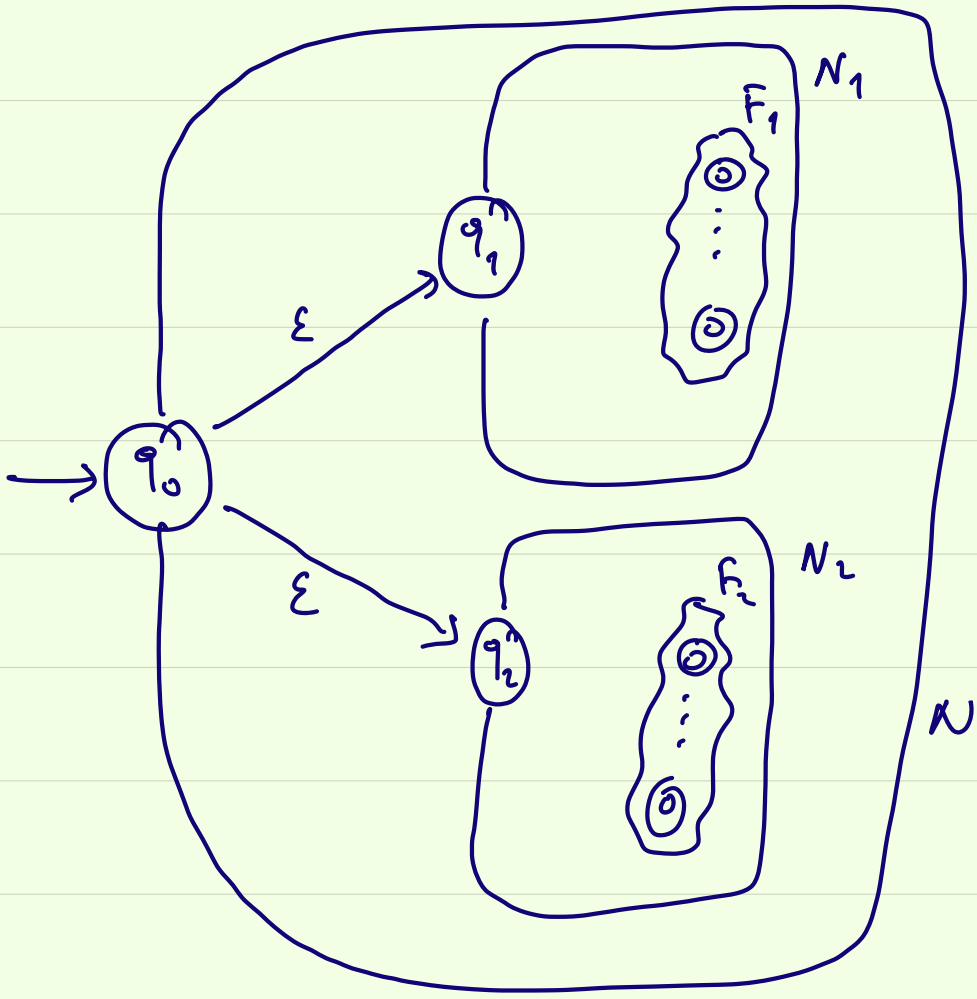
1) $\alpha = \alpha_1 + \alpha_2$ $\qquad L(\alpha) = L(\alpha_1) \cup L(\alpha_2)$

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

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

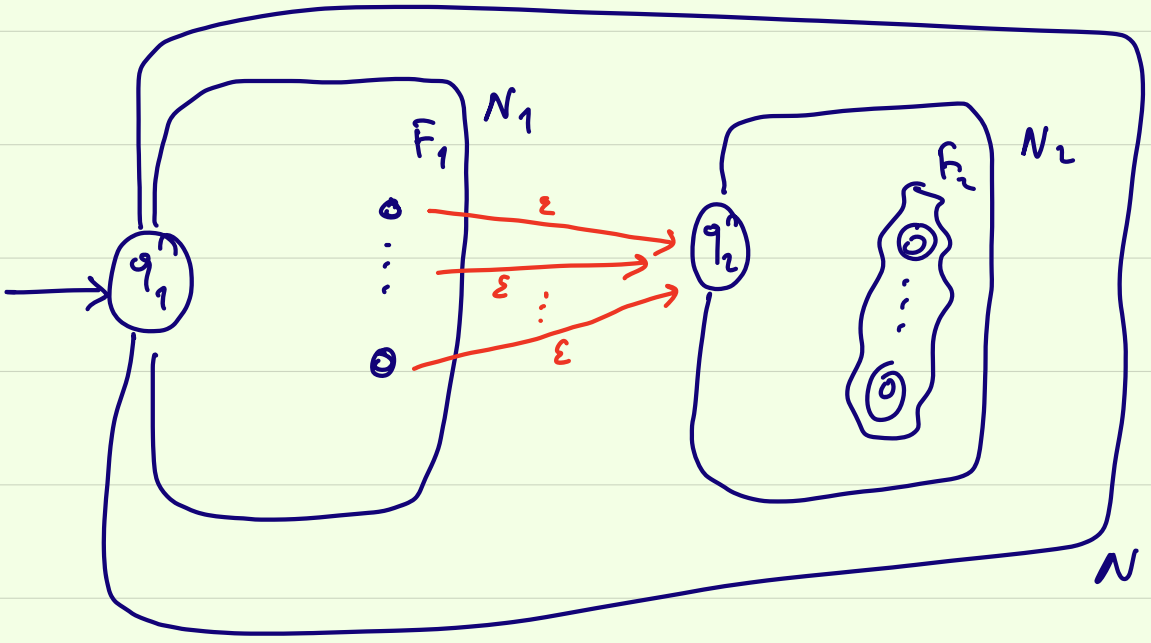
2) $\alpha = \alpha_1 \cdot \alpha_2$ $\qquad L(\alpha) = L(\alpha_1) \cdot L(\alpha_2)$

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

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

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

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



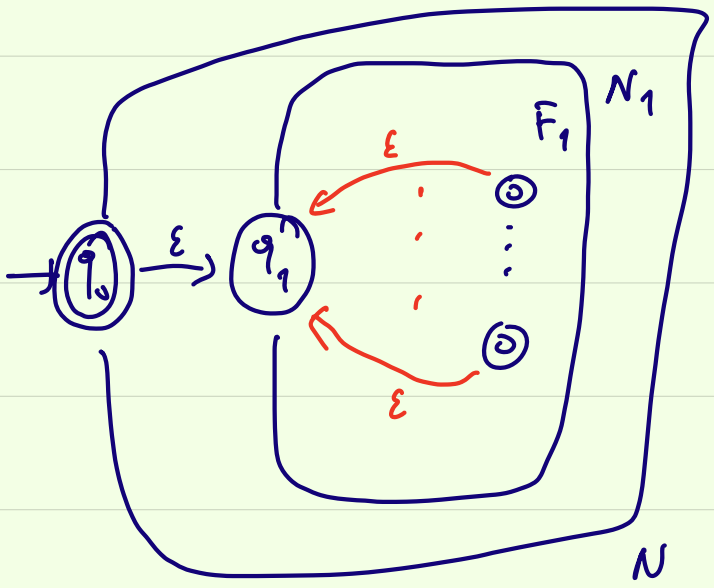
$$N = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F_2)$$

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

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



$w \in L(\alpha)$

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

$\Rightarrow w = x^n$

$n = 0, 1, \dots$

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

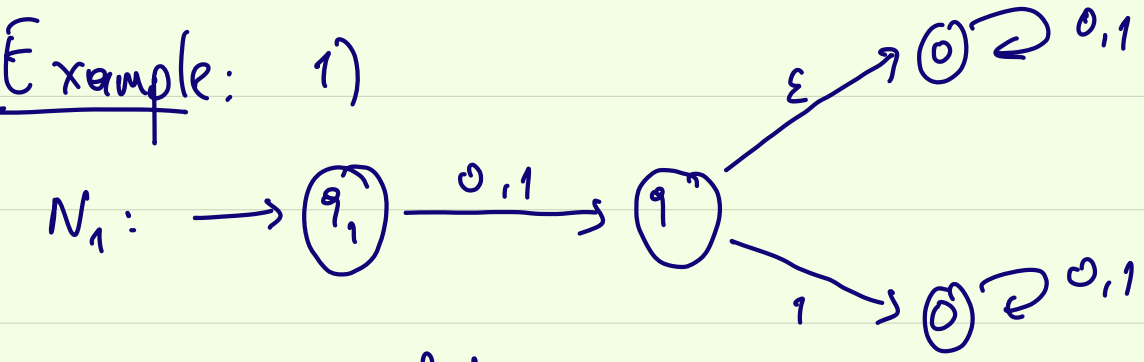
where

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

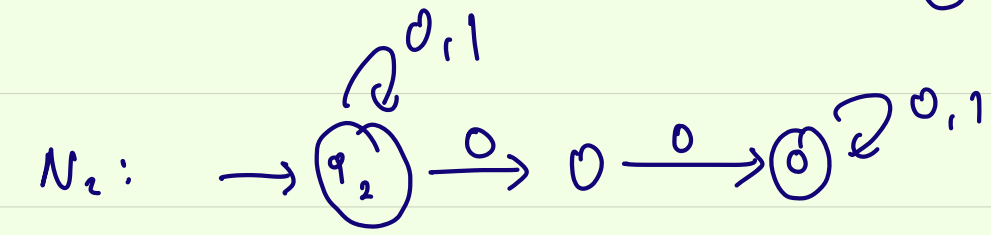
– End of proof of $q \Rightarrow p$ –

Example: 1)

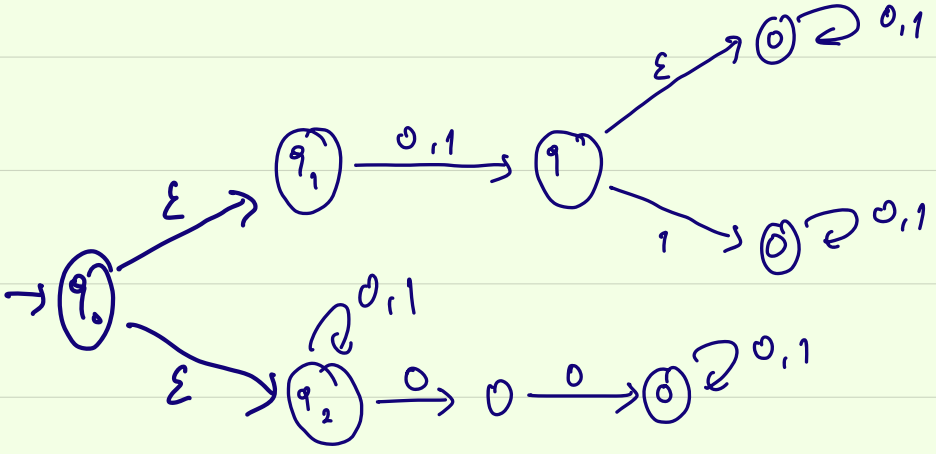
α_1 N_1:



α_2 N_2:

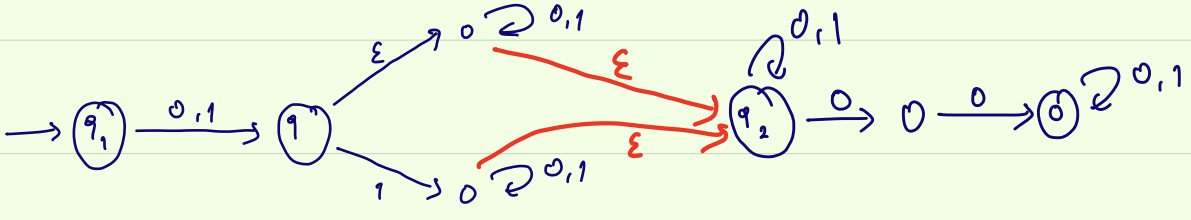


NFA for $\alpha = \alpha_1 + \alpha_2$

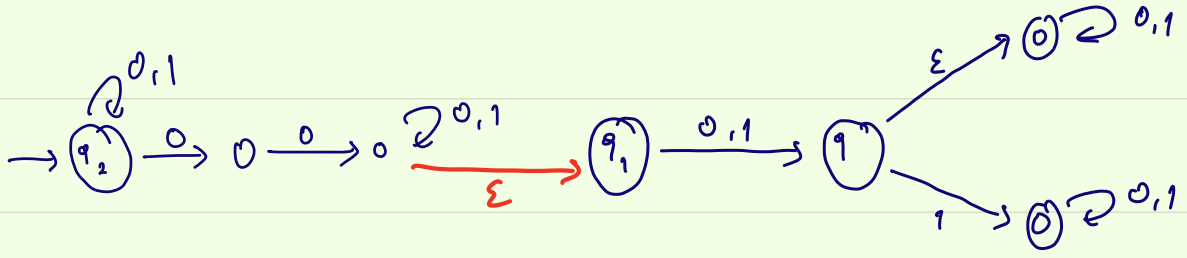


2) NFA for $\alpha = \alpha_2 + \alpha_1$? Same.

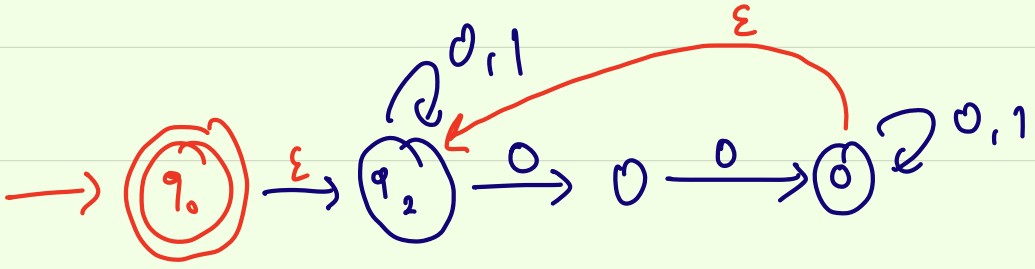
3) NFA $\alpha = \alpha_1 \cdot \alpha_2$



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



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



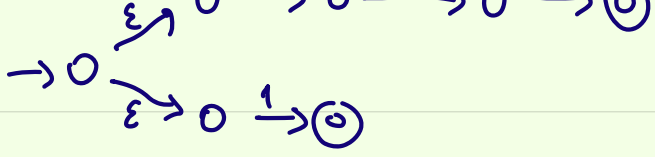
6) $\alpha = (00+1)^*0 + \varepsilon$ $\quad \Sigma = \{0,1\}$

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

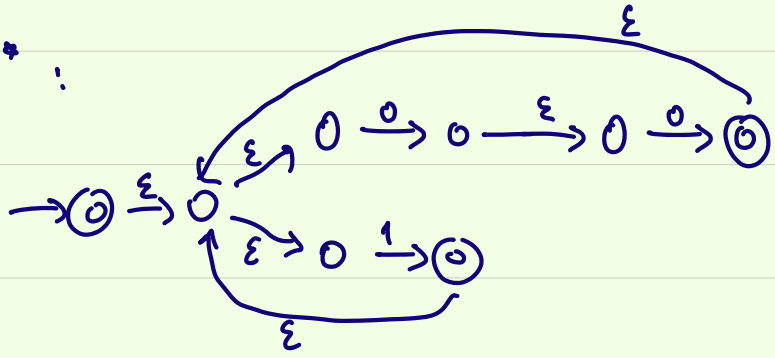
For 00: $\rightarrow 0 \xrightarrow{0} 0 \xrightarrow{\varepsilon} 0 \xrightarrow{0} ((0))$

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

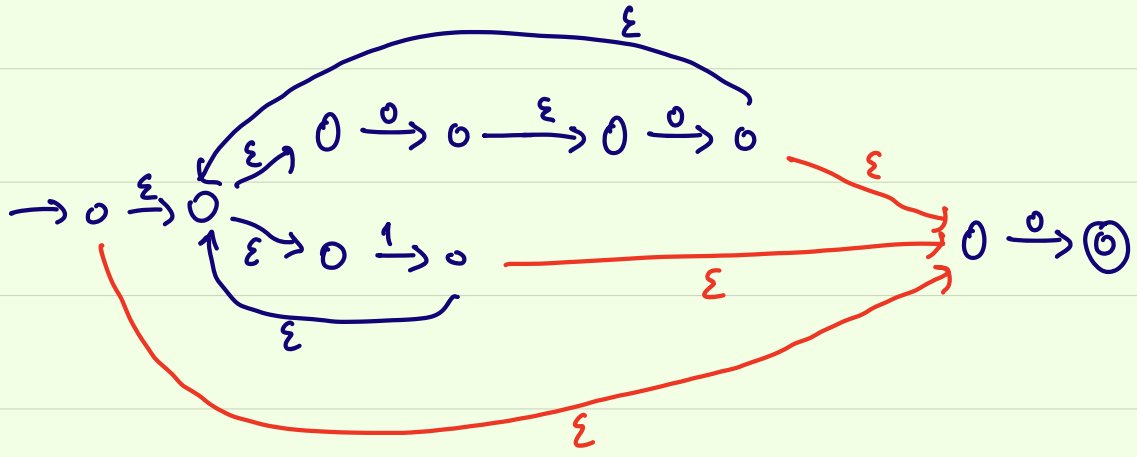
For 00+1 :



For $(00+1)^*$:

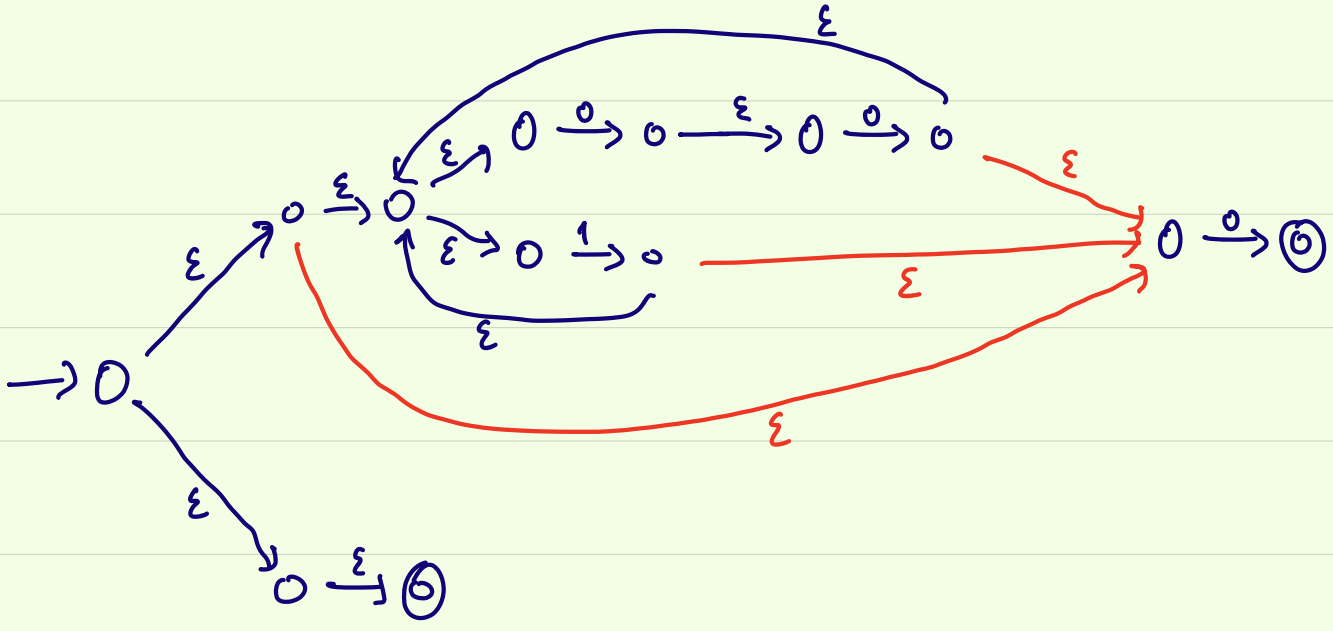


For $(00+1)^*.0$



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)

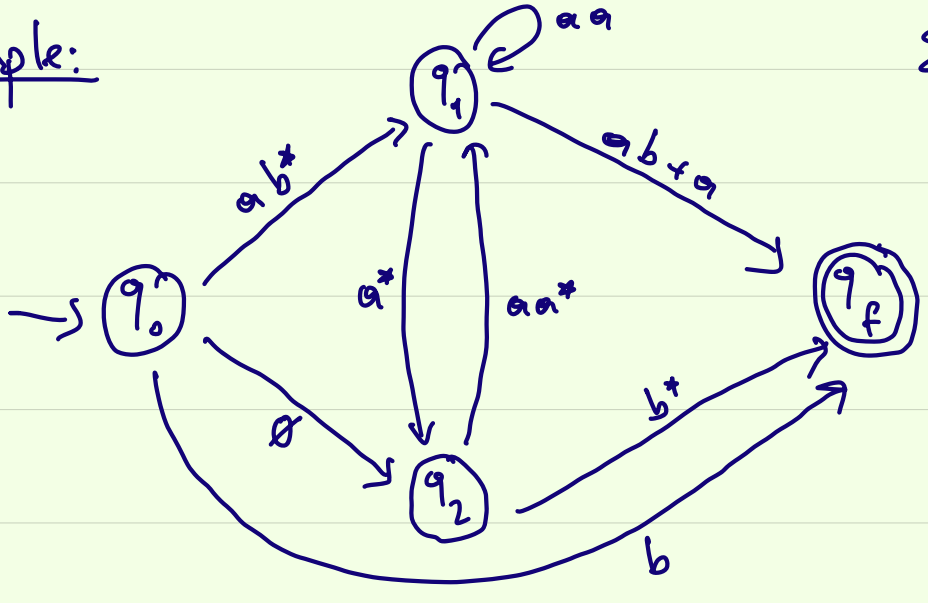
A GNFA has

- 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:

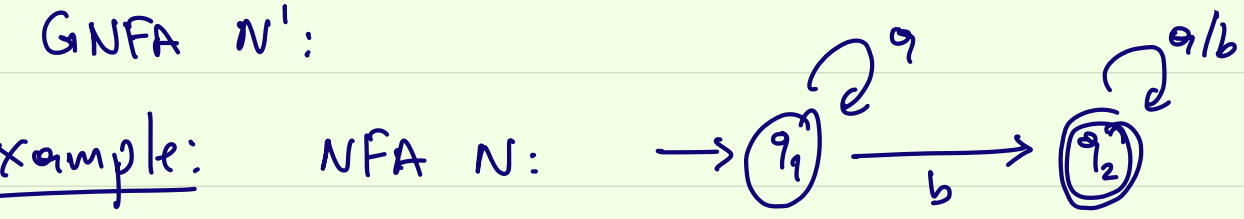
Outline:

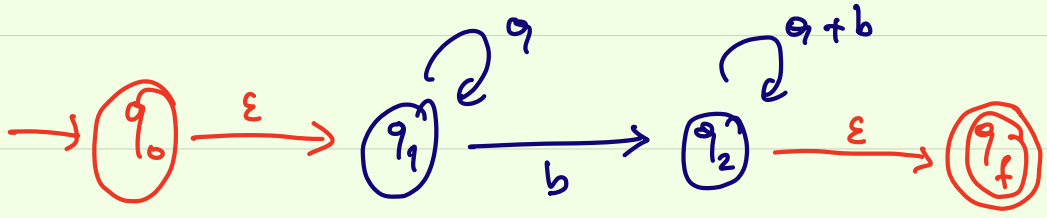
NFA N $\xrightarrow{\text{convert}}$ GNFA N' $\xrightarrow[\text{states}]{\text{Eliminate}}$ GNFA N'' $\longrightarrow$ The equiv. R.E. α

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

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:

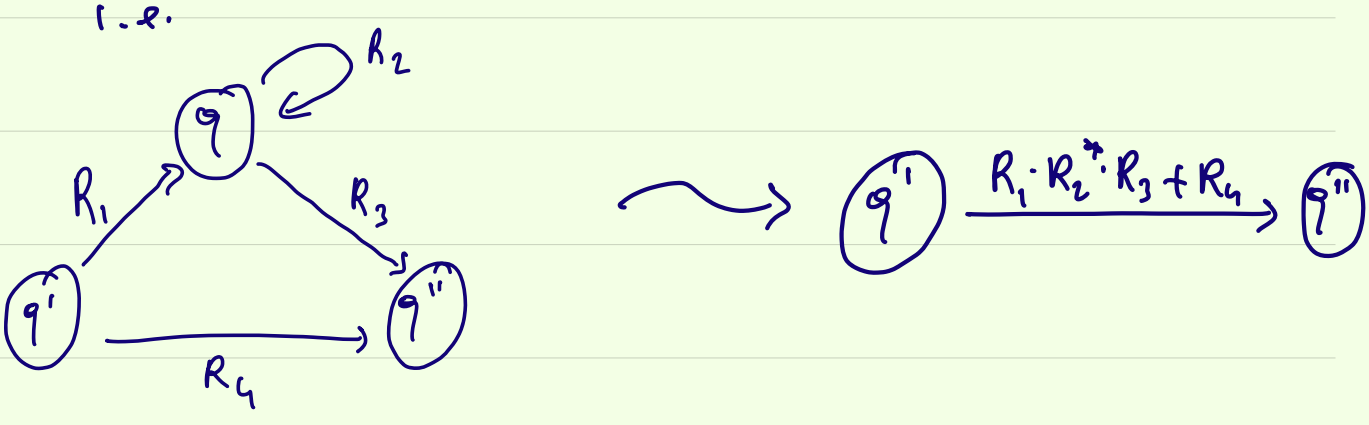
Repeat the following until there are two states left (q_0 and q_f namely).

- 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 $q'' \in Q - \{q, q_0\}$

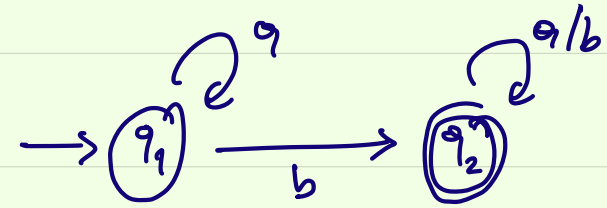
label the transition from q' to q'' with $R_1 \cdot R_2^* \cdot R_3 + R_4$ where

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

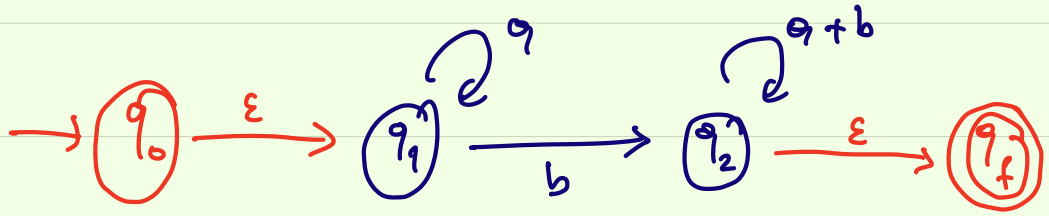
i.e.



Example: NFA N:

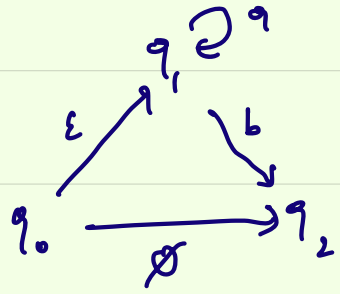


N':



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

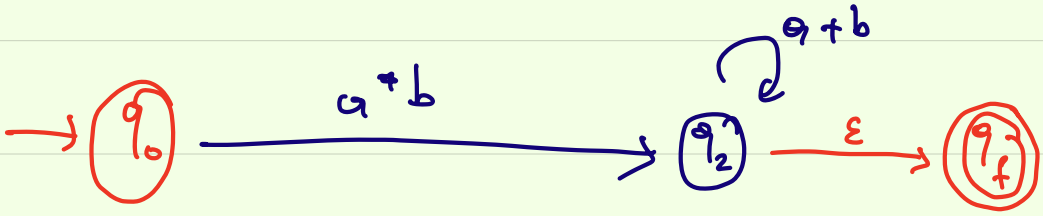
Eliminate q_1:



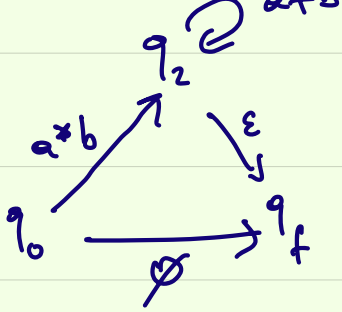
$\rightsquigarrow$ $q_0 \xrightarrow{\varepsilon \cdot a^* b + \emptyset} q_2$

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

We get,

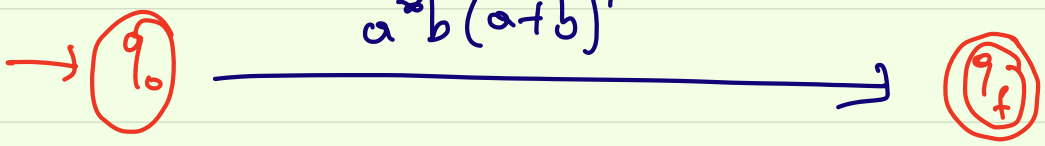


Eliminate q_2:



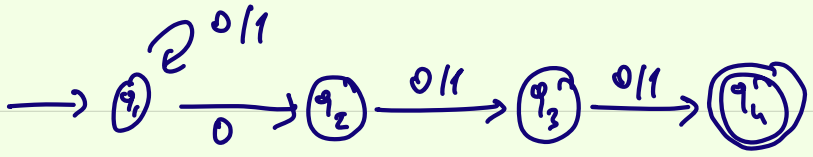
$\rightsquigarrow$ $q_0 \xrightarrow{a^* b (a+b)^* \cdot \varepsilon + \emptyset} q_f$

We get

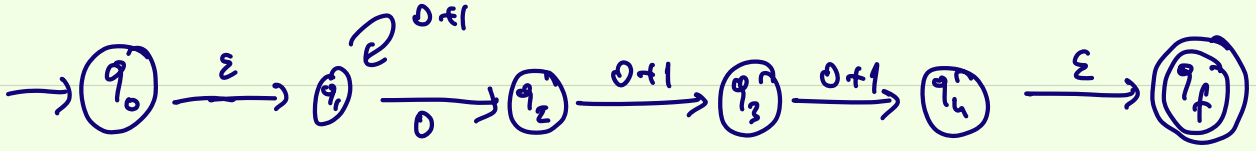


The equiv. RE is $a^*b(a+b)^*$.

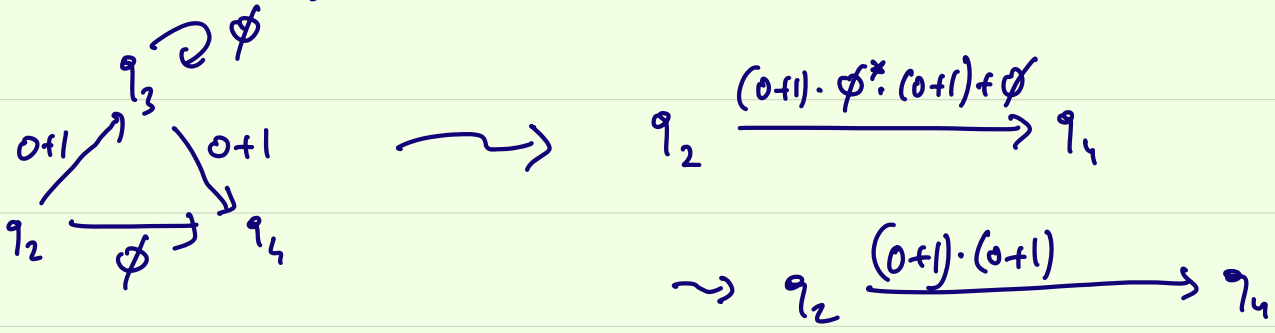
Example: N:

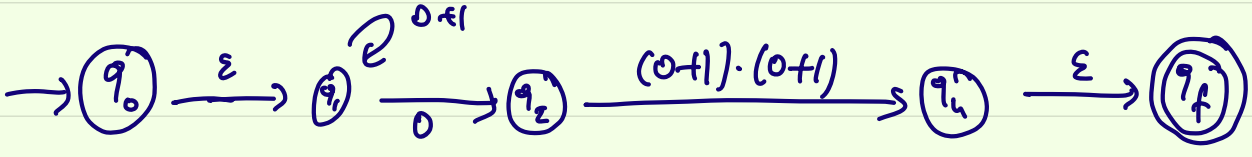


GNFA N':

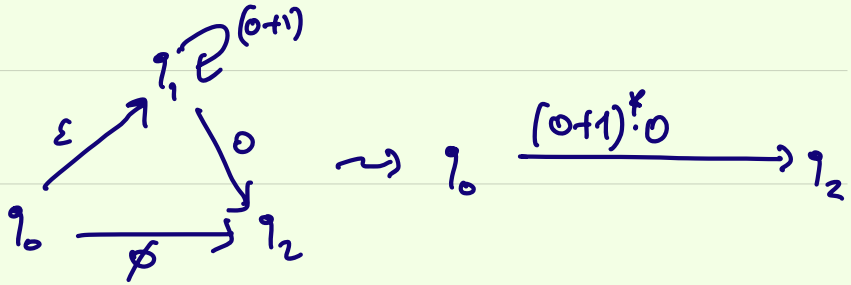


Eliminate q_3:

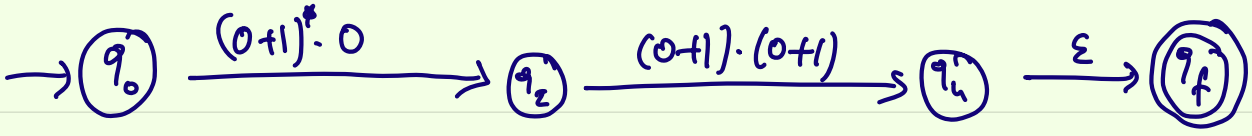




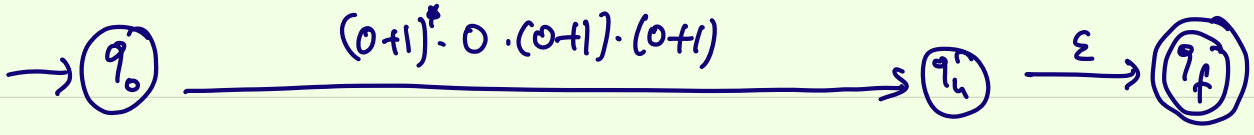
Eliminate q_1:



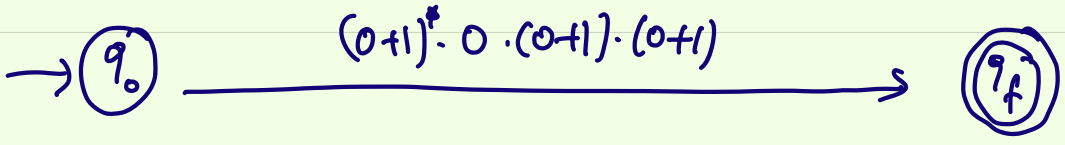
we get



Eliminate q_2:

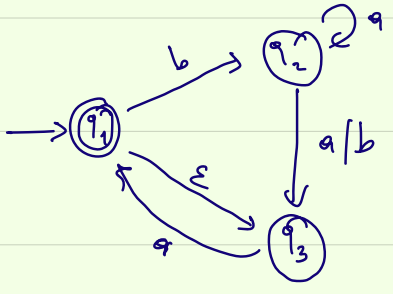


Eliminate q_4:



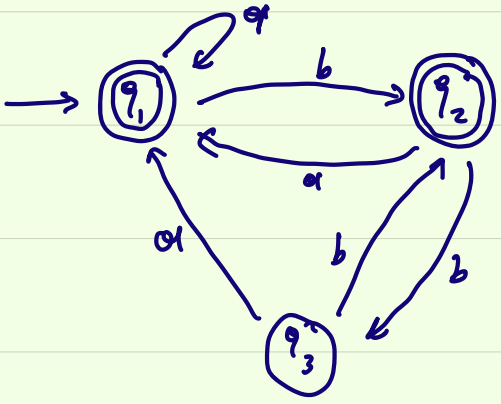
The equiv. RE $(0+1)^*\cdot 0\cdot(0+1)\cdot(0+1)$.

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



Lecture 10

Example: M:

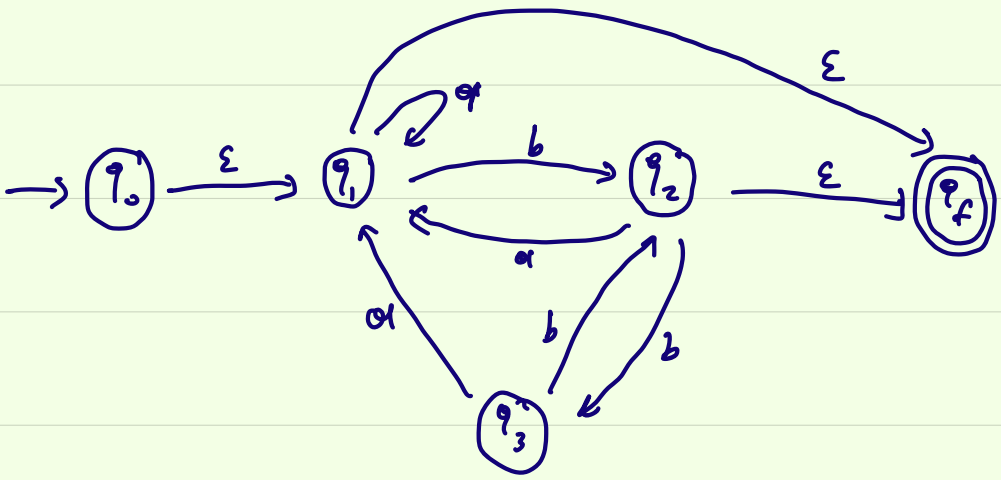


The equiv. RE?

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

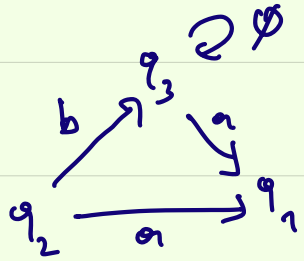
The equiv. GNFA M'

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



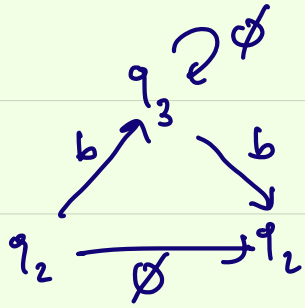
Eliminate q_3:

Recall: $\emptyset^* = \varepsilon$



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

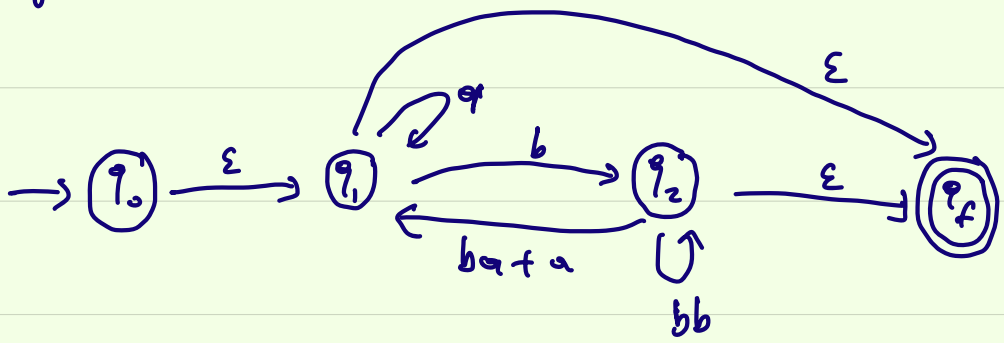
$\rightsquigarrow$ $q_2 \xrightarrow{ba \,+\, a} q_1$



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

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

We get

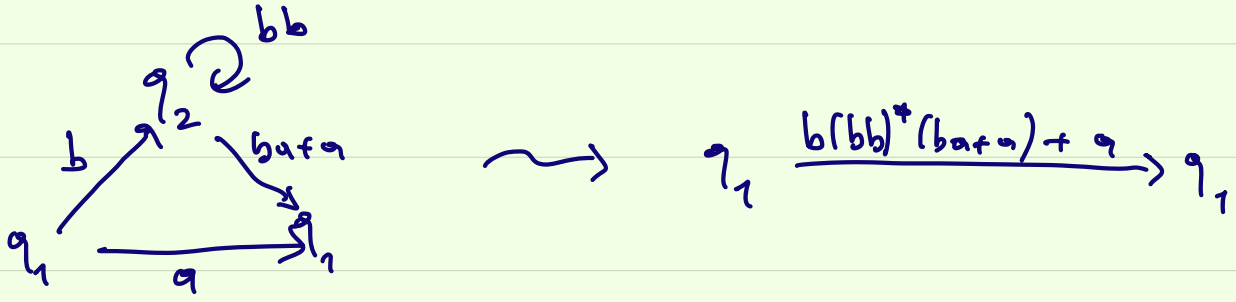


Eliminate q_2:

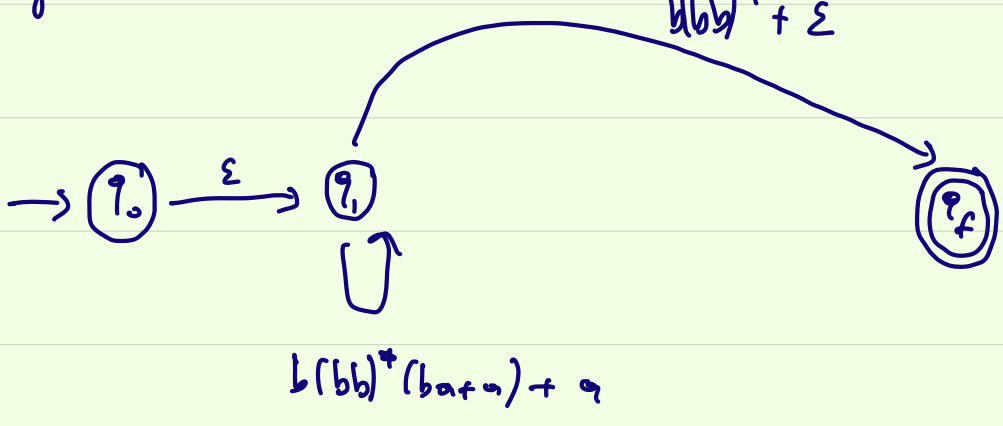
Note: $\alpha + \varepsilon \neq \alpha$

$\alpha + \phi = \alpha$



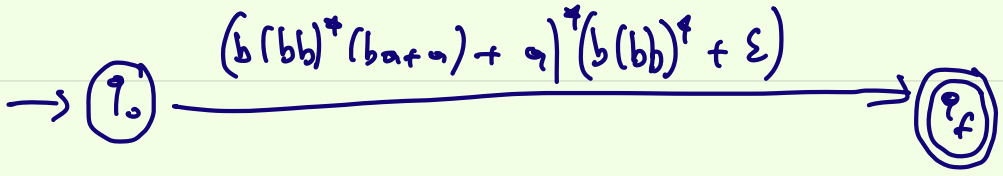


We get



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

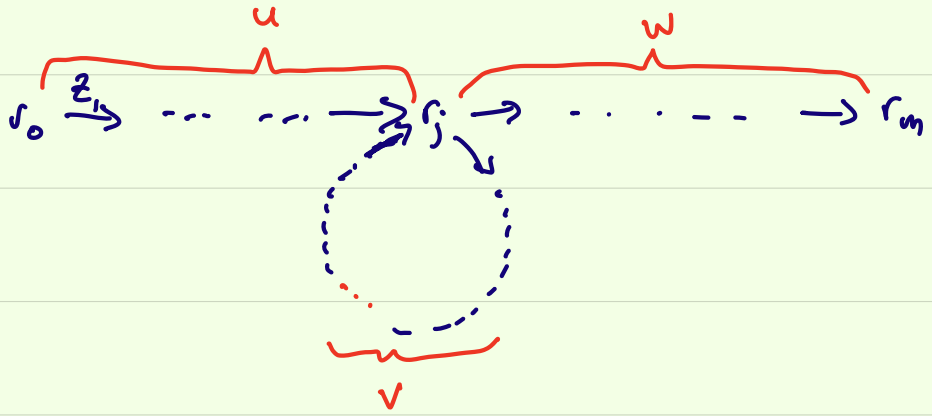
Recall: A computation of a DFA $M = (Q, \Sigma, \delta, q_0, F)$ on a string z has the following form:

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

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.

Lecture 11

Example: Prove $L = \{ 0^{k^2} \mid k \geq 1 \}$ is not regular.

1) Let $n > 0$.

2) Let $z = 0^{n^2}$. Then $z \in L$ and $|z| \geq n$.

3) Consider a factorization $z = u \cdot v \cdot w$ s.t.

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

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

and $|v| = \ell$.

$\Rightarrow |z'| = |u \cdot v^2 \cdot w| = |u \cdot v \cdot w| + |v| = n^2 + \ell$

$n^2 < n^2 + \ell \leq n^2 + n < n^2 + 2n + 1 = (n+1)^2$

$\Rightarrow n^2 < n^2 + \ell < (n+1)^2$

$\Rightarrow n^2 + \ell$ is not a perfect square.

$\Rightarrow z' \notin L$.

5) L is not regular.

Exercise: In step 4, try $i = 0$.

Example: Prove $L = \{ ww \mid w \in \{0,1\}^* \}$ is not regular.

1) Let $n > 0$.

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

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

4) Let $i = 2$. Then $z' = u \cdot v^2 \cdot w =$
If $|v| = \ell$, then $z' = 0^{n+\ell} 1 0^n 1$.
Note: $\ell \geq 1$.
$\Rightarrow n + \ell \neq n$.
$\Rightarrow z' \notin L$.

5) L is not regular.

Example: Show that $L = \{ 0^i 1^j \mid i > j \}$ is not regular.

1) Let $n > 0$.

2) Let $z = 0^{n+1} 1^n$. Then $z \in L$ and $|z| \geq n$.

3) Consider a factorization $z = u \cdot v \cdot w$ s.t.

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

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

$\Rightarrow z' = u \cdot v^0 \cdot w = u \cdot w = 0^{n+1-\ell} 1^n$.

Note that $n+1-\ell \leq n$ since we have $\ell \geq 1$.

$\Rightarrow$ For z', $|z|_0 > |z|_1$ is not true.

$\Rightarrow z' \notin L$.

5) L is not regular.

Closure Properties of Regular Languages

Proposition: Regular languages are closed under union, concatenation and Kleene star operation.

Proof: Recall the definition of regular expressions and the proof equivalence between NFAs and RE. The result follows by def.

Proposition: Regular languages are closed under complement. (i.e. if L is regular, then so is $\bar{L}$)

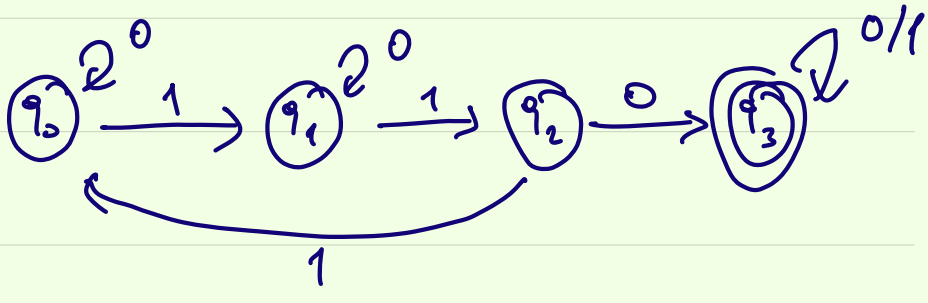
Proof: If M is a DFA for L, then we construct a DFA M' using M to accept $\bar{L}$ as follows:

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

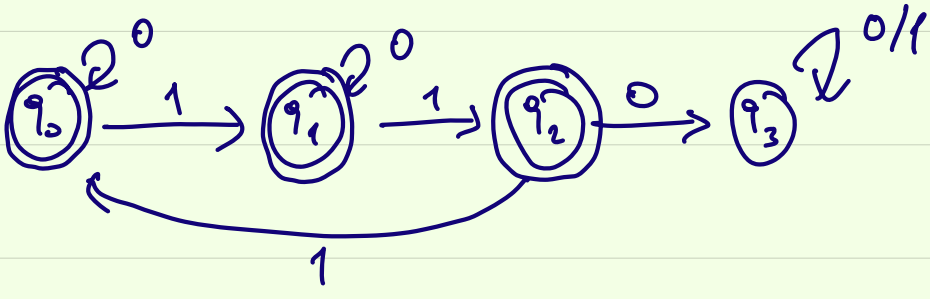
$$\Rightarrow M' = (Q, \Sigma, \delta, q_0, Q - F)$$

(i.e. make final states non final and non final states final in M')

e.g. M: L



M':



$\bar{L} = \Sigma^* - L$

Proposition Regular languages are closed under intersection. (i.e. if L_1, L_2 are reg., so is $L_1 \cap L_2$)

Proof: L_1, L_2 ... regular.

$$L_1 \cap L_2 = \overline{\overline{L_1 \cap L_2}}$$

$$= \overline{\overline{L_1} \cup \overline{L_2}}$$

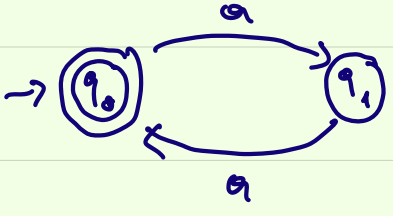
(Recall: $\overline{\overline{A}} = A$, $\overline{A \cap B} = \overline{A} \cup \overline{B}$)

We are done because $\overline{L_1}, \overline{L_2}$ are regular and reg. lang. are closed under union.

e.g. $L_1 = \{ a^{2n} \mid n \geq 0 \}$ $\quad \Sigma = \{a\}$

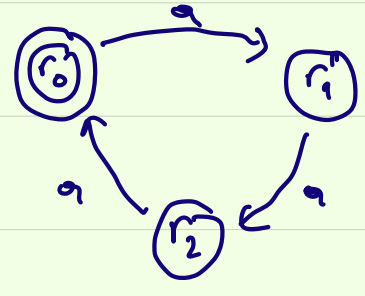
$L_2 = \{ a^{3n} \mid n \geq 0 \}$

M_1:



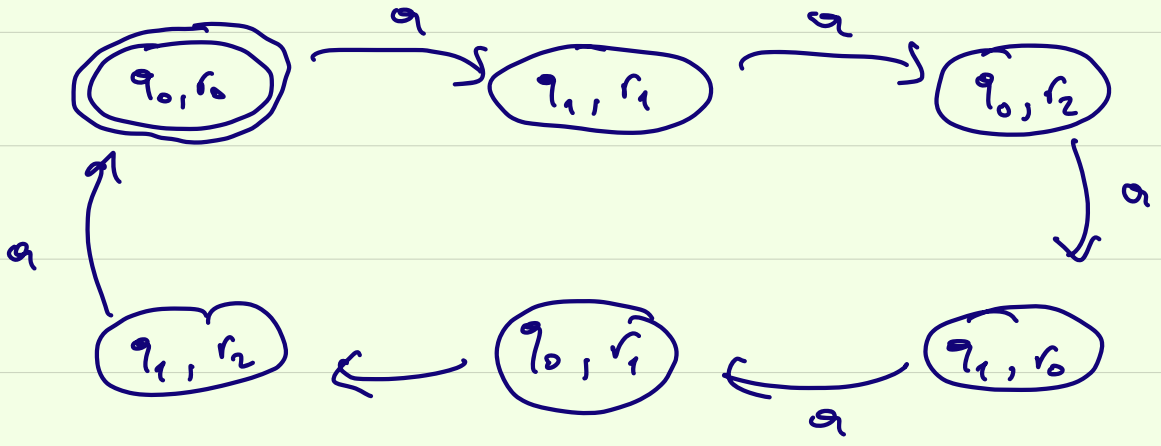
$Q_1 = \{q_0, q_1\}$

M_2:



$Q_2 = \{r_0, r_1, r_2\}$

M_3 for $L_1 \cap L_2$ ... $Q_3 = Q_1 \times Q_2$



$$L_3 = L_1 \cap L_2 = \{ a^{6n} \mid n \geq 0 \}$$

Example: Prove that $L = \{ 0^i 1^j \mid i \neq j \}$ is not regular

- We know that $L' = \{ 0^i 1^j \mid i = j \}$ is not regular.
- Suppose L is regular.

  $L \cap L(0^*1^*) = L'$ should be regular due to closure under complement and intersection. That's a contradiction.

  L' is known to be nonregular.

  $\Rightarrow$ L is not regular.

Example: Let $L \subseteq \Sigma^*$ regular lang. and $w \in \Sigma^*$

Prove that $L' = \{x \in L \mid x = \Sigma^* w \Sigma^*, |x| = 2k\}$

is regular.

Proof: $L_1 = \{\Sigma^* w \Sigma^* \mid w \in \Sigma^*\}$ ... regular

$L_2 = L((\Sigma^*)^2)$ .. regular

(i.e. $L((0+1)^* \cdot (0+1)^*)$

$\Rightarrow L' = L \cap L_1 \cap L_2$ regular due to closure under intersection

Exercise: If L_1, L_2 are regular, is $L_1 - L_2$ regular?

Lecture 12

CONTEXT-FREE GRAMMARS (CFG)

Definition: A CFG G is a quadruple

$G = (V, \Sigma, R, S)$ where

- V is a finite set of nonterminals / variables
- Σ is a finite set of terminals / constants
- $R \subseteq V \times (\Sigma \cup V)^*$ is a set of rules of the form $A \to \alpha$ ($A \in V$, $\alpha \in (\Sigma \cup V)^*$)
- S is the initial symbol. ($S \in V$)

Example: G : $S \to 0S1$, $S \to \#$ $\quad V = \{S\}$, $\Sigma = \{0, 1, \#\}$

$L(G)$

e.g.
- $\varepsilon \notin L(G)$
- $1 \notin L(G)$
- $\# \in L(G)$
- $0\#1 \in L(G)$

  $S \Rightarrow 0S1 \Rightarrow 0\#1$
- $000\#11 \notin L(G)$
- $00\#11 \in L(G)$

  $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 00\#11$

In general $L(G) = \{0^n \#1^n \mid n \geq 0\}$

Exercise: Prove that $L(G)$ is not regular.

Exercise: CFG for $L = \{0^n 1^n \mid n \geq 0\}$

CFG for $L = \{0^n 1^n \mid n \geq 1\}$

Decision Problems for Regular Languages

Emptiness Problem

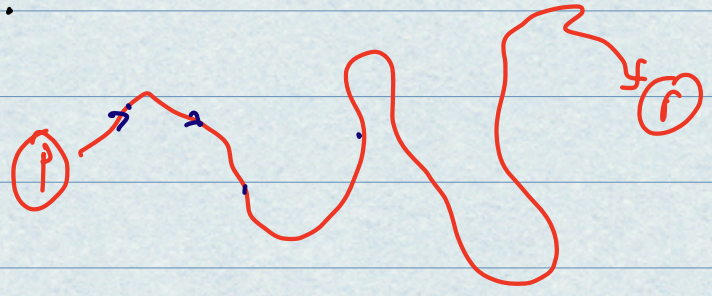
Input: A DFA M

Decide: Is $L(M) = \emptyset$?

Solution:

We first define a notion of **reachability** (accessibility).

Definition: Given $M = (Q, \Sigma, \delta, q_0, F)$ and $p, r \in Q$. We say "**r is reachable from p**" if there is a path from p to r in M.



For a state $p \in Q$, define

- $Acc(p) = \{ r \mid r$ is reachable from p via a path of length $\geq 0 \}$

- $Acc_+(p) = \{r \mid r$ is reachable from p via a path of length $\geq 1\}$

Note: • $p \in Acc(p)$
• $Acc_+(p) \subseteq Acc(p)$
• $Acc(p) = Acc_+(p) \cup \{p\}$

Q: How to compute $Acc(p)$ and $Acc_+(p)$ effectively for a state p?

Algorithm: to compute $Acc_+(p)$
Idea: Compute inductively.

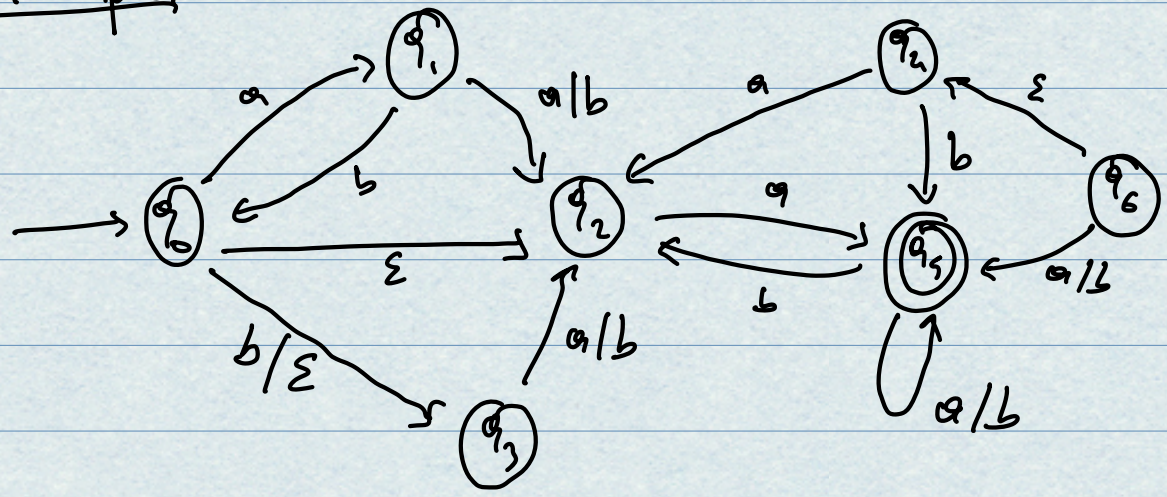
- Collect states that are reachable from p in one step.

- Recursively collect states that are reachable from the newly added states

- Repeat the above until no more states are added.

Example:

M:



- Compute $Acc(q_0)$ and $Acc_t(q_0)$

$A_1 = \{q_1, q_2, q_3\}$

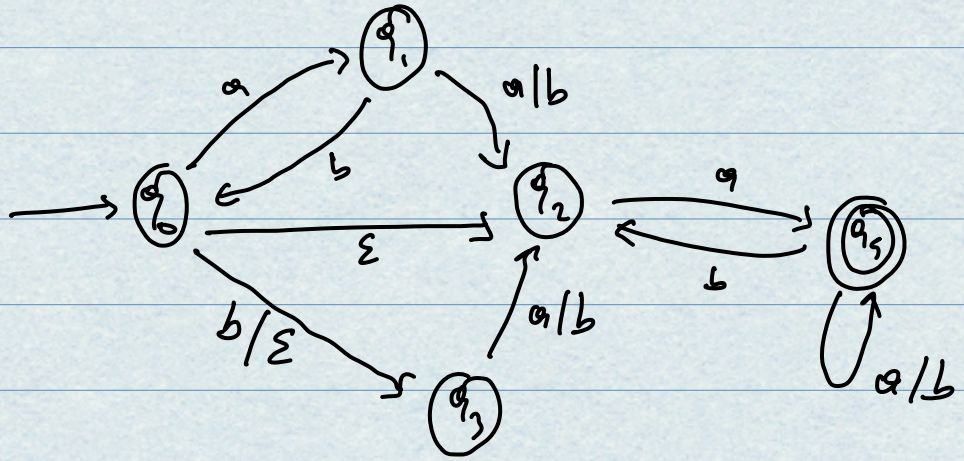
$A_2 = \{q_1, q_2, q_3, q_0, q_5\}$

$A_3 = \{q_1, q_2, q_3, q_0, q_5\} = Acc_t(q_0)$

$\Rightarrow$ q_4 and q_6 are not reachable from q_0.

$\Rightarrow Acc(q_0) = Acc_t(q_0) \cup \{q_0\}$

Now q_4 and q_6 can be removed from M.



Proposition: Every FA can be converted to an equivalent FA in which all states are reachable from q_0, the initial state.

* Emptiness problem is solvable.

$L(m) \neq \emptyset \iff Acc_+(q_0) \cap F \neq \emptyset$.

Finiteness Problems (FP)

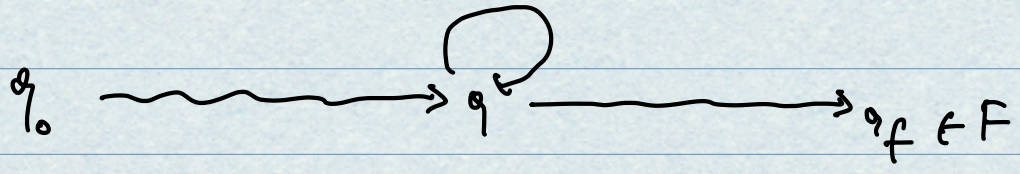
Input: A DFA m

Decide: Is $L(m)$ finite?

Proposition: FP is solvable

- Suppose $L(m)$ is infinite.
  $\Rightarrow$ there exist strings in $L(m)$ that are arbitrarily long.

- Let n = # of states of m
  By pumping lemma, a string $z \in L(m)$ with $|z| \geq n$ can be accepted by a computation



Note:

$L(M)$ is infinite $\Rightarrow$
- $q \in Acc(q_0)$
- $q \in Acc_+(q)$
- $Acc(q) \cap F \neq \emptyset$.

Also the converse of the above holds.

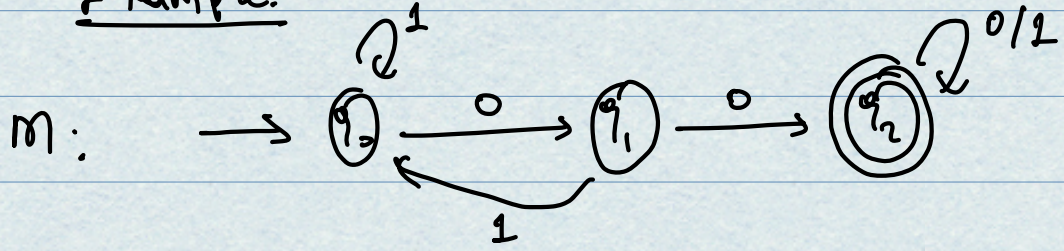
Therefore,

$L(M)$ is infinite $\Leftarrow$
- $q \in Acc(q_0)$ (1)
- $q \in Acc_+(q)$ (2)
- $Acc(q) \cap F \neq \emptyset$. (3)

To check if $L(M)$ is infinite
- Compute $Acc(q_0)$
- Find a state $q \in Q$
- Compute $Acc(q)$ and $Acc_+(q)$
- Check if (1), (2) and (3) hold.

If some $q \in Q$ can be found, then $L(M)$ is infinite, otherwise not!

Example:



- Let us pick q_2.

(1) $q_2 \in Acc_+(q_0)$

(2) $q_2 \in Acc(q_0)$

(3) $Acc_+(q_2) \cap F \neq \emptyset$

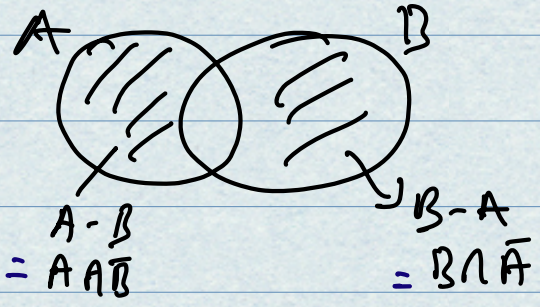
$\Rightarrow L(M)$ is infinite.

Equivalence Problem:

Input: Two DFAs M_1 and M_2

Decide: $L(M_1) = L(M_2)$?

Recall: $A = B \iff A - B = \emptyset \wedge B - A = \emptyset$



$\Rightarrow A = B \iff A \cap \bar{B} = \emptyset \wedge B \cap \bar{A} = \emptyset.$

Hence $L(M_1) = L(M_2)$ iff

$$\underbrace{L(M_1) \cap \overline{L(M_2)} = \emptyset}_{(*)} \wedge \underbrace{L(M_2) \cap \overline{L(M_1)} = \emptyset}_{(**)}$$

We can construct DFAs for (*) and (**) and check if their languages are empty, which we know how to solve.

Example:

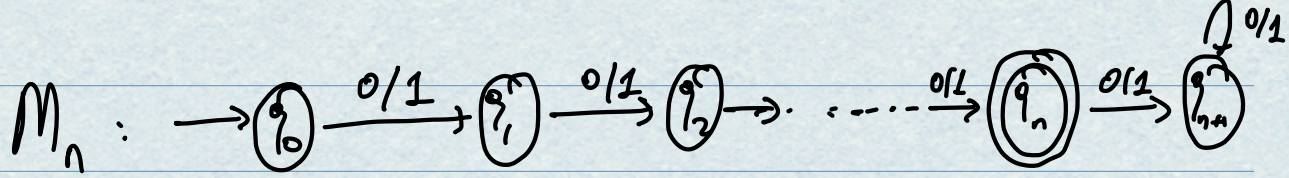
Input: A DFA M and $n \geq 0$.

Question: Does M accept a string z with $|z| = n$.

Show that this problem is solvable.

Proof:

- Given: M, $n \geq 0$
- Construct a DFA M_n that accepts strings in Σ^n (set of strings of length n)



Note: - $L(M)$ is regular

- $L(M_n)$ is regular

- Define a DFA $M \times M_n$ to accept $L(M) \cap L(M_n)$.

$\Rightarrow$ M accepts a string of length n iff $L(M) \cap L(M_n) \neq \emptyset$.

(solvable)

Lecture 13

Example: A CFG that describes the syntax for regular expressions over $\Sigma = \{a, b\}$.

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

$V = \{re\}$

$re \longrightarrow \emptyset$

$re \longrightarrow \varepsilon$

$re \longrightarrow a$

$re \longrightarrow b$

$re \longrightarrow re \cdot re$

$re \longrightarrow re + re$

$re \longrightarrow re^*$

e.g. $ab^* + \varepsilon$

$re \Rightarrow re + re$

$\Rightarrow re \cdot re + re$

$\Rightarrow a \cdot re + re$

$\Rightarrow a \cdot re^* + re$

$\Rightarrow a \cdot b^* + re$

$\Rightarrow a \cdot b^* + \varepsilon$

Definition: Let $G = (V, \Sigma, R, S)$ be a CFG

For a rule $A \to \alpha$, and strings $\beta, \gamma \in (\Sigma \cup V)^*$, we write $\beta A \gamma \Rightarrow_G \beta \alpha \gamma$

i.e. $\beta A \gamma$ directly rewrites to $\beta \alpha \gamma$.

- For $\theta_0, \theta_1, \ldots, \theta_n$, $n \geq 0$, $\theta_i \in (\Sigma \cup V)^*$ $i = 0, \ldots, n$, such that

$$\theta_0 \Rightarrow_G \theta_1 \Rightarrow_G \theta_2 \Rightarrow_G \cdots \Rightarrow_G \theta_n$$

we write $\theta_0 \Rightarrow_G^* \theta_n$

Remark:

$A \to \alpha$ context-free

e.g. $aAb \to \alpha$ context-sensitive

e.g. $aBAc \to cA$

Definition: Given a CFG $G = (V, \Sigma, R, S)$

- If $S \Rightarrow^* \alpha$, then α is called a sentential form.
- If $S \Rightarrow^* \alpha$ and $\alpha \in \Sigma^*$, then α is called a sentence.

- The language generated by G is

$$L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$$

- Two CFGs G_1 and G_2 are equivalent if $L(G_1) = L(G_2)$.
- A language $L \subseteq \Sigma^*$ is "context-free" if $L = L(G)$ for some CFG G.

Note: If we have rules $A \to \alpha_1, A \to \alpha_2, \ldots, A \to \alpha_n$, then we write

$$A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$$

Example: G: $S \to aSa \mid aBa$, $B \to bB \mid b$ $\quad L(G) = ?$

$S \Rightarrow^* aba$

$S \not\Rightarrow^* a$

$S \not\Rightarrow^* \varepsilon$

$S \Rightarrow^* ab$

$S \Rightarrow aSa$

$\Rightarrow a^k S a^k$

$\Rightarrow a^{k+1} B a^{k+1}$

$\Rightarrow a^{k+1} b^m B a^{k+1}$

$\Rightarrow a^{k+1} b^{m+1} a^{k+1}$

Note: $k+1 > 0$, $m+1 > 0$

$$L(G) = \{ a^{k+1} b^{m+1} a^{k+1} \mid k+1 \geq 0,\ m+1 > 0 \}$$

Lecture 14

Example:

G_1: $S \to AB$

$A \to aA \mid a$

$B \to bB \mid \varepsilon$

G_2: $S \to aS \mid aB$

$B \to bB \mid \varepsilon$

$$S \Rightarrow AB \overset{k}{\Rightarrow} a^kAB \Rightarrow a^{k+1}B \overset{m}{\Rightarrow} a^{k+1}b^mB \Rightarrow a^{k+1}b^m$$

$k+1 > 0$

$\Rightarrow k \geq 0$

$$L(G_1) = \{ a^k b^m \mid k > 0, \ m \geq 0 \}$$

$$= L(a^+ b^*)$$

$$= L(G_2)$$

$\Rightarrow$ G_1 and G_2 are equivalent.

Derivation Trees

Definition: Given a CFG $G = (V, \Sigma, R, S)$, a labeled ordered tree T is called a derivation tree in G if

- the root of T is labeled by some nonterminal of G.
- a node with label A (where $A \in V$) has children $Y_1, Y_2, \ldots, Y_k$, then there must be a rule in R s.t. $A \longrightarrow Y_1 \cdot Y_2 \cdot \ldots Y_k$.

Example: G: $S \to aB \mid bA$

$A \to aS \mid bAA \mid a$

$B \to bS \mid aBB \mid b$

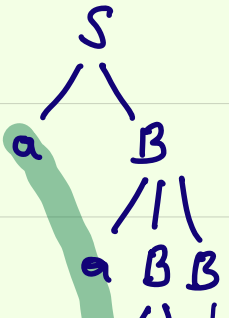
$w = aabaabb$

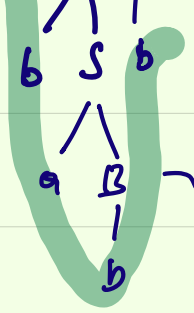
$S \Rightarrow a\underline{B} \Rightarrow aa\underline{BB} \Rightarrow aab\underline{S}B$

$\Rightarrow aabaB\underline{B}$

$\Rightarrow aababB$

$\Rightarrow aababb$





aabaabb

→ frontier of the derivation tree.

$S \Rightarrow^* w$ iff $\exists$ deriv. tree with frontier w.

* left-most derivation: Replace the nonterminal on the left-most position (at any step).

Designing CFGs

Although there exists no general algorithm to design a CFG for a given language, we use two practical methods:

1) Recursion technique
2) Decomposition technique

Recursion Technique:

Idea: Define the language recursively and extract rules from it.

Example: Set of palindromes 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.

CFG for set of palindromes over $\Sigma = \{0,1\}$.

- Let S be a nonterminal that stands for a palindrome.

$S \to \lambda \mid 0 \mid 1 \mid 0S0 \mid 1S1$

Exercise: - CFG for even length palindromes?

- " " odd " " ?

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. $x = ()()$ $\quad y = (())$

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

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

$$S \rightarrow \lambda \mid (S) \mid S \cdot S$$

Example: CFG for $L = \{a^i b^j \mid i < j < 2i\}$

e.g. $ab^2 \notin L$

- $i = 0$ no j.
- $i = 1$ no j
- $i = 2 \Rightarrow 2 < j < 4 \Rightarrow j = 3$

$\Rightarrow a^2 b^3 \in L$

Basis: $a^2 b^3 \in L$

- $i = 3 \Rightarrow 3 < j < 6$

$j = 4, 5$

$\Rightarrow a^3 b^4$, $a^3 b^5$

$a^3b^4 = a \cdot a^2b^3 \cdot b$

$a^3b^5 = a \cdot a^2b^3 \cdot b^2$

- i

Rec. Step: If $w \in L$, so are awb and awb^2.

No other rule is required.

CFG G: $S \to a^2b^3 \mid aSb \mid aSb^2$

Exercise: CFG for $L = \{0^n 1^n \mid n \geq 0\}$?

Basis: ?

Rec. Step: ?

Decomposition Technique

Idea: Decompose the language into simpler ones.

Example: CFG for $L = \{a^i b^j \mid i \neq j\}$

e.g. $ab \notin L$ $ab^2 \in L$

$a^2b \in L$

$L = L_1 \cup L_2$ where

$L_1 = \{ a^i b^j \mid i < j \}$

$L_2 = \{ a^i b^j \mid i > j \}$

CFG for L_1 : $B \to aBb \mid Bb \mid b$

CFG for L_2 : $A \to aAb \mid aA \mid a$

CFG for $L_1 \cup L_2$: $S \to A \mid B$

$B \to aBb \mid Bb \mid b$

$A \to aAb \mid aA \mid a$

Correctness of CFGs *

L ... CFG G

Verify if $L = L(G)$

not in the scope of exams

The CORRECTNESS PROBLEM

Q: How can we justify the CFG we designed?

For instance, let us consider the following CFG

$$S \to aB \mid bA$$
$$A \to aS \mid bAA \mid a$$
$$B \to bS \mid aBB \mid b$$

and prove that

claim → $L(G) = \{ w \in \{a,b\}^+ \mid \#_a(w) = \#_b(w) \}$.

$$L_a = \{ w \in \{a,b\}^+ \mid \#_a(w) = \#_b(w) + 1 \}$$

$$L_b = \{ w \in \{a,b\}^+ \mid \#_b(w) = \#_a(w) + 1 \}$$

$$L_= = L(G)$$

To prove the claim above, we need to prove the following:

(1) $A \Rightarrow^* w$ iff $w \in L_a$

(2) $B \Rightarrow^* w$ iff $w \in L_b$

(3) $S \Rightarrow^* w$ iff $w \in L_=$

Proof: "$\Rightarrow$" (the only-if direction)

by induction on the length of derivations:

Basis:

1) $A \Rightarrow^* w$ length 1.

Then, we have $A \Rightarrow a \in L_a$

2) $B \Rightarrow^* w$ with length 1.

$\Rightarrow B \Rightarrow b \in L_b$

3) $S \Rightarrow^* w$ with length 2. (Why?)

Thus, it must be either

- $S \Rightarrow aB \Rightarrow ab \in L_=$

or

- $S \Rightarrow bA \Rightarrow ba \in L_=$.

Inductive Step:

Suppose for some odd integer $m \geq 1$ that

Note: This is inductive hyp.

(1) $A \overset{n}{\Rightarrow} w \implies w \in L_a$ (length of derivation)

(2) $B \overset{n}{\Rightarrow} w \implies w \in L_b$

(3) $S \overset{n+1}{\Rightarrow} w \implies w \in L_=$.

for all odd n s.t. $1 \leq n \leq m$.

For part (1): Consider a longer derivation $A \overset{n+2}{\Rightarrow} w$. Note that there are two possible first steps:

Case 1.1: $A \Rightarrow aS \overset{n+1}{\Rightarrow} a \cdot y = w$

Since $S \overset{n+1}{\Rightarrow} y$ is of length $m+1$, by ind. hyp. $y \in L_=$.

$w = a \cdot y \in L_a$.

Case 1.2:

$A \Rightarrow bAA \overset{n_1}{\Rightarrow} bxA \overset{n_2}{\Rightarrow} bxy = w$

Since $A \overset{n_1}{\Rightarrow} x$, $A \overset{n_2}{\Rightarrow} y$ and $n_1, n_2 \leq m$

by ind. hyp. $x, y \in L_a$

Recall the CFG:

$S \to aB \mid bA$

$A \to aS \mid bAA \mid a$

$B \to bS \mid aBB \mid b$

Note that $w = b\,x\,y \in L_a$

because $\#_a(w) = \#_b(w) + 1$.

For part (2): Consider a longer derivation $B \overset{m+2}{\Longrightarrow} w$.

As in case (1), by using similar arguments, we obtain $w \in L_b$.

e.g. $B \Longrightarrow bS \overset{m+1}{\Longrightarrow} by \ldots y \in L_=$, $by \in L_b$.

For part (3): Consider a longer derivation $S \overset{m+3}{\Longrightarrow} w$.

For the first step there are two possibilities

Case 3.1: $S \Longrightarrow aB \overset{m+2}{\Longrightarrow} ay = w$.

As proved in (2) $B \overset{m+2}{\Longrightarrow} y$ implies $y \in L_b$.

Then, $w = ay \in L_=$.

Case 3.2: $S \Longrightarrow bA \overset{m+2}{\Longrightarrow} bx = w$

By similar argument as in case 3.1, we have $x \in L_a$, $w = bx \in L_=$.

This concludes the inductive step, and hence "$\Longrightarrow$"-direction is proven.

We now prove "$\Longleftarrow$" - direction

We show that

(1) $w \in L_a$ implies $A \overset{+}{\Longrightarrow} w$

(2) $w \in L_b$ implies $B \overset{+}{\Longrightarrow} w$

(3) $w \in L_=$ implies $S \overset{+}{\Longrightarrow} w$.

Proof: By induction on $|w|$.

Basis:

(1) $w = a \Longrightarrow A \Longrightarrow a$

(2) $w = b \Longrightarrow B \Longrightarrow b$

(3) $w = ab$ or $w = ba$

Clearly, $S \Longrightarrow aB \Longrightarrow ab$

and $S \Longrightarrow bA \Longrightarrow ba$.

Inductive Step:

Inductive Hypothesis:

Suppose for some odd $m \geq 1$ that

(1) $w \in L_a \wedge |w| = n$ implies $A \overset{*}{\Longrightarrow} w$

(2) $w \in L_b \wedge |w| = n$ implies $B \overset{*}{\Longrightarrow} w$

(3) $w \in L_= \wedge |w| = n+1$ implies $S \overset{*}{\Longrightarrow} w$

for all odd n s.t. $1 \leq n \leq m$.

For part (1): Let $w \in L_a$ and $|w| = m+2$.

Note that w begins with a or b.

Case (1.1): $w = bx$.

Then $|x| = m+1$. and

$$\#_a(x) = \#_b(x) + 2.$$

Q: How to generate bx from A.

It has to be of the form

$$A \Longrightarrow bAA \overset{*}{\Longrightarrow} bx.$$

i.e. $AA \overset{*}{\Longrightarrow} yA \overset{*}{\Longrightarrow} yz = x.$

Recall:

$S \to aB \mid bA$
$A \to aS \mid bAA \mid a$
$B \to bS \mid aBB \mid b$

If this is to be achieved, then one should be able to write $x = yz$ and $y, z \in L_a$.

Indeed one must be able to prove the following fact:

Fact: If $\#_a(x) = \#_b(x) + 2$, then x can be written as $x = yz$. s.t. $y, z \in L_a$.

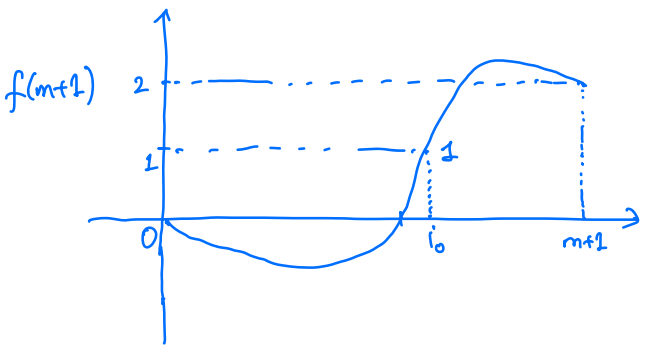
Proof of the fact: Let $x = x_1 \ldots x_{m+1}$.

(Recall $|x| = m+1$).

Define a function $f: \{0, 1, \ldots, m+1\} \to \mathbb{Z}$.

by $f(i) = \#_a(x_1 \ldots x_i) - \#_b(x_1 \ldots x_i)$.

Then $f(0) = 0$ by definition and
$f(m+1) = 2$ by assumption.
The graph of f has the form



So there exists $0 < i_0 < m+1$ s.t.
$$f(i_0) = 1.$$
Letting $y = x_1 \dots x_{i_0}$ and $z = x_{i_0+1} \dots x_{m+1}$,
we get $y, z \in L_a \wedge x = yz$. □

Now, both $y, z \in L_a$ and $|y|, |z| \leq m$,
so by inductive hypothesis
$$A \Rightarrow^* y, \quad A \Rightarrow^* z.$$
Therefore, we have $A \Rightarrow bAA \Rightarrow^* byA \Rightarrow^* byz = w$.

Case (1.2): $w = ax$.

Then, $x \in L_=$ and $|x| = m+1$.
By inductive hypothesis, $S \Rightarrow^* x$.
Therefore, $A \Rightarrow aS \Rightarrow^* ax = w$.

For part (2): Similar to (1).

Exercise: Try working out the details for part (2) on your own.

For part (3): Let $w \in L_=$ and $|w| = m+3$.

Case (3.1) $w = ax$.

Then $x \in L_b$, $|x| = m+2$.

From (2) we get that $B \Rightarrow^* x$.

Therefore, $S \Rightarrow aB \Rightarrow^* ax = w$.

Case (3.2): $w = bx$.

Then $x \in L_a$, $|x| = m+2$.

From (1), it follows that $A \Rightarrow^* x$.

Then, $S \Rightarrow bA \Rightarrow^* bx = w$.

This completes the proof of "$\Leftarrow$"-direction.

Since we proved both $\Rightarrow$ and $\Leftarrow$ directions, we are done with proof of the claim. $\square$

REGULAR GRAMMARS

A subclass of CFGs that generate regular languages.

Definition: A reg. grammar is a CFG in which every rule is in one of the following forms:

- $A \to aB$
- $A \to a$
- $A \to \varepsilon$

for all $A, B \in V$ and $a \in \Sigma$.

Example:

$S \to aB \mid \varepsilon$

$B \to bS \mid bA$

$A \to aA \mid \varepsilon$



$L(G) = (ab)^+ a^* + \varepsilon$ (Why?)

Example:

$S \to aB \mid bA$

$A \to aS \mid bAA \mid a$

$B \to bS \mid aBB \mid b$

Not regular! $A \to bAA$ is not in correct form.

Regular Grammars & Finite Automata

Goal: To show reg. grammars $\Leftrightarrow$ FA.

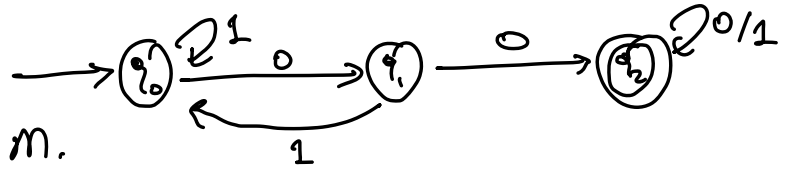
Proposition: $L = L(M)$ for some DFA M
$\Rightarrow$ $L = L(G)$ for some reg. gr. G.

Proof: Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, the reg. grammar $G = (V, \Sigma, R, S)$ is constructed as follows

- $V = Q$ $\qquad \Sigma_M = \Sigma_G$
- $S = q_0$
- R contains the following rules
  - $\delta(q, a) = q' \iff q \longrightarrow aq'$
  - For each $q_f \in F$, add $q_f \to \varepsilon$.

- Transitions $\longrightsquigarrow$ Productions
- Computation $\longrightsquigarrow$ Derivations

Example:



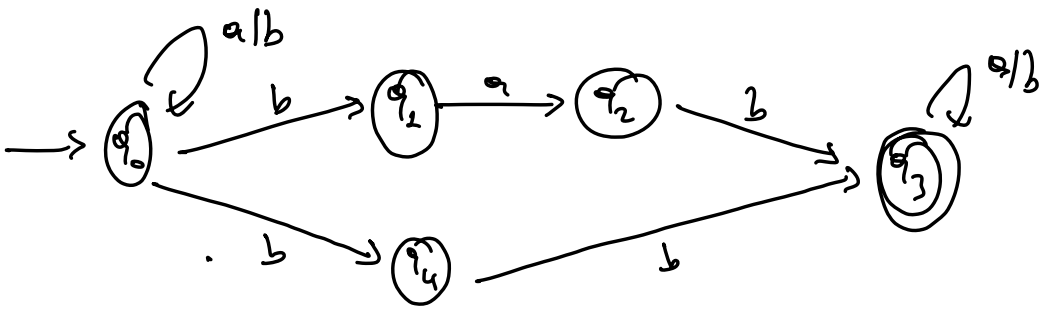
DFA M.

CFG G:

$q_0 \to 1q_0$
$q_0 \to 0q_1$
$q_1 \to 1q_0$
$q_1 \to 0q_2$
$q_2 \to 1q_2$
$q_2 \to 0q_2$
$q_2 \to \varepsilon$.

$q_0 \to 1q_0 \mid 0q_1$
$q_1 \to 1q_0 \mid 0q_2$
$q_2 \to 1q_2 \mid 0q_2 \mid \varepsilon$

Example: Same technique applies to NFAs as well.



$q_0 \to aq_0 \mid bq_0 \mid bq_1 \mid bq_4$
$q_1 \to aq_2$
$q_2 \to bq_3$
$q_4 \to bq_3$
$q_3 \to aq_3 \mid bq_3 \mid \varepsilon$

Proposition: $L = L(G)$ for some reg. grammar G.
$\Rightarrow$ $L = L(M)$ for some NFA M.

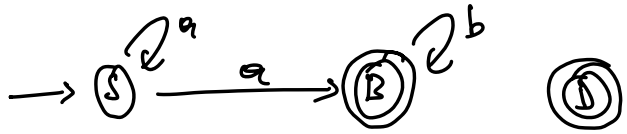
Proof: For a given reg. grammar $G = (V, \Sigma, R, S)$, the equivalent NFA $N = (Q, \Sigma, \delta, q_0, F)$ is constructed as follows:

1) $Q = V \cup \{D\}$
2) $q_0 = S$
3) $F = \{A \in V \mid A \to \varepsilon \in R\} \cup \{D\}$
4) δ is defined by
   - $\delta(A, a) = B$ iff $A \to aB \in R$.
   - $\delta(A, a) = D$ iff $A \to a \in R$

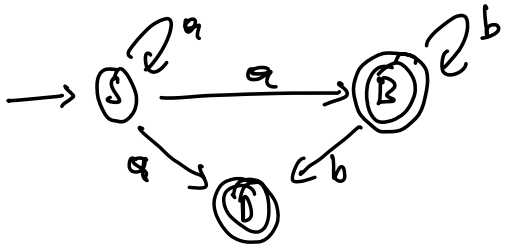
□

Example: G: $S \to aS \mid aB$
$B \to bB \mid \varepsilon$

The equivalent NFA N is



* Suppose G also contains $S \to a$ and $B \to b$

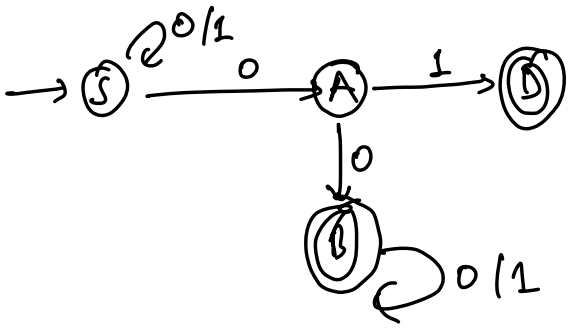


Example:

$S \rightarrow 0S \mid 1S \mid 0A$

$A \rightarrow 0B \mid 1$

$B \rightarrow 0B \mid 1B \mid \varepsilon$



Lecture 15

Regular Grammars

Definition: A regular grammar is a CFG in which each rule is in one of the following forms:

- $A \longrightarrow aB$
- $A \longrightarrow a$
- $A \longrightarrow \varepsilon$

$(A, B \in V, a \in \Sigma)$

Example:

1) $B \longrightarrow aBb \mid Bb \mid b$ ... not regular

2) $S \longrightarrow aB \mid \varepsilon$
$B \longrightarrow bS \mid bA$
$A \longrightarrow aA \mid \varepsilon$
... regular

3) G: $S \longrightarrow aSa \mid aBa$
$B \longrightarrow bB \mid b$
... not regular

Equivalence between Reg. Grammars and FA

Lemma: $L = L(M)$ for some DFA M
$\Rightarrow L = L(G)$ for some reg. gr. G.

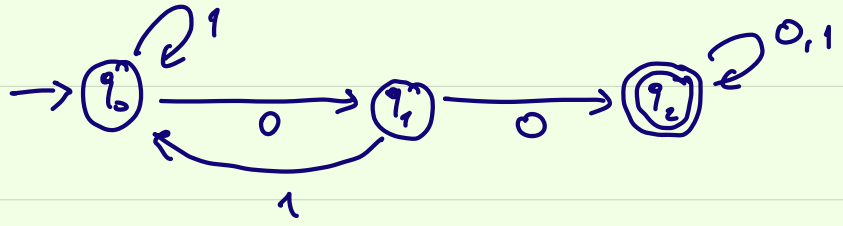
Proof: Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, we construct a reg. gr. $G = (V, \Sigma, R, S)$ as follows:

- $V = Q$
- Input alphabet Σ is now the set of terminals.
- $S = q_0$
- R contains the following rules
  - For $\delta(q, a) = q'$, add a rule $q \to aq'$
  - For each $q_f \in F$, add a rule $q_f \to \varepsilon$



Rule as transitions
Derivation as computations

Example:

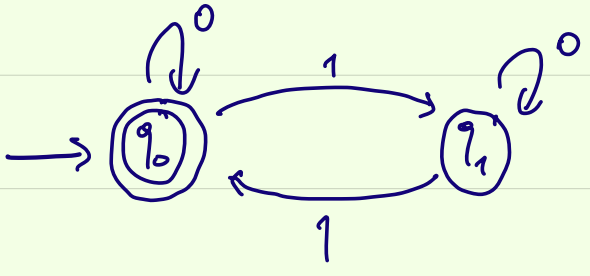


$q_0 \to 1q_0$
$q_0 \to 0q_1$
$q_1 \to 1q_0$
$q_1 \to 0q_2$
$q_2 \to 0q_2$
$q_2 \to 1q_2$
$q_2 \to \varepsilon$

or

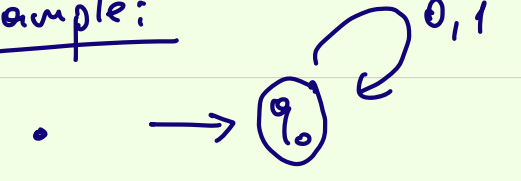
$q_0 \to 1q_0 \mid 0q_1$
$q_1 \to 1q_0 \mid 0q_2$
$q_2 \to 0q_2 \mid 1q_2 \mid \varepsilon$

Example:

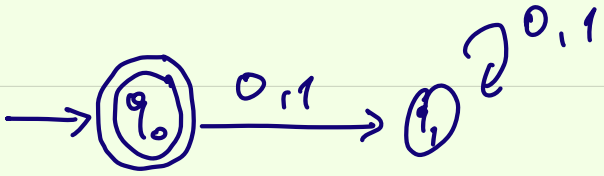


$q_0 \to 0q_0 \mid 1q_1 \mid \varepsilon$
$q_1 \to 0q_1 \mid 1q_0$

Example:



$q_0 \to 0q_0 \mid 1q_0$



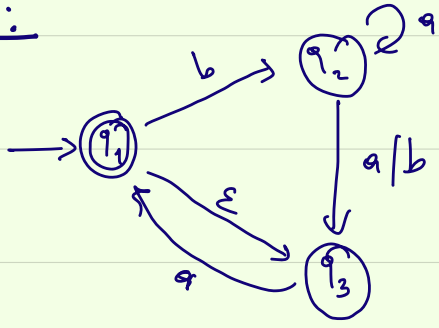
$q_0 \to 0q_1 \mid 1q_1 \mid \varepsilon$

$q_1 \to 0q_1 \mid 1q_1$

Exercise: Reg. gr for 

Remark: The construction applies to NFAs as well.

Example:

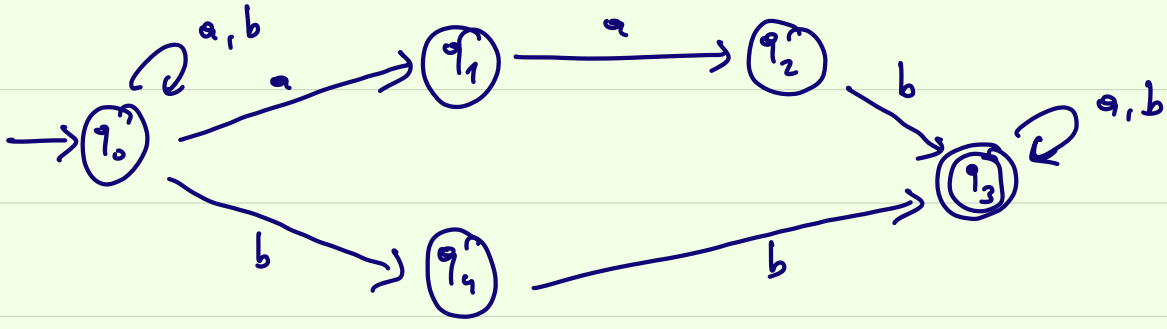


$q_1 \to bq_2 \mid \varepsilon q_3 \mid \varepsilon$

$q_2 \to aq_2 \mid aq_3 \mid bq_3$

$q_3 \to aq_1$

Example:



$q_0 \to aq_0 \mid aq_1 \mid bq_0 \mid bq_4$

$q_1 \to aq_2$

$q_4 \to bq_3$

$q_2 \to bq_3$

$q_3 \to aq_3 \mid bq_3 \mid \varepsilon$

Lemma: $L = L(G)$ for some reg. gr G
$\Rightarrow$ $L = L(M)$ for some DFA M.

Proof: Given a reg. gr $G = (V, \Sigma, R, S)$, we construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$ as follows:

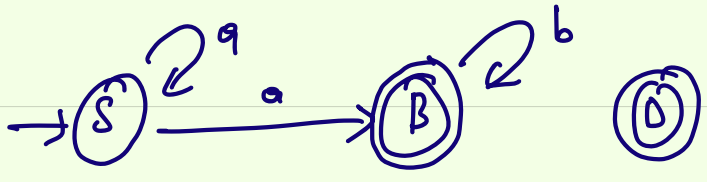
- $Q = V \cup \{D\}$
- $q_0 = S$
- Σ is the set of terminals of G.
- $F = \{A \in V \mid A \to \varepsilon \in R\} \cup \{D\}$
- δ is defined by

$$\delta(A, a) = B \quad \text{iff} \quad A \to aB \in R$$

$$\delta(A, a) = D \quad \text{iff} \quad A \to a \in R$$

Example: $S \to aS \mid aB$

$B \to bB \mid \varepsilon$

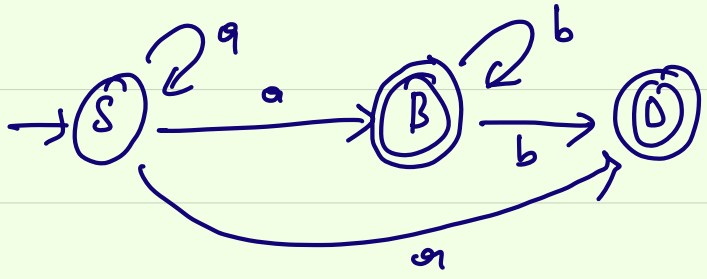


(Now, convert this NFA into an equiv. DFA. left as an exercise.)

Example:

$S \to aS \mid aB \mid a$

$B \to bB \mid \varepsilon \mid b$

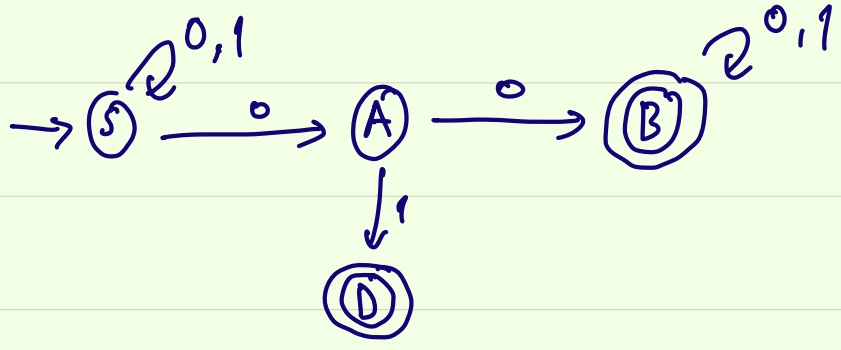


Example:

$S \to 0S \mid 1S \mid 0A$

$A \to 0B \mid 1$

$B \to 0B \mid 1B \mid \varepsilon$



Note: This is an NFA. Convert it to a DFA.

Lecture 16

NORMAL FORMS OF CFGs

Outline:

- Elimination of useless symbols
- " " ε-productions
- " " chain productions
- Converting CFGs to their Chomsky Normal Forms.

Elimination of useless symbols

Useless symbols are the ones that are not used in any terminal derivation.

Let $G = (V, \Sigma, R, S)$ be a CFG.

A grammar symbol $X \in \Sigma \cup V$ is useful if there is a derivation $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ where $\alpha, \beta \in (\Sigma \cup V)^*$ and $w \in \Sigma^*$.

Equivalently, X is useful iff it appears in a terminal derivation tree.



v is a substring of w.

Remark: If X is not useful, then it is useless.

In a derivation like $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$, consider the following two subderivations:

- $S \Rightarrow^* \alpha X \beta$ $\quad \alpha, \beta \in (\Sigma \cup V)^*$
- $X \Rightarrow^* v$ $\quad v \in \Sigma^*$

Definition: A nonterminal $X \in V$ is productive if there is a derivation $X \Rightarrow^* v$, $v \in \Sigma^*$.

e.g. $S \to AB$
$B \to bB \mid A$
$A \to a$

$A \Rightarrow a$.

A is productive

$S \Rightarrow AB \Rightarrow aB \Rightarrow aA \Rightarrow aa$.

Definition: A grammar symbol $A \in \Sigma \cup V$ is reachable if there is a derivation $S \Rightarrow^* \alpha A \beta$, $\alpha, \beta \in (\Sigma \cup V)^*$.

Proposition: Let $G = (V, \Sigma, R, S)$ be a CFG. Then all symbols in $(V \cup \Sigma)$ is useful iff

i) Every nonterminal is productive

ii) Every symbol is reachable.

Elimination nonproductive nonterminals:

Idea: First find all the productive nonterminals.

Algorithm (Compute productive nonterminals)

Input: A CFG $G = (V, \Sigma, R, S)$

Output: $P \subseteq V$, set of prod. nonterminals.

$$P_1 := \{ A \in V \mid \exists A \to v \in R : v \in \Sigma^* \}$$

$i = 1$

repeat

$i = i + 1$

$$P_i = P_{i-1} \cup \{ B \in V \mid \exists\, B \to \alpha \in R \quad \alpha \in (P_{i-1} \cup \Sigma)^* \}$$

until $P_i = P_{i-1}$.

return P_i.

e.g. $A \to ab$

$B \to aAb$

$C \to aABA$

Example: G: $S \to AC \mid BS \mid B$

$A \to aA \mid aF$

$B \to CF \mid b$

$C \to aC \mid Da$

$D \to aD \mid BD \mid C$

$E \to aA \mid BSA$

$F \to bB \mid b$

$P_1 = \{B, F\}$

$P_2 = \{B, F, S, A\}$

$P_3 = \{B, F, S, A, E\}$

$P_4 = \{B, F, S, A, E\} = P_3$

$\Rightarrow$ C, D are not productive.

Q: How to eliminate C and D?

Remove all the rules that involve them.

G': $S \to BS \mid B$

$A \to aA \mid aF$

$B \to b$

$E \to aA \mid BSA$

$F \to bB \mid b$

Note:

$L(G) = L(G')$

Elimination of unreachable symbols:

Idea: Find all the reachable symbols first.

Algorithm:

Input: A CFG $G = (V, \Sigma, R, S)$

Output: $U \subseteq V \cup \Sigma$, the set of reachable symbols

- $U_0 = \{S\}$
- $i = 0$
- repeat
  - $i = i+1$
  - $U_i = U_{i-1} \cup \{X \in V \cup \Sigma \mid \exists A \to \alpha X \beta \in R, A \in U_{i-1}\}$.
- until $U_i = U_{i-1}$
- Return $U = U_i$.

Example:

G: $S \to AC \mid BS \mid B$

$A \to aA \mid aF$

$B \to CF \mid b$

$C \to aC \mid D$

$D \to aD \mid BD \mid C$

$E \to aA \mid BSA$

$F \to bB \mid b$

G': $S \to BS \mid B$

$A \to aA \mid aF$

$B \to b$

$E \to aA \mid BSA$

$F \to bB \mid b$

$P_1 = \{B, F\}$

$P_2 = \{B, F, S, A\}$

$P_3 = \{B, F, S, A, E\}$

$P_4 = \{B, F, S, A, E\} = P_3$

$\Rightarrow$ C, D are not productive.

G': $S \to BS \mid B$

$A \to aA \mid aF$

$B \to b$

$E \to aA \mid BSA$

$F \to bB \mid b$

Let us find reachable symbols in G':

$U_0 = \{S\}$

$U_1 = \{S, B\}$

$U_2 = \{S, B, b\}$

$U_3 = \{S, B, b\}$

$\Rightarrow$ A, E, F, a are not reachable.

Now remove all unreachable symbols from G' by deleting the rules that involve them.

G'': $S \to BS \mid B$

$B \to b$

Note:

$L(G) = L(G') = L(G'') = L(bb^*)$

Example: G: $S \to ABC \mid Bab$

$A \to aA \mid BaC \mid aaa$

$B \to bBb \mid a$

$C \to CA \mid AC$

Prod. nonterminals:

$P_1 = \{A, B\}$

$P_2 = \{A, B, S\}$

$P_3 = P_2$

$\Rightarrow$ C is not productive.

Remove C:

G': $S \to Bab$

$A \to aA \mid aaa$

$B \to bBb \mid a$

Reachable Symbols in G':

$V_0 = \{S\}$

$V_1 = \{S, B, a, b\}$

$V_2 = V_1$

$\Rightarrow$ A is unreachable.

Remove A from G'

G'': $S \to Bab$

$B \to bBb \mid a$

Remark: From a given CFG G, we construct a CFG G' with only useful symbols s.t. $L(G) = L(G')$ by

1) Removing all nonproductive nonterminals
2) Removing all unreachable symbols.

Note that applying step 2 first may lead to incorrect results.

Example: G: $S \to AB \mid a$

$A \to a$

- All symbol are reachable: No change.
- B is not productive:

$\Rightarrow$ G': $S \to a$, $A \to a$ but still we have A, which is unreachable!

Exercise:

G: $S \to AB \mid AC$

$A \to aAb \mid bAa \mid a$

$B \to bbA \mid aaB \mid AB$

$C \to abCa \mid aDb$

$D \to bD \mid aC$

$E \to bE \mid \varepsilon$

Lecture 17

Elimination of ε-productions

$A \to \varepsilon$ ... an ε-production

As ε-productions produce nothing, we want to remove them from the CFG G while preserving the language $L(G)$.

G ..... $L(G)$

$\Downarrow$ $\Downarrow$

G' $L(G') = L(G) - \{\varepsilon\}$

without ε-prod.

Observe: Let $A \Rightarrow^* \varepsilon$ be a derivation.

Then we have:

- $A \Rightarrow \varepsilon$ or
- $A \Rightarrow A_1 A_2 \cdots A_k \Rightarrow^* \varepsilon$

where $A_1 \Rightarrow^* \varepsilon$, $A_2 \Rightarrow^* \varepsilon$, ..., $A_k \Rightarrow^* \varepsilon$.

Definition: A nonterminal $A \in V$ is **nullable** if there is a derivation $A \Rightarrow^* \varepsilon$.

Example: G:

$S \to ACA$

$A \to aAa \mid B \mid C$

$B \to bB \mid b$

$C \to cC \mid \varepsilon$

$N_0 = \{C\}$

$N_1 = \{C, A\}$

$N_2 = \{C, A, S\}$

$N_3 = N_2$

$\Rightarrow$ C, A, S are nullable.

- Let's take a production rule

$$A \to \alpha_0 A_1 \alpha_1 A_2 \alpha_2$$

where
- A_1, A_2 are nullable.
- $\alpha_0, \alpha_1, \alpha_2$ don't contain nullable symbols. (Note: $\alpha_0, \alpha_1, \alpha_2 \in (V \cup \Sigma)^*$)

Replace $A \to \alpha_0 A_1 \alpha_1 A_2 \alpha_2$ with the following rules:

$$A \to \alpha_0 \alpha_1 \alpha_2 \mid \alpha_0 \alpha_1 A_2 \alpha_2 \mid \alpha_0 A_1 \alpha_1 \alpha_2 \mid \alpha_0 A_1 \alpha_1 A_2 \alpha_2$$

After applying this transformation eagerly, remove all of the ε-productions.

Exercise: How to replace $A \to \alpha_0 A_1 \alpha_1 A_2 \alpha_2 A_3 \alpha_3$?

Example: G: $S \to ACA$
$A \to aAa \mid B \mid C$
$B \to bB \mid b$
$C \to cC \mid \varepsilon$

$N_0 = \{C\}$
$N_1 = \{C, A\}$
$N_2 = \{C, A, S\}$
$N_3 = N_2$

$\Rightarrow$ C, A, S are nullable.

G': $S \to ACA \mid CA \mid AA \mid AC \mid A \mid C$
$A \to aAa \mid aa \mid B \mid C$
$B \to bB \mid b$
$C \to cC \mid c$

Example: G: $S \to \underline{A}B\underline{C}B\underline{C}\underline{D}$
$A \to \underline{C}\underline{D}$
$B \to \underline{C}b$
$C \to \underline{A} \mid \varepsilon$
$D \to b\underline{D} \mid \varepsilon$

$N_0 = \{C, D\}$
$N_1 = \{C, D, A\}$
$N_2 = N_1$

$\Rightarrow$ C, D, A are nullable.

G': $S \to ABCBCD \mid BCBCD \mid ABBCD \mid ABCBD \mid ABCBC \mid$
$BBCD \mid BCBD \mid BCBC \mid ABBD \mid ABBC \mid ABCB$
$BBD \mid BBC \mid BCD \mid ABB \mid BB$
$A \to CD \mid C \mid D$

$B \to Cb \mid b$

$C \to A$

$D \to bD \mid b$

Note: $L(G') = L(G) - \{\varepsilon\}$

Elimination of Chain Productions

A production of the form $A \to B$, $A, B \in V$, is called a chain production.

e.g. $A \to B \Rightarrow C \Rightarrow D \Rightarrow E \Rightarrow F \Rightarrow \alpha$

s.t. $|\alpha| \geq 2$.

We want to replace this with $A \Rightarrow \alpha$.

Define, for each $A \in V$, the set

$CHAIN(A) = \{ B \in V \mid \exists$ a derivation $A \Rightarrow^* B$ using only chain rules. $\}$

Example:

$S \to A \mid cB$

$A \to C \mid D$

$B \to bB \mid b$

$C \to cC \mid c$

$D \to dD \mid d$

$CHAIN(S) = \{A, C, D\}$

$CHAIN(A) = \{C, D\}$

$CHAIN(B) = \emptyset$

$= CHAIN(C)$

$= CHAIN(D)$

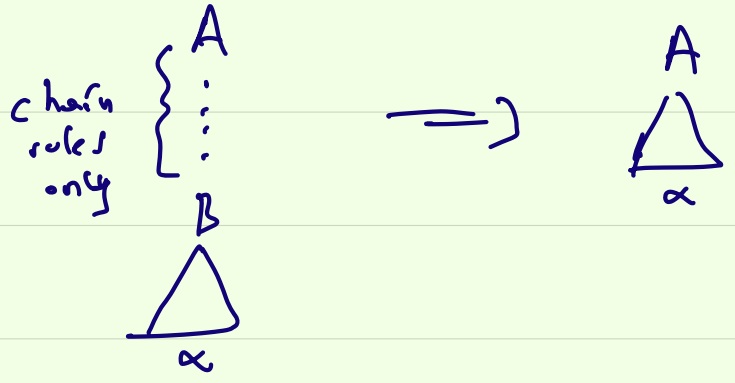
Q: How to remove chain rules?

A: Apply the following rule:

For every pair of nonterminals $A, B \in V$ s.t. $B \in CHAIN(A)$ and $B \to \alpha \in R$ (which is not a chain rule), add a rule $A \to \alpha$ to R.

Then remove all the chain rules.

i.e.



Example: G: $S \to A \mid cB$

$A \to C \mid D$

$B \to bB \mid b$

$C \to cC \mid c$

$D \to dD \mid d$

$CHAIN(S) = \{A, C, D\}$

$CHAIN(A) = \{C, D\}$

$CHAIN(B) = \emptyset$

$= CHAIN(C)$

$= CHAIN(D)$

G': $S \to cB \mid cC \mid c \mid dD \mid d$

$A \to cC \mid c \mid dD \mid d$

$B \to bB \mid b$

$C \to cC \mid c$

$D \to dD \mid d$

Example: G' : $S \to ACA \mid CA \mid AA \mid AC \mid A \mid C$

$A \to aAa \mid aa \mid B \mid C$

$B \to bB \mid b$

$C \to aC \mid c$

$CHAIN(S) = \{A, B, C\}$

$CHAIN(A) = \{B, C\}$

G'' : $S \to ACA \mid CA \mid AA \mid AC$

$aAa \mid aa \mid bB \mid b \mid cC \mid c$

$A \to aAa \mid aa \mid bB \mid b \mid cC \mid c$

$B \to bB \mid b$

$C \to aC \mid c$

Example:

G' : $S \to ABCBCD \mid BCBCD \mid ABBCD \mid ABCBD \mid ABCBC \mid$

$BBCD \mid BCBD \mid BCBC \mid ABBD \mid ABBC \mid ABCB$

$BBD \mid BBC \mid BCD \mid ABB \mid BB$

$A \to CD \mid C \mid D$

$B \to Cb \mid b$

$C \to A$

$D \to bD \mid b$

$CHAIN(S) = \emptyset$

$CHAIN(A) = \{C, D\}$

$CHAIN(C) = \{A, D\}$

$G'': S \rightarrow ABCBCD \mid BCBCD \mid ABBCD \mid ABCBD \mid ABCBC \mid$

$BBCD \mid BCBD \mid BCBC \mid ABBD \mid ABBC \mid ABCB$

$BBD \mid BBC \mid BCD \mid ABB \mid BB$

$A \rightarrow CD \mid bD \mid b$

$B \rightarrow Cb \mid b$

$C \rightarrow CD \mid bD \mid b$

$D \rightarrow bD \mid b$

Lecture 18

CHOMSKY NORMAL FORM (CNF)

A CFG G is in CNF if every rule in G has the form:

- $A \to BC$ or $(A, B, C \in V)$
- $A \to a$ $(a \in \Sigma, A \in V)$

(Assume: All symbols are useful and $\varepsilon \notin L(G)$)

Theorem: Given a CFG G, there is an algorithm to convert it into its CNF.

i.e. $G \rightsquigarrow G'$ s.t. $L(G) = L(G')$, G' in CNF

Proof: W.l.o.g. we assume that G has no ε-productions, no chain rules and no useless symbols. Then we apply two more steps:

Step 1: For each $a \in \Sigma$, define a new nonterminal D_a and a new rule $D_a \to a$. Then, replace each rule $A \to \alpha$, $|\alpha| \geq 2$,

with a new rule $A \to \bar{\alpha}$ s.t. $\bar{\alpha}$ is obtained from α by replacing each $a \in \Sigma$ in it with D_a.

Step 2: Let $A \to A_1 \cdot A_2 \cdot \dots A_m$, $m \geq 3$, be a rule. Replace it by $A \to A_1 C_1$, $C_1 \to A_2 C_2$, ... , $C_{m-2} \to A_{m-1} A_m$.

where $C_1, \dots, C_{m-2}$ are new nonterminals. ∎

e.g. $A \to BCD \rightsquigarrow A \to BE_1$, $E_1 \to CD$.

Example: G: $S \to AACD$
$A \to aA \mid \varepsilon$
$C \to aC \mid a$
$D \to aD \mid Db \mid \varepsilon$

① **Remove ε-productions:**

A, D are nullable.

G': $S \to AACD \mid ACD \mid AAC \mid CD \mid AC \mid C$
$A \to aA \mid a$
$C \to aC \mid a$
$D \to aD \mid a \mid Db \mid b$

② Remove chain rules:

$CHAIN(S) = \{C\}$

G'': $S \to AACD \mid ACD \mid AAC \mid CD \mid AC \mid aC \mid a$

$A \to aA \mid a$

$C \to aC \mid a$

$D \to aD \mid a \mid Db \mid b$

③ Remove useless symbols: No change.

- All nonterminals are productive.
- All symbols are reachable

④ Convert to CNF:

Step 1:

G''': $S \to AACD \mid ACD \mid AAC \mid CD \mid AC \mid D_a C \mid a$

$A \to D_a A \mid a$

$C \to D_a C \mid a$

$D \to D_a D \mid a \mid D D_b \mid b$

$D_a \to a$

$D_b \to b$

Step 2:

$G^{(iv)}$: $S \to CD \mid AC \mid D_a C \mid a$

$S \to AE_1 \quad E_1 \to AE_2 \quad E_2 \to CD$

$S \to AF_1 \quad F_1 \to CD$

$S \to A G_1 \quad G_1 \to AC$

$A \to D_a A \mid a$

$C \to D_a C \mid a$

$D \to D_a D \mid a \mid D D_b \mid b$

$D_a \to a$

$D_b \to b$

Example: G: $S \to ACA$

$A \to aAa \mid B \mid C$

$B \to bB \mid b$

$C \to cC \mid \varepsilon$

① Remove ε-productions:

C, A, S are nullable.

$N_0 = \{C\}$

$N_1 = \{C, A\}$

$N_2 = \{C, A, S\}$

$N_3 = N_2$

G': $S \to ACA \mid CA \mid AA \mid AC \mid A \mid C$

$A \to aAa \mid aa \mid B \mid C$

$B \to bB \mid b$

$C \to cC \mid c$

② Remove chain rules:

$CHAIN(S) = \{C, A, B\}$

$CHAIN(A) = \{B, C\}$

G'': $S \to ACA \mid CA \mid AA \mid AC \mid aAa \mid aa \mid cC \mid c \mid bB \mid b$

$A \to aAa \mid aa \mid cC \mid c \mid bB \mid b$

$B \to bB \mid b$

$C \to aC \mid c$

③ Remove useless symbols: No change. (Why?)

④ Convert to CNF:

Step 1:

G''': $S \to ACA \mid CA \mid AA \mid AC \mid D_aAD_a \mid D_aD_a \mid D_cC \mid c \mid D_bB \mid b$

$A \to D_aAD_a \mid D_aD_a \mid D_cC \mid c \mid D_bB \mid b$

$B \to D_bB \mid b$

$C \to D_cC \mid c$

$D_a \to a$

$D_b \to b$

$D_c \to c$

Step 2:

G''': $S \to CA \mid AA \mid AC \mid D_aD_a \mid D_cC \mid c \mid D_bB \mid b$

$S \to AE_1 \quad E_1 \to CA$

$S \to D_aF_1 \quad F_1 \to AD_a$

$A \to D_aD_a \mid D_cC \mid c \mid D_bB \mid b$

$A \to D_a G_1 \quad G_1 \to A D_a$

$B \to D_b B \mid b$

$C \to D_c C \mid c$

$D_a \to a$

$D_b \to b$

$D_c \to c$

Example: G: $S \to ABCBCD$

$A \to CD$

$B \to Cb$

$C \to A \mid \varepsilon$

$D \to bD \mid \varepsilon$

$N_0 = \{C, D\}$

$N_1 = \{C, D, A\}$

$N_2 = N_1$

$\Rightarrow$ C, D, A are nullable.

① Remove ε-productions:

G': $S \to ABCBCD \mid BCBCD \mid ABBCD \mid ABCBD \mid ABCBC \mid$

$BBCD \mid BCBD \mid BCBC \mid ABBD \mid ABBC \mid ABCB$

$BBD \mid BBC \mid BCD \mid ABB \mid BB$

$A \to CD \mid C \mid D$

$B \to Cb \mid b$

$C \to A$

$D \to bD \mid b$

② Remove chain rules:

CHAIN(A) = {C, D}

CHAIN(C) = {A, D}

G'': S → ABCBCD | BCBCD | ABBCD | ABCBD | ABCBC |
BBCD | BCBD | BCBC | ABBD | ABBC | ABCB
BBD | BBC | BCD | ABB | BB

A → CD | bD | b

B → Cb | b

C → CD | bD | b

D → bD | b

③ Remove useless symbols: No change.

④ Convert to CNF

Step 1:

G''': S → ABCBCD | BCBCD | ABBCD | ABCBD | ABCBC |
BBCD | BCBD | BCBC | ABBD | ABBC | ABCB
BBD | BBC | BCD | ABB | BB

A → CD | D_bD | b

B → Cb | b

$C \to CD \mid D_b D \mid b$

$D \to D_b D \mid b$

$D_b \to b$

Step 2:

$G^{(iv)}$: $S \to AA_1$ $A_1 \to BA_2$ $A_2 \to CA_3$ $A_2 \to BA_4$ $A_4 \to CD$

$S \to BB_1$ $B_1 \to CB_2$ $B_2 \to BB_3$ $B_3 \to CD$

$S \to AC_1$ $C_1 \to BC_2$ $C_2 \to BC_3$ $C_3 \to CD$

$S \to AD_1$ $D_1 \to BD_2$ $D_2 \to CD_3$ $D_3 \to BD$

$S \to AE_1$ $E_1 \to BE_2$ $E_2 \to CE_3$ $E_3 \to BC$

$S \to BF_1$ $F_1 \to BF_2$ $F_2 \to CD$

$S \to BG_1$ $G_1 \to CG_2$ $G_2 \to BD$

$S \to BH_1$ $H_1 \to CH_2$ $H_2 \to BC$

$S \to AJ_1$ $J_1 \to BJ_2$ $J_2 \to BD$

$S \to AK_1$ $K_1 \to BK_2$ $K_2 \to BC$

$S \to AL_1$ $L_1 \to BL_2$ $L_2 \to CB$

$S \to BM_1$ $M_1 \to BD$

$S \to BN_1$ $N_1 \to BC$

$S \to BP_1$ $P_1 \to CB$

$S \rightarrow AQ_1 \qquad Q_1 \rightarrow BB$

$S \rightarrow BB$

$A \rightarrow CD \mid D_bD \mid b$

$B \rightarrow Cb \mid b$

$C \rightarrow CD \mid D_bD \mid b$

$D \rightarrow D_bD \mid b$

$D_b \rightarrow b$