Exercise Sheet Exercise 1: Pavel Ghazaryan
March 14th, 2022 ID: 10756505

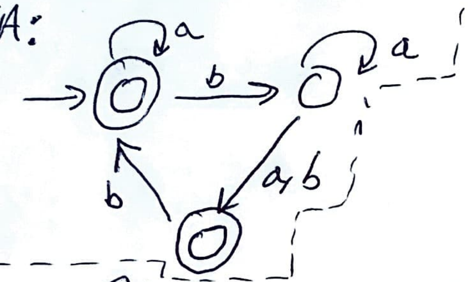
Exercise 1

a) Accepted: aba; babb; bbbb; ababb

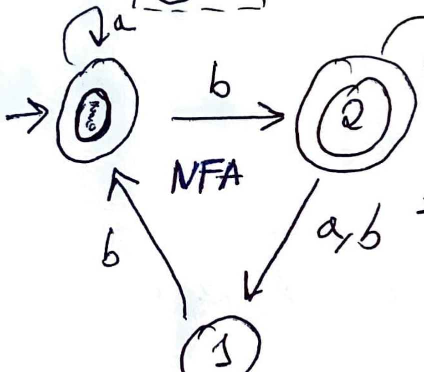
b) $(\epsilon \mid a^* \mid a^* b a^* (a|b) \mid a^* b a^* (a|b) b a^*)$

In order to see that this is true, we can make all non-accepting states into accepting ones and accepting state into non-accepting. Afterwards, we can just read of the patterns from

NFA:

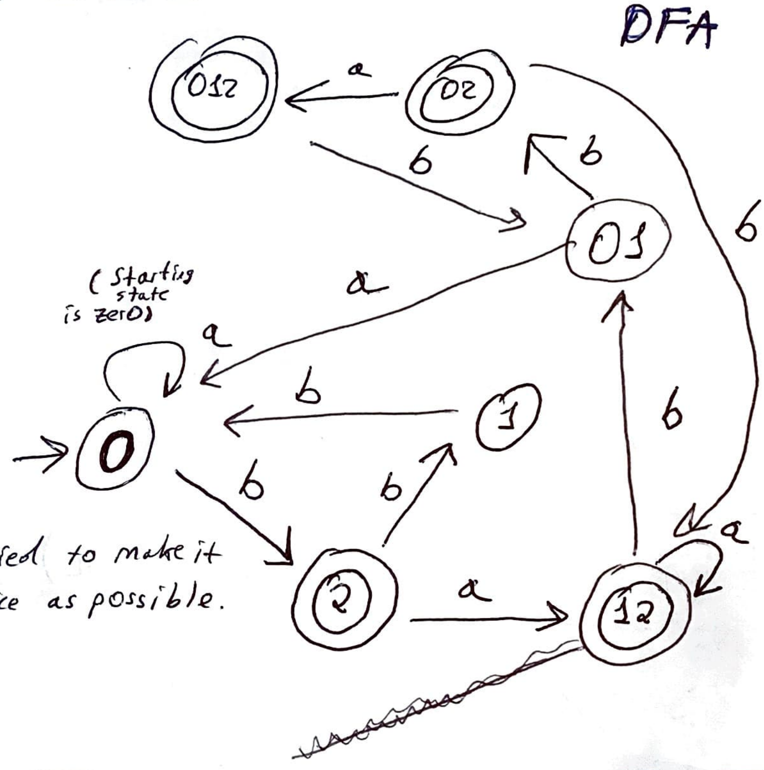


c)



⇒

DFA



I tired to make it as nice as possible.

Exercise 2

a) a simulation from A to B exists:

(X; 0) (Y; 2)

(X; 1) (Y; 3)

b) Exists: (0; X) (3; Y)

(1; X)

(2; Y)

Exercise 3

$(p_1(p_2|p_3))$: In order to prove this we need to analyze each statement separately. In the first statement we have s_1 which matches p_1; s_2 which matches p_2 or p_3 or both. The final pattern is the concatenation of s_1 and s_2 being s.

$((p_1p_2)|(p_1p_3))$: Here we have the s_1 to match p_1; s_2 match p_2 and p_3 match s_3. So the final s matches either s_1 concat s_2 or s_1 concat s_3 or both. Meaning the difference again is ending with p_2 or p_3. So we can see that logically the two regex match the same words.

We can also use ~~propositional~~ distributive law for opening alternative brackets with concatenation.

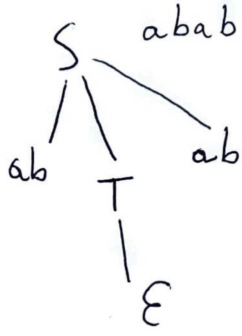
$p_1 \cdot (p_2|p_3) = (p_1p_2)|(p_1p_3)$ so basically the second statement is equivalent to the first one.

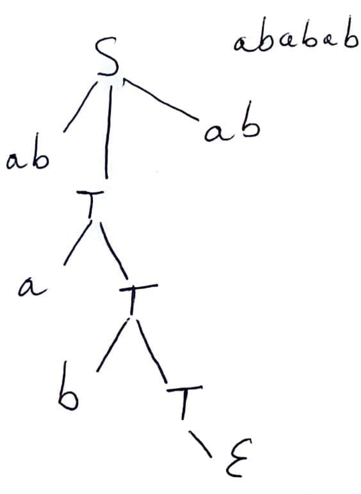
Exercise 4

$\Sigma = \{a, b\}$ a) $= \{S, T\}$

$S \to abTab \mid ab$

$T \to aT \mid bT \mid \varepsilon$





It doesn't generate invalid strings baab, bb as it cannot start with anything else besides ab.

b) $\{a^i b^j c^k \mid i,j,k \in \mathbb{N},\ j \geq i+k\}$ $\Sigma = \{a,b,c\}$

$\Xi = \{S, T, Q, N\}$

$S \to QT \mid \varepsilon$

$T \to bTc \mid N$

$Q \to aQb \mid \varepsilon$

$N \to bN \mid \varepsilon$

This grammar is non-ambiguous as the main way of generating the words is from left to right. It is like that as in Start symbol I have the concatenation of two non-terminal symbols. Furthermore, in Q and T no word can be generated in two ways as every time the letters are put in the middle. Also in order to have no problem with repetitive bs. I have used one more symbol which allows to add any number of bs without causing ambiguity.

Exercise 5

a) $|w_a| > |w_b|$

(ii) For this language we cannot produce a DFA. The reason is that we need to count the number of a's and the number b's and remember what they were in order to make sure that we have more a's than b's. Even if we try to build a pattern with or's (aab|aaab|abaa|...) there is going to be an infinite of them.

Answer: DFA cannot be constructed.

b) $|w_a| \bmod 2 = |w_b| \bmod 2$

(i) DFA: