TARİH

FİRMA

KONU

1.) To prove that the class of regular languages closed under intersection, we must show that if L_1 and L_2 are regular languages, then their intersection $L_1 \cap L_2$ is also regular.

Let L_1 be a regular language recognized by DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
Let L_2 be a regular language recognized by DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Construct a new DFA M that recognizes $L_1 \cap L_2$, $M = (Q, \Sigma, \delta, q_0, F)$ produced automaton

- $Q = Q_1 \times Q_2$
- $q_0 = (q_1, q_2)$
- $\delta((q_a, q_b), a) = (\delta_1(q_a, a), \delta_2(q_b, a))$
- $F = \{(p, q) \in Q_1 \times Q_2 \mid p \in F_1 \text{ and } q \in F_2\}$

This automaton simulates both M_1 and M_2 in parallel. It only accepts a string if both automata would accept it.

Since we can construct a DFA for $L_1 \cap L_2$, and DFAs recognizes regular languages, $L_1 \cap L_2$ is regular. Therefore, the class of regular languages closed under intersection.

2.)

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

$R = (a \cup bc^*d)^* bc^*$ * should have at least one "b"

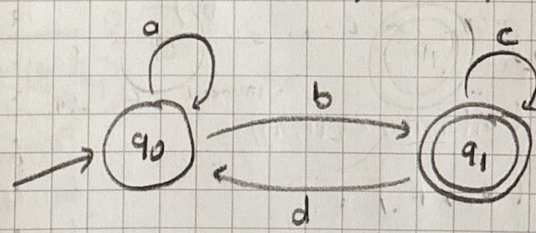
$L(R) = \{aab, bc, bdb, abdbc, b \dots\}$

$Q = \{q_0, q_1\}$

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

start state : q_0

$F = \{q_1\}$



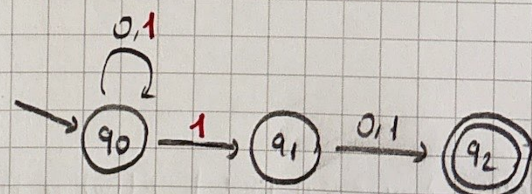
$\delta =$

	a	b	c	d
q_0	q_0	q_1	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	q_1	q_0

b.) $R = (0 \cup 1)^* 1 (0 \cup 1)$

$L(R) = \{10, 11, 010, 111, 110, \dots\}$

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



$Q = \{q_0, q_1, q_2\}$

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

start state q_0

$F = \{q_2\}$

$\delta =$

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

transition from q_0 to q_1 non deterministically

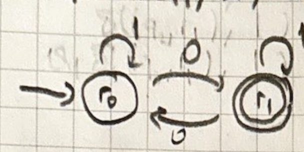
3.) Let M be a NFA for the set of all binary strings that have either the number of 0s odd, or the number of 1s not a multiple of 3, or both.

a.)

- We can build two automata, and then combine them using the union operation.

$L_1 = \{ w \in \{0,1\}^* \mid \#0s \text{ in } w \text{ is odd} \}$, $L_1 = L(M_1)$

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$



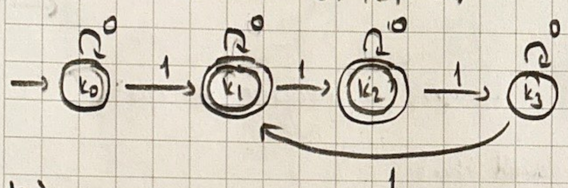
$Q_1 = \{r_0, r_1\}$
$\Sigma = \{0,1\}$
$q_1 = r_0$
$F_1 = \{r_1\}$

$\delta_1 =$

	0	1
r_0	r_1	r_0
r_1	r_0	r_1

$L_2 = \{ w \in \{0,1\}^* \mid \#1 \text{ is } w \text{ is not multiple of } 3 \}$, $L_2 = L(M_2)$

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$



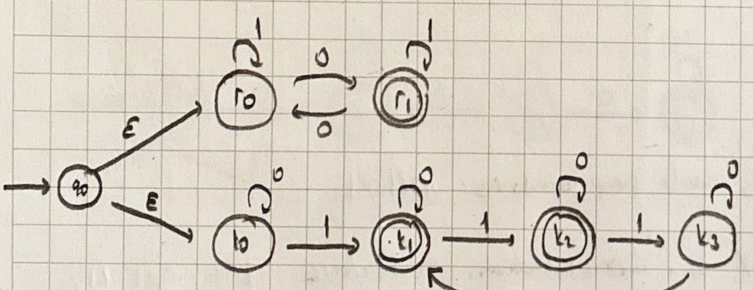
$Q_2 = \{k_0, k_1, k_2, k_3\}$
$\Sigma = \{0,1\}$
$q_2 = k_0$
$F_2 = \{k_1, k_2\}$

$\delta_2 =$

	0	1
k_0	k_0	k_1
k_1	k_1	k_2
k_2	k_2	k_3
k_3	k_3	k_1

b.)

Let's define $M = (Q, \Sigma, \delta, q_0, F)$, $M = M_1 \cup M_2$



$Q = \{q_0, r_0, r_1, k_0, k_1, k_2, k_3\}$
$\Sigma = \{0,1\}$
$q_0 = q_0$
$F = \{r_1, k_1, k_2\}$

$\delta =$

	0	1	ε
q_0	$\emptyset$	$\emptyset$	$\{r_0, k_0\}$
r_0	$\{r_1\}$	$\{r_0\}$	$\emptyset$
r_1	$\{r_0\}$	$\{r_1\}$	$\emptyset$
k_0	$\{k_0\}$	$\{k_1\}$	$\emptyset$
k_1	$\{k_1\}$	$\{k_2\}$	$\emptyset$
k_2	$\{k_2\}$	$\{k_3\}$	$\emptyset$
k_3	$\{k_3\}$	$\{k_1\}$	$\emptyset$

c.) M is the combination of DFAs M_1 and M_2, which are recognizes the languages L_1 and L_2, and M is capable of recognizing L_1 and L_2. Therefore A is not the only language that M recognizes

d.) $w = 0101101$

$\delta(q_0, \varepsilon) = k_0$
$\delta(k_0, 0) = k_0$
$\delta(k_0, 1) = k_1$
$\delta(k_1, 0) = k_1$
$\delta(k_1, 1) = k_2$
$\delta(k_2, 1) = k_3$

$\delta(k_3, 0) = k_3$
$\delta(k_3, 1) = k_1$

$k_1 \in F$, therefore $w \in A$

e.) Yes it's possible.

- Let R_1 is the regular expression for $L_1 = \{ w \in \{0,1\}^* \mid \#0(w) \text{ is odd} \}$

$R_1 = \underbrace{(1^* 0 1^* 0)^*}_{\text{even part}} (1^* 0 1^*)$

- Let R_2 is the regular exp. for $L_2 = \{ w \in \{0,1\}^* \mid \#1(w) \not\equiv 0 \bmod 3 \}$ #of 1s in w

$R_2 = \underbrace{(0^* 1 0^* 1 0^* 1)^*}_{\#1 \text{ is } 3} \underbrace{(0^* 1 \cup 0^* 1 0^* 1)}_{\#1 \text{ is } 1 \text{ or } 2} 0^*$

since we can create a NFA recognizing A, we can also define R which is the regular exp of R.

$R = R_1 \cup R_2$

$= (1^* 0 1^* 0)^* (1^* 0 1^*) \cup (0^* 1 0^* 1 0^* 1)^* (0^* 1 \cup 0^* 1 0^* 1) 0^*$

4.) $L = \{ ww \mid w \in \{0,1\}^* \}$

- We use Pumping Lemma to prove or disprove that a language is regular or not.
- Pumping Lemma says that it can divide $s = xyz$ such that
  1.) $xy^iz \in A$ for all $i \geq 0$
  2.) $y \neq \varepsilon$
  3.) $|xy| \leq p$

Proof by Contradiction:

Assume L is regular

Let $s = 0^p 1 0^p 1 \in L$

can't pump satisfying $0^p 1 0^p 1$

- $xyyz \notin L$. Contradiction. Therefore L is not a Regular language.

$s = \underbrace{000}_{x}\underbrace{0...}_{y}\underbrace{1\,0000...1}_{z}$

$|xy| \leq p$

5.)

a.) $R = 00^* 11^+$

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

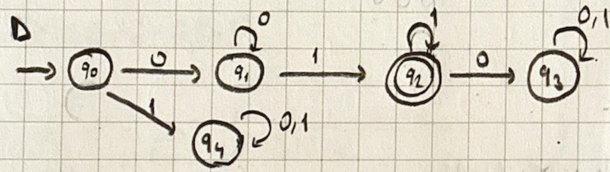
G:

$S \longrightarrow 0F1$
$F \longrightarrow 0F \mid F1 \mid 0 \mid 1 \mid \varepsilon$

$V = \{S, F\}$
$\Sigma = \{0, 1\}$
R = 6 rules given left
$S = S$

b.) We can define a CFG for RL corresponding the complement of $R = 00^* 11^+$ by:
1- construct a DFA D for R
2- Take the complement of D
3- Define CFG G by using the transition rules in $\overline{F}$.

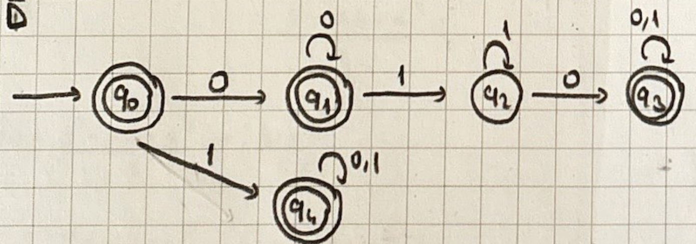
D



note: in order to take a compliment of a DFA, it should be complete. Therefore states q_3 and q_4 added as dead states.

take the complement by replacing final states with non-final states, and vice versa.

$\overline{D}$



$\overline{D} = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = \{0, 1\}$
$q_0 = q_0$
$F = \{q_0, q_1, q_3, q_4\}$

δ	0	1
q_0	q_1	q_4
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_3
q_4	q_4	q_4

dead states

use the transition rules in $\overline{\delta}$

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

$V = \{Q_0, Q_1, Q_2, Q_3, Q_4\}$
$\Sigma = \{0, 1\}$
$S = Q_0$
R:

R:

$Q_0 \longrightarrow 0Q_1 \mid 1Q_4 \mid \varepsilon$
$Q_1 \longrightarrow 0Q_1 \mid 1Q_2 \mid \varepsilon$
$Q_2 \longrightarrow 0Q_3 \mid 1Q_2 \mid$
$Q_3 \longrightarrow 0Q_3 \mid 1Q_3 \mid \varepsilon$
$Q_4 \longrightarrow 0Q_4 \mid 1Q_4 \mid \varepsilon$

(final states includes ε)

* sample derivation of $w = 000110$:

$Q_0 \Rightarrow 0Q_1$
$\Rightarrow 00Q_1$
$\Rightarrow 000Q_1$
$\Rightarrow 0001Q_2$
$\Rightarrow 00011Q_2$
$\Rightarrow 000110Q_3$
$\Rightarrow 000110$

6.) Consider a nondeterministic PDA that has
- a single stack
- can move input head in both directions on input tape
- capable of detecting when its input head is at <u>either</u> end of its input tape
- accepts by entering an accept state.

$L = \{ a^i b^i c^i \mid i \geq 0 \}$

001122

- Step 1: Count a's
  - Start at the left end of the tape (input)
  - For each a:
    - move right.
    - push X on the stack.
  - Stop when the first b is found.
  - reject if encountered with c.
- Step 2: Match b's
  - For each b:
    - check if the stack is empty or not
    - if it's empty, reject (too many b's)
    - if it's not, pop one X from the stack and move right
  - Stop when the first c is found. reject if encountered with a
- Step 3: Move Back to Left End
  - repeat step 1 and count a's
- Step 4: Move to start of c's
  - move right skipping a's and b's, reach first c
  - For each c:
    - if stack is empty, reject (too many c's)
    - if stack is not empty, pop one X from the stack and move right
  - If any other symbol than c is encountered, reject
  - If the end of input tape is reached and the stack is empty accept, otherwise reject.